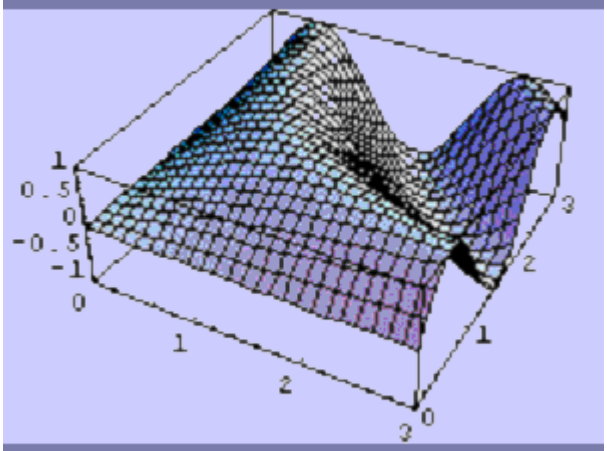
Breve manual de MATHEMATICA 5.1



Robert Ipanaqué Chero
Ricardo Velesmoro León

BREVE MANUAL DE *MATHEMATICA* 5.1

Robert Ipanaqué Chero

Ricardo Velesmoro León

DEPARTAMENTO ACADÉMICO DE MATEMÁTICA
UNIVERSIDAD NACIONAL DE PIURA, PERÚ

ISBN: 84-689-6093-4

editado por eumed.net

BREVE MANUAL DE *MATHEMATICA* 5.1®

Robert Ipanaqué Chero
robertchero@hotmail.com
http://www.unp.edu.pe/pers/ripanaque

Ricardo Velesmoro León
velesmoro_ricardo@hotmail.com

Departamento Académico de Matemática
Universidad Nacional de Piura, Urb. Miraflores s/n, Castilla - Piura, Perú

ISBN:
Nº Registro:

Setiembre 2005, editado por el grupo **eumed•net**
Grupo de Investigación de la Universidad de Málaga, España
http://www.eumed.net

Para citar este libro puede utilizar el siguiente formato:
Ipanaqué Chero y Velesmoro León (2005) Breve Manual de Mathematica 5.1. Edición a texto completo en www.eumed.net/libros/2005/

Índice

Breve manual de *Mathematica* 5.1

Este libro da una introducción a *Mathematica*, concentrándose en usar *Mathematica* como sistema interactivo para resolver problemas. Cuando lo haya leído, debe tener suficiente conocimiento de *Mathematica* para abordar muchas clases de problemas prácticos.

1. Funcionamiento de Mathematica

Para saber cómo se instala y funciona *Mathematica* debe leer la documentación que vino con su copia de *Mathematica*. Los detalles difieren a partir de un sistema informático a otro, y son afectados por las varias clases de arreglo para requisitos particulares que se pueden hacer en *Mathematica*. Sin embargo, esta sección delinea dos casos comunes.

Observe que aunque los detalles de funcionamiento de *Mathematica* difieren a partir de un sistema informático a otro, la estructura de los cálculos de *Mathematica* es igual en todos los casos. El usuario digita la entrada (input), *Mathematica* la procesa, y devuelve un resultado.

1.1. Interfase de Cuaderno

utilice un icono o el menú de Inicio	formas gráficas de inicializar *Mathematica*
`mathematica`	comando para inicializar *Mathematica*
finalizar texto con <Shift>+<Enter>	entrada para *Mathematica*
elegir el ítem salida del menú	salir de *Mathematica*

Funcionamiento de *Mathematica* con una interfase de cuaderno.

En una interfase de “cuaderno”, usted interactúa con *Mathematica* creando documentos interactivos.

Si usa su computadora vía una interfase puramente gráfica haremos como de costumbre doble en el icono de inicio de *Mathematica*. En cambio, si la usa vía un sistema operativo basado en texto digitaremos el comando `mathematica` para iniciar *Mathematica*.

Cuando *Mathematica* inicializa usualmente presenta un cuaderno en blanco. Usted digita la entrada (input), luego presiona (en simultáneo) <Shift>+<Enter> para que *Mathematica* procese su entrada. <Shift>+<Enter> indica a *Mathematica* que usted ha finalizado su entrada. Si su teclado posee teclado numérico puede usar la tecla <Enter> del mismo en lugar de <Shift>+<Enter>.

Después de ejecutar una entrada en *Mathematica* desde un cuaderno, *Mathematica* etiquetará su entrada con **`In[`*`n`*`]:=`**. También etiqueta la correspondiente salida `Out[`*`n`*`]=`.

Usted digita `1+1`, luego finaliza su entrada con <Shift>+<Enter>. *Mathematica* procesa la entrada, luego agrega la etiqueta a la entrada **`In[1]:=`**, y devuelve la respectiva salida.

A lo largo de este libro los "diálogos" con *Mathematica* se mostrarán de la siguiente manera:

Con una interfase de cuaderno, usted sólo digita `1+1`. *Mathematica* añade la etiqueta **`In[1]:=`**, e imprime el resultado.

```
1 + 1
2
```

Recuerde que los cuadernos corresponden a la "parte visible" ("front end") de *Mathematica*. El núcleo de *Mathematica* que realiza realmente los cálculos puede funcionar en la misma computadora que el front end, o en otra computadora conectada vía alguna clase de red o de línea. En la mayoría de los casos, el núcleo incluso no se inicializa hasta que el usuario hace realmente un cálculo con *Mathematica*.

Para salir de *Mathematica*, usted elige el ítem salida del respectivo menú en la interfase de cuaderno.

1.2. Interfase Basada en Texto

`math`	comando del sistema operativo para inicializar *Mathematica*
finalizar texto con <Enter>	entrada para *Mathematica*
Control-D, Control-Z ó `Quit[]`	salir de *Mathematica*

Funcionamiento de *Mathematica* con una interfase basada en Texto.

Con una interfase basada en texto, usted interactúa con su computadora digitando texto en el teclado.

Para inicializar *Mathematica* con una interfase basada en texto, se digita el comando `math` en el prompt del sistema operativo. En algunos sistemas, también es posible inicializar *Mathematica* con una interfase basada en texto haciendo doble clic en el icono del núcleo de *Mathematica*.

Cuando *Mathematica* ha inicializado, imprimirá el prompt **`In[1]:=`**, esto significa que esta lista para que usted haga su entrada. Puede entonces digitar su entrada, terminándola con <Enter>.

*Mathematic*a procesa la entrada, y genera un resultado, el mismo que etiquetará con `Out[1]=`.

A lo largo de este libro los "diálogos" con *Mathematica* se mostrarán de la siguiente manera:

La computadora imprime **`In[1]:=`**. Usted sólo digita `1+1`. La línea que empieza con `Out[1]=` es el resultado de *Mathematica*.

```
In[1]:= 1 + 1
Out[1]= 2
```

Observe que usted no digita explícitamente el prompt **`In[1]:=`**; sólo digita el texto que sigue después de este aviso.

Observe también que la mayor parte de los diálogos dados en el libro muestran salidas en la forma que usted obtendría con una interfase de cuaderno de *Mathematica*; la salida con una interfase basada en texto luce similar, pero carece de características tales como caracteres especiales y cambio de tamaño de fuente.

Para salir de *Mathematica*, digite Control-D, Control-Z ó `Quit[ ]` en el prompt de la entrada.

2. Cálculos numéricos

2.1. Aritmética

Usted puede hacer aritmética con *Mathematica* tal y como lo haría con una calculadora.

Aquí tenemos la suma de dos números.

```
5.6 + 3.7
9.3
```

Con * indicamos el producto de dos números.

```
5.6 * 3.7
20.72
```

Con el espacio en blanco también se indica el producto de dos números.

```
5.6 3.7
20.72
```

Usted puede digitar operaciones aritméticas haciendo uso de los paréntesis.

```
(2 + 3) ^ 3 – 4 (6 + 7)
73
```

Los espacios no son necesarios, aunque a menudo hacen su entrada más fácil de leer.

```
(2+3)^3–4(6+7)
73
```

x^*y*	potencia
–*x*	menos
x/*y*	división
x y z o *x***y***z*	producto
x+*y*+*z*	suma

Operaciones aritméticas en *Mathematica*.

Las operaciones aritméticas en *Mathematica* se agrupan de acuerdo con las convenciones estándares de la matemática. Como es usual, `2+3/7`, por ejemplo, significa `2+(3/7)`, y no `(2+3)/7`. Usted puede controlar siempre la forma de agrupar explícitamente usando paréntesis.

Este resultado se da en notación científica.

```
3.7 ^ 36
```

2.85274×10^{20}

Usted puede incorporar números en notación científica de esta forma.

```
4.6 10^45
```

4.6×10^{45}

O de esta otra forma.

```
4.6*^45
```

4.6×10^{45}

2.2. Resultados exactos y aproximados

Una calculadora electrónica hace todos sus cálculos con una precisión determinada, digamos de diez dígitos decimales. Con *Mathematica*, en cambio, usted puede obtener resultados *exactos*.

Mathematica da un resultado *exacto* para 2^{300}, a pesar que éste tiene 91 dígitos decimales.

```
2^300
```

```
2037035976334486086268445688409378161051468393665936250636140449354381299763336706183397376
```

Usted puede pedir a *Mathematica* que devuelva un resultado aproximado, tal como lo daría una calculadora, para ello debe finalizar su entrada con `//N`.

Esto da un resultado numérico aproximado.

```
2^300//N
```

2.03704×10^{90}

Mathematica puede dar resultados en términos de números racionales.

```
1/3 + 2/7
```

$$\frac{13}{21}$$

`//N` siempre da un resultado numérico aproximado.

```
1/3 + 2/7//N
0.619048
```

exp `//N`	da un valor numérico aproximado para *exp*

Obteniendo aproximaciones numéricas.

Cuando usted digita un entero como 7, *Mathematica* asume que es exacto. Si usted digita un número como 4.5, con un punto decimal explícito, *Mathematica* asume que desea efectuar cálculo numéricos aproximados.

Esto es tomado como un número racional exacto, y es llevado a una fracción irreducible.

```
26/78
```

$$\frac{1}{3}$$

Cuando usted digita un número con un punto decimal explícito, *Mathematica* produce un resultado numérico aproximado.

```
26.7/78
0.342308
```

Aquí otra vez, la presencia del punto decimal hace que *Mathematica* dé un resultado numérico aproximado.

```
26./78
0.333333
```

Cuando cualquier número en una expresión aritmética es digitado con un punto decimal explícito, usted obtiene un resultado numérico aproximado para toda la expresión.

```
5. + 9 / 78 - 5/8
4.49038
```

2.3. Algunas funciones matemáticas

Mathematica incluye una gran colección de funciones matemáticas. A continuación mencionamos las más comunes.

`Sqrt[`*x*`]`	raíz cuadrada ($\sqrt{x}$)
`Exp[`*x*`]`	exponencial (e^x)
`Log[`*x*`]`	logaritmo natural ($\log_e x$)
`Log[`*b*`, `*x*`]`	logaritmo en base *b* ($\log_b x$)
`Sin[`*x*`], Cos[`*x*`], Tan[`*x*`], Cot[`*x*`], Sec[`*x*`], Csc[`*x*`]`	funciones trigonométricas (con argumentos en radianes)
`ArcSin[`*x*`], ArcCos[`*x*`], ArcTan[`*x*`], ArcCot[`*x*`], ArcSec[`*x*`], ArcCsc[`*x*`]`	funciones trigonométricas inversas
n`!`	factorial de *n* (producto de los enteros 1, 2, ..., *n*)
`Abs[`*x*`]`	valor absoluto
`Round[`*x*`]`	redondeo del entero *x*
`Mod[`*n*`, `*m*`]`	*n* módulo *m* (resto de la división de *n* entre *m*)
`Random[ ]`	número seudo aleatorio entre 0 y 1
`Max[`*x*`, `*y*`, ...], Min[`*x*`, `*y*`, ...]`	máximo, mínimo de *x*, *y*, ...
`FactorInteger[`*n*`]`	factores primos de *n*

Algunas de las funciones matemáticas más comunes.

- Los argumentos de todas las funciones en *Mathematica* se colocan entre *corchetes*.
- Los nombres de las funciones incorporadas en *Mathematica* empiezan con *letra mayúscula*.

Dos puntos importantes acerca de funciones en *Mathematica*.

Es importante recordar que todos los argumentos de funciones se colocan entre corchetes, no entre paréntesis. Los paréntesis en *Mathematica* se usan solamente para indicar agrupación de términos, y jamás para encerrar argumentos de funciones.

Esto da $\log_e(15.7)$. Note la letra mayúscula para `Log`, y los corchetes para el argumento.

```
Log[15.7]
2.75366
```

Esto devuelve $\sqrt{64}$ como un entero exacto.

```
Sqrt[64]
8
```

Esto da un valor numérico aproximado para $\sqrt{6}$.

```
Sqrt[6]//N
2.44949
```

La presencia explícita de un punto decimal le indica a *Mathematica* que dé un resultado numérico aproximado.

```
Sqrt[6.]
2.44949
```

En este caso *Mathematica* devuelve un número en forma simbólica exacta.

```
Sqrt[6]
```

$\sqrt{6}$

Aquí tenemos un resultado entero exacto para 30×29× ... ×1. Usted puede digitar números grandes para calcular factoriales. Por ejemplo, puede calcular 2000! en corto tiempo.

```
40!
815915283247897734345611269596115894272000000000
```

Esto da un valor numérico aproximado del factorial.

```
40!//N
8.15915×10^47
```

`Pi`	$\pi \approx 3.14159$
`E`	$e \approx 2.71828$ (normalmente aparece como)
`Degree`	$\pi/180$: factor de conversión de grados a radianes (normalmente aparece como °)
`I`	$i = \sqrt{-1}$ (normalmente aparece como)
`Infinity`	∞

Algunas constantes matemáticas comunes.

Note que todos los nombres de las constantes incorporadas en *Mathematica* empiezan con mayúscula.

Este es el valor numérico de π^2.

```
Pi^2//N
9.8696
```

Esto devuelve un resultado exacto para sen(π/2).Note que los argumentos de las función trigonométricas siempre se dan en radianes.

```
Sin[Pi/2]
1
```

Esto devuelve el valor numérico de sen(20°). Multiplicando por la constante `Degree` convertimos el argumento a radianes.

```
Sin[20 Degree]//N
0.34202
```

`Log[`*x*`]` devuelve el logaritmo de *x* en base *e*.

```
Log[E^15]
15
```

Usted puede obtener logaritmos en cualquier base *b* usando `Log[`*x*`]`. Como una notación estándar de *Mathematica* la *b* es opcional.

```
Log[3,81]
4
```

2.4. Cálculos con precisión arbitraria

Cuando usted utiliza `//N` para obtener un resultado numérico, *Mathematica* hace que lo que haría una calculadora estándar: devuelve el resultado con un número fijo de cifras significativas. No obstante, usted puede indicarle a *Mathematica* las cifras significativas con las que desea operar. Esto permite obtener resultados numéricos en *Mathematica* con cualquier grado de precisión.

exp `//N` o `N[`*exp*`]`	valor numérico aproximado para *exp*
`N[`*exp*`, `*n*`]`	valor numérico de *exp* calculado con *n* dígitos de precisión

Funciones de evaluación numérica.

Esto devuelve el valor numérico de π con un número fijo de cifras significativas. Digitar `N[Pi]` es equivalente a `Pi//N`.

```
N[Pi]
3.14159
```

Esto devuelve π con 50 dígitos.

```
N[Pi, 50]
3.1415926535897932384626433832795028841971693993751
```

Aquí tenemos a $\sqrt{7}$ con 40 dígitos.

```
N[Sqrt[7], 40]
2.645751311064590590501615753639260425710
```

Al realizar cualquier tipo de cálculo numérico puede introducir pequeños errores de redondeo en sus resultados. Cuando se aumenta la precisión numérica estos errores se hacen más pequeños. Asegurarse que usted obtiene la misma respuesta al aumentar la precisión numérica es a menudo una buena forma de verificar los resultados.

La cantidad $e^{\pi\sqrt{163}}$ esta bastante próxima a ser entera. Para verificar que el resultado no es, de hecho, un entero, usted tiene que usar la precisión numérica suficiente.

```
N[Exp[Pi Sqrt[163]], 40]
2.625374126407687439999999999992500725972×10^17
```

2.5. Números complejos

Puede ingresar números complejos en *Mathematica* con sólo incluir la constante `I`, igual a $\sqrt{-1}$. Asegúrese que la letra `I` sea mayúscula. Si está usando cuadernos, también puede ingresar `I` como i digitando <Esc> ii <Esc>. La forma i es la que se usa normalmente como salida. Note que una `i` ordinaria significa una variable llamada *i*, pero no $\sqrt{-1}$.

Esto devuelve como resultado el número imaginario 2*i*.

```
Sqrt[-4]
2 i
```

Esto devuelve la división de dos números complejos.

```
(8 + 4 I)/(-2 + I)
-2 - 6ⅈ
```

Aquí tenemos el valor numérico de un exponencial complejo.

```
Exp[11 + 5 I]//N
16984. - 57414.8ⅈ
```

x + `I` y	el número complejo $x + i\,y$
`Re[`z`]`	parte real
`Im[`z`]`	parte imaginaria
`Conjugate[`z`]`	complejo conjugado z^* o $\bar{z}$
`Abs[`z`]`	módulo de z
`Arg[`z`]`	el argumento φ en $\lvert z\rvert e^{i\varphi}$

Operaciones con números complejos.

2.6. Acostumbrándose a *Mathematica*

- Los argumentos de las funciones se colocan entre *corchetes*.
- Los nombres de las funciones incorporadas empiezan con *letra mayúscula*.
- La multiplicación se puede representar por un espacio.
- Las potencias se denotan por ^.
- Los números en la notación científica se ingresan, por ejemplo, como `2.5*^-4` ó `2,5 10^-4`.

Puntos importantes a recordar en *Mathematica*.

Esta sección le ha dado una primera idea de *Mathematica*. Si ha utilizado otros sistemas informáticos antes, habrá advertido probablemente algunas similitudes y algunas diferencias. A menudo encontrará en las diferencias las partes más difíciles de recordar. Esto puede ayudarle, sin embargo, a entender un poco de por qué *Mathematica* funciona en la manera que lo hace, y por qué existen tales diferencias.

Una característica importante de *Mathematica* que difiere de otros lenguajes de programación, y de la anotación matemática convencional, es que los argumentos de una función se encierran en corchetes, no en paréntesis. Los paréntesis en *Mathematica* se reservan específicamente para indicar la agrupación de términos. Hay obviamente una diferencia conceptual entre dar argumentos a una función y

agrupar términos entre sí; el hecho que la misma notación a menudo se ha utilizado para ambas cosas es en gran parte una consecuencia de la tipografía y de los primeros teclados de computadora. En *Mathematica*, los conceptos son distinguidos por la diferente notación.

Esta diferencia tiene varias ventajas. En la notación de paréntesis, no está claro si $c(1+x)$ significa `c[1 + x]` ó `c*(1 + x)`. Usar los corchetes para los argumentos de una función quita esta ambigüedad. También permite que la multiplicación sea indicada sin el explícito `*` o algún otro caracter. Por consiguiente, *Mathematica* puede manejar expresiones como `2x` y `ax` ó `a(1 + x)`, tratándolos tal como en la notación matemática estándar.

Habrá visto en esta sección que las funciones incorporadas de *Mathematica* tienen a menudo nombres absolutamente largos. Puede preguntarse por qué, por ejemplo, la función del número pseudaleatorio se llama Random, en lugar de, por ejemplo, rand. La respuesta, que depende mucho del diseño de *Mathematica*, es la consistencia. Hay una convención general en *Mathematica* que todos los nombres de funciones estén representados con palabras inglesas completas, a menos que haya una abreviatura matemática estándar para ellos. La gran ventaja de este esquema es que es fiable. Una vez que usted sabe lo que una función hace, usted por lo general será capaz de adivinar exactamente cual es su nombre. Si los nombres fueran abreviados, tendría que recordar siempre cual abreviatura de las palabras inglesas estándares fue utilizada.

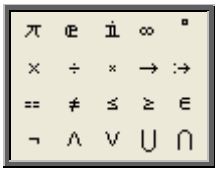
Otra característica de los nombres de funciones incorporadas en *Mathematica* es que todos comienzan con mayúscula. En secciones posteriores, usted verá cómo definir variables y funciones por si mismo. La convención de la mayúscula hace fácil distinguir objetos incorporados. Si *Mathematica* utilizara `max` en vez de `Max` para representar la operación de encontrar el máximo, después usted nunca podría utilizar `max` como el nombre de una de sus variables. Además, cuando usted lee los programas escritos en *Mathematica*, las mayúsculas de los nombres de objetos incorporados los hace más fáciles seleccionar.

2.7. Notaciones matemáticas en cuadernos

Si usa una interfase basado en texto con *Mathematica*, entonces las entradas que usted vaya a hacer consistirán solamente en caracteres que digite directamente con su teclado. Pero si usa una interfase de cuaderno entonces es posible que realice otro tipo de entradas.

Usualmente se incorporan paletas las cuales operan como extensiones de su teclado, y poseen botones sobre los que usted puede hacer clic para ingresar alguna forma en particular. Puede acceder a las paleta estándares usando el submenú Pelettes del menú File.

Haciendo clic en el botón π de esta paleta ingresará pi en su cuaderno.



Haciendo clic en el primer botón una estructura en blanco par ingresar una potencia. Usted puede usar el mouse para rellenar esta estructura o desplazarse en ella con la tecla <Tab>.

También puede hacer ingresos como los anteriores mediante teclas especiales de su teclado.

ESC p ESC	el símbolo π
ESC inf ESC	el símbolo ∞
ESC ee ESC	el símbolo ⅇ para constante exponencial (equivalente a `E`)
ESC ii ESC	el símbolo ⅈ para $\sqrt{-1}$ (equivalente a `I`)
ESC deg ESC	el símbolo ° (equivalente a `Degree`)
CTRL[^] or CTRL[6]	lleva al exponente de una potencia
CTRL[/]	lleva al denominador de una fracción
CTRL[0] ó CTRL[2]	lleva al subradical de una raíz cuadrada
CTRL[␣] (Control-Espacio)	regresa de un exponente, denominador o raíz cuadrada

Algunas maneras de ingresar notaciones especiales.

Esta es un cálculo ingresado mediante los caracteres comunes de un teclado.

```
N[E^3/5]
4.01711
```

Este es el mismo cálculo ingresado usando una paleta o teclas especiales.

$$N\left[\frac{e^2}{6}\right]$$

```
4.01711
```

Esta es una secuencia de las teclas que puede usarse para ingresar la entrada anterior.

esintesxctrl[^]nctrl[_] esddesx

```
4.01711
```

En un lenguaje de programación tradicional tal como C, FORTRAN, Java o Perl, la entrada que usted da siempre debe consistir de una cadena de caracteres ordinarios que se puedan digitar directamente del teclado. Pero el lenguaje de *Mathematica* también le permite dar entradas que contienen caracteres especiales, exponentes, fracciones, etcétera.

El lenguaje incorpora muchas características de la notación matemática tradicional. Pero debe comprender que el objetivo del lenguaje es proporcionar una forma exacta y constante de especificar cálculos. Y por consiguiente, no sigue todos los detalles de la notación matemática tradicional.

Sin embargo, es posible conseguir que *Mathematica* produzca salidas que imitan cada aspecto de notación tradicional matemática. Y es también posible para *Mathematica* importar texto que usa tal notación, y en cierta medida traducirlo en su propio lenguaje.

3. Cálculos

3.1. Uso de resultados previos

Al realizar cálculos, muchas veces usted necesita usar resultados previamente obtenidos. En *Mathematica*, % siempre hace referencia al último resultado.

%	el último resultado generado
%%	el penúltimo resultado
%%...% (*k* veces)	el *k*-ésimo previo resultado
%*n*	el resultado de la línea de salida `Out[`*n*`]`

Formas de hacer referencia a resultados previos.

Aquí tenemos el primer resultado.

```
In[1]:= 5^3
Out[1]= 125
```

Esto agrega 6 al último resultado.

```
In[2]:= % + 6
Out[2]= 131
```

Esto utiliza los dos últimos resultados.

```
In[3]:= 5 + % 2 - %%
Out[3]= 142
```

Hemos mencionado que las entradas y salidas en *Mathematica* son numeradas. Estos números pueden ser usados para hacer referencia a resultados previos.

Esto suma los resultados de las líneas 1 y 3.

```
In[4]:= %1 + %3
Out[4]= 267
```

Si utiliza una interfase basada en texto en *Mathematica*, entonces las líneas sucesivas de entradas y salidas aparecerán siempre en orden, como se muestra en este ítem. Sin embargo, si utiliza una interfase de cuaderno, entonces varias líneas sucesivas de entradas y salidas no necesariamente aparecen en orden. Usted puede por ejemplo "volver atrás" e insertar el cálculo siguiente dondequiera que desee en el cuaderno. Tenga en cuenta que `%` siempre invoca el último resultado que *Mathematica* generó. Éste puede o no ser el resultado que aparece inmediatamente encima de su actual posición en el cuaderno. Con una interfase de cuaderno, la única manera de saber cuándo un resultado particular fue generado es mirar la etiqueta de `Out[`*n*`]` que tiene. Como usted puede insertar y suprimir en todas partes en un cuaderno, de acuerdo a su necesidad, el ordenamiento de los resultados en el cuaderno, por lo general, no tiene ninguna relación con el orden en el cual los resultados fueron generados.

3.2. Definición de variables

Cuando usted efectúa cálculos extensos, muchas veces es conveniente dar *nombre* a los resultados intermedios. De la misma manera que en las matemáticas tradicionales o en otros lenguajes de programación, usted puede hacer esto introduciendo *variables* nombradas.

Esto inicializa el valor de la variable `x` con 6.

```
x = 6
6
```

Donde quiera que aparezca `x`, *Mathematica* la reemplaza por su valor 6.

```
x^3 - 25
191
```

Esto asigna un nuevo valor para `x`.

```
x = 11 + 5
16
```

`pi` es inicializada con el valor numérico de π con 20 dígitos de exactitud.

```
pi = N[Pi, 20]
3.1415926535897932385
```

Aquí esta el valor de `Sqrt[pi]`.

```
Sqrt[pi]
1.7724538509055160273
```

x = *value*	asigna un valor a la variable *x*
x = *y* = *value*	asigna un valor a las variable *x* e *y*
x = . o `Clear[`*x*`]`	quita cualquier valor asignado a *x*

Asignando valores a variables.

Es muy importante recordar que los valores asignados a las variables son *permanentes*. Una vez que usted ha asignado un valor a una variable en particular, el valor será almacenado hasta que usted lo remueva explícitamente. El valor, claro está, desaparecerá si usted inicia una nueva sesión con *Mathematica*.

Olvidarse de las definiciones hechas es la causa más común de errores al usar *Mathematica*. Si usted pusiera `x = 5`, *Mathematica* asume que usted siempre quiere que `x` tenga el valor `5`, hasta o a menos que usted e indique explícitamente otra cosa. Para evitar los errores, usted debe quitar los valores definidos en cuanto haya terminado de usarlos.

- Quite valores que asigne a las variables en cuanto termine de usarlos.

Un principio útil al usar *Mathematica*.

Las variables que usted define pueden tener cualquier nombre. No hay límite par longitud de sus nombres. Un inconveniente, sin embargo, es que los nombres de las variables nunca pueden empezar con números. Por ejemplo, `x3` puede ser una variable, pero `2x` significa `2*x`.

Mathematica usa letras minúsculas y mayúsculas. Hay una convención para los objetos incorporados en *Mathematica*, los nombres de todos ellos empiezan con letra mayúscula. Para evitar alguna confusión, usted siempre debe escoger nombres que empiecen con letra minúscula.

aaaaa	una variable cuyo nombre contiene sólo letras minúsculas
Aaaaa	una variable cuyo nombre empieza con una letra mayúscula (aplicable por lo general a objetos incorporados en *Mathematica*)

Convenciones para nombrar variables.

Usted puede digitar fórmulas que involucren variables en *Mathematica* de la misma manera que lo haría matemáticamente. Sin embargo, hay algunos puntos importantes que observar.

- `x y` significa `x` por `y`.
- `xy` sin espacio es la variable con nombre `xy`.
- `5x` significa `5` por `x`.
- `x^2y` significa `(x^2) y`, no `x^(2y)`.

Algunos puntos a observar cuando usamos variables en *Mathematica*.

3.3. Construcción de listas

Al realizar cálculos, a veces es conveniente agrupar ciertos objetos, y tratarlos como una sola entidad. Las *listas* proporcionan una manera de coleccionar objetos en *Mathematica*. Como veremos después las listas son muy importantes en *Mathematica*.

Una lista como `{4,7,6}` es una colección de tres objetos. Pero en muchos casos usted puede tratar esta lista como un solo objeto. Usted puede, por ejemplo, hacer aritmética con esta lista o asignarla a una variable.

Aquí hay una lista de tres números.

```
{4, 7, 6}
{4, 7, 6}
```

Esto eleva al cubo cada número de la lista, y le resta 2.

```
{4, 7, 6}^3 - 2
{62, 341, 214}
```

Esto toma la diferencia entre los correspondientes elementos de las dos listas. Las listas deben ser de la misma longitud.

```
{9.5, 8, 7} - {3, 4, 2.3}
{6.5, 4, 4.7}
```

El valor de % es el de la lista

```
%
{6.5, 4, 4.7}
```

Usted puede aplicar cualquier función matemática a la lista.

```
e^%//N
{665.142, 54.5982, 109.947}
```

Esto asigna una lista en la variable u.

```
u = {5, 3, 8.2}
{5, 3, 8.2}
```

Esto asigna una lista en la variable u.

```
(u + 1)/(u - 6)
```

$\{-6, -\frac{4}{3}, 4.18182\}$

3.4. Manipulación de los elementos de una lista

Muchas de las más potentes operaciones de manipulación de listas en *Mathematica* tratan a las listas como un solo objeto. A veces, sin embargo, extraer un conjunto individual de elementos de una lista.

Usted puede referirse a un elemento de una lista en *Mathematica* mediante su "índice". Los elementos son numerados en orden, empezando en 1.

{*a*, *b*, *c*}	una lista
Part[*list*, *i*] o *list*[[*i*]]	el *i*-ésimo elemento de *list* (el primer elemento es *list*[[1]])
Part[*list*, {*i*, *j*, …}] o *list*[[*i*, *j*, …]]	una lista formada por los elementos *i, j*, ... de *list*

Operaciones con elementos de una lista.

Esto extrae el tercer elemento de la lista.

```
{4, -1, 8, -6}[[3]]
8
```

Esto extrae una lista de elementos.

```
{4, -1, 8, -6}[[ {2, 3, 1, 2, 3, 4, 1} ]]
{-1, 8, 4, -1, 8, -6, 4}
```

Esto asigna una lista en la variable u.

```
u = {7, 2, 4, 6}
{7, 2, 4, 6}
```

Usted puede extraer elementos de u.

```
u[[3]]
4
```

Al asignar una lista a una variable, usted puede usar las listas en *Mathematica* de manera similar a los "arrays" en otros lenguajes de programación. De esta manera, por ejemplo, usted puede resetear un elemento de una lista asignando un valor a u[[*i*]].

Part[*u*, *i*] o u[[*i*]]	extrae el *i*-ésimo elemento de una lista
Part[*u*, *i*] = *value* o u[[*i*]] = *value*	resetea el *i*-ésimo elemento de una lista

Operando listas como arrays.

Aquí hay una lista.

```
u = {2, 1, 5, 7}
{2, 1, 5, 7}
```

Esto resetea el segundo elemento de la lista.

```
u[[2]] = 0
0
```

Ahora la lista asignada a `u` se ha modificado.

```
u
{2, 0, 5, 7}
```

3.5. Las cuatro formas de agrupar en *Mathematica*

En las secciones anteriores hemos introducido cada una de las formas de agrupar en *Mathematica*. Cada una de ellas tiene diferente significado. Es importante que usted los recuerde.

(*terms*)	paréntesis para agrupar
f[*x*]	corchetes para funciones
{*a*, *b*, *c*}	llaves para listas
u[[*i*]]	corchetes dobles para indexar (Part[*u*, *i*])

Las cuatro formas de agrupar en *Mathematica*.

Cuando la expresión que usted digita es complicada, a veces es una buena idea ingresar un espacio en blanco para conjunto de agrupaciones. Esto hace que sea relativamente fácil ubicar pares de agrupaciones. *u*[[{*a*, *b*}]] permite, por ejemplo, fácilmente reconocer a *u*[[{*a*, *b*}]].

3.6. Secuencia de operaciones

Al realizar cálculos con *Mathematica*, usted usualmente lo hace mediante una secuencia de pasos. Si usted desea puede realizar cada paso en una línea separada. A veces, sin embargo, es conveniente ingresar varios pasos en una misma línea. Usted puede hacer esto simplemente separando cada una de las partes ingresadas con punto y coma.

exp_1; exp_2; exp_3	hace varias operaciones y da el resultado en la última línea
exp_1; exp_2;	hace las operaciones pero no imprime la salida

Formas de hacer secuencias de operaciones en *Mathematica*.

Esto realiza tres operaciones en una misma línea. El resultado corresponde a la última operación.

```
x = 3; y = 7; z = y - 2
5
```

Si usted finaliza su entrada con un punto y coma, esto es como si usted estuviera ingresando una secuencia de operaciones, con un punto y coma al final. Esto tiene el efecto de hacer que *Mathematica* calcule las operaciones especificadas, pero no muestre la salida.

exp;	realiza una operación, pero no muestra la salida

Inhibiendo la salida.

Añadiendo un punto y coma al final de la línea le *Mathematica* que no muestre la salida.

```
x = 47 + 5;
```

Usted puede usar % para poder visualizar la salida anterior.

```
%
52
```

4. Uso del Sistema Mathematica

4.1. La estructura de *Mathematica*

Kernel (núcleo) de *Mathematica*	la parte que realmente realiza los cálculos
Front end (parte visible) de *Mathematica*	la parte que se ocupa de interactuar con el usuario

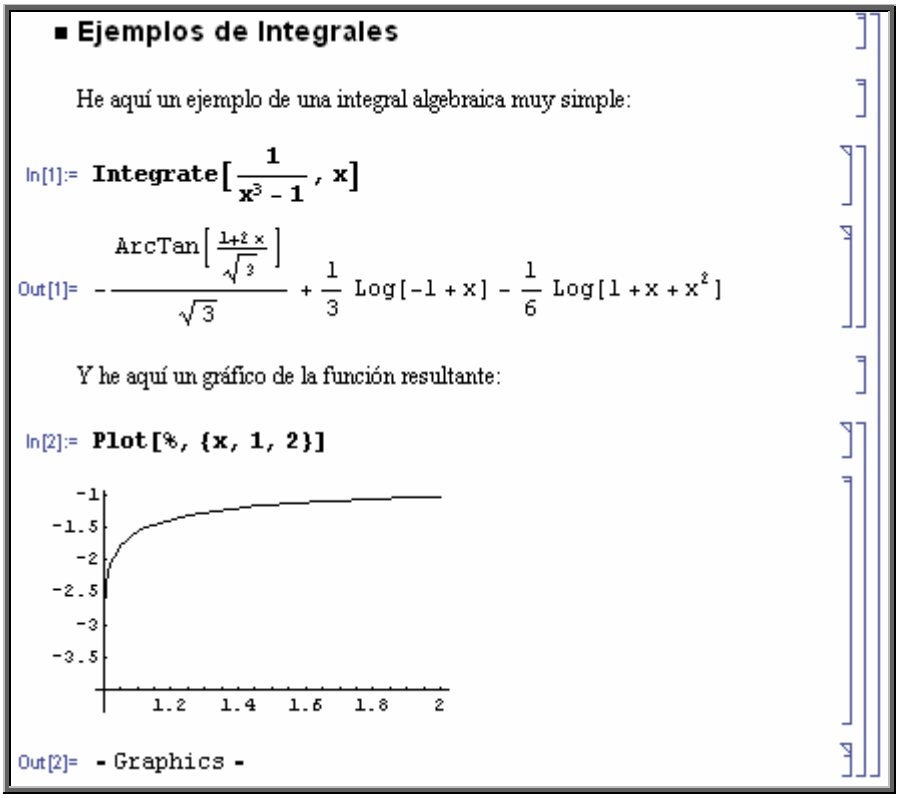
Partes básicas del Sistema *Mathematica*.

Mathematica es un sistema de software modular, en el cual el *kernel* que realmente realiza los cálculos está separado del *front end* que se ocupa de la interacción con el usuario. El tipo más común de front end de *Mathematica* está basado en documentos interactivos conocidos como *cuadernos*. Los cuadernos mezclan las entradas y salidas de *Mathematica* con texto, gráficos, paletas y otros materiales. Usted puede usar los cuadernos para hacer cálculos continuos o como medio para presentar o publicar resultados.

Interfase de cuaderno	documentos interactivos
Interfase basada en texto	texto desde el teclado
Interfase de *MathLink*	comunicación con otro programas

Tipos comunes de interfase con *Mathematica*.

Un cuaderno que mezcla texto, gráficos con entradas y salidas de *Mathematica*.



En algunos casos, puede que usted no necesite usar la interfase de cuaderno, y que desee en cambio interactuar directamente con el núcleo de *Mathematica*. Usted puede hacer esto usando la Interfase basada en texto, en la cual usted digita el texto en el teclado y éste va directamente al núcleo.

Un diálogo con *Mathematica* usando interfase basada en texto.

```
In[1]:= 2^100

Out[1]= 1267650600228229401496703205376

In[2]:= Integrate[1/(x^3 - 1), x]

                1 + 2 x
          ArcTan[-------]                                  2
                 Sqrt[3]     Log[-1 + x]   Log[1 + x + x ]
Out[2]= -(---------------) + ----------- - ---------------
              Sqrt[3]             3               6
```

Un aspecto importante de *Mathematica* es que no solamente puede interactuar con usuarios humanos sino que también puede hacerlo con otros programas. Esto se logra principalmente a través del *MathLink*, el cual es un protocolo estandarizado para la comunicación bidireccional entre programas externos y el núcleo de *Mathematica*.

Un fragmento de código C que se comunica vía *MathLink* con el kernel de *Mathematica*.

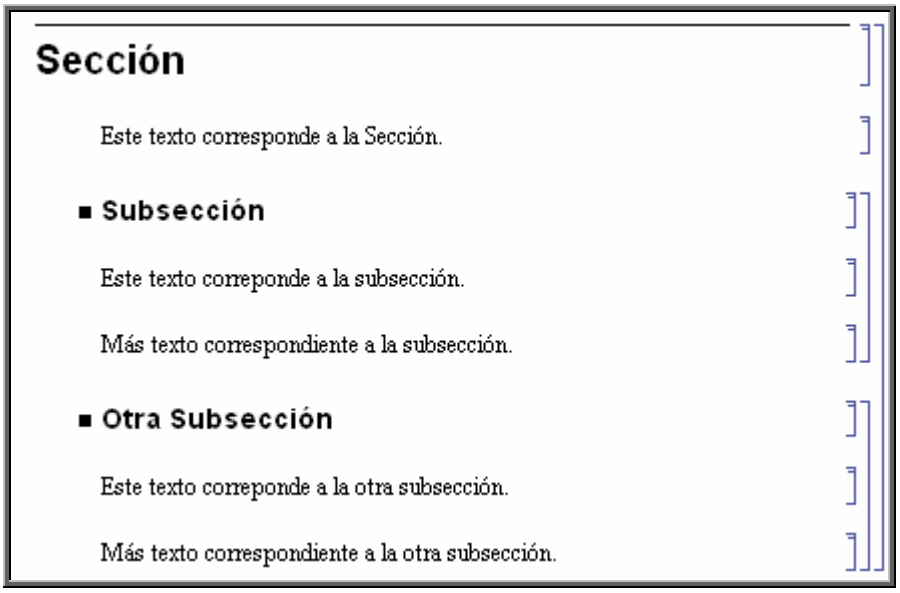
```
#include "mathlink.h"

int bitand(int x,int y);
void complements(intp_nt px,long nx);

int bitand(int x,int y)
{return(x& y);}

void complements(intp_nt px,long nx)
{long i;
  for(i=0;i<nx;i++) px[i]= ~px[i];
  MLPutIntegerList(stdlink,px,nx);}
```

4.2. Cuadernos como documentos

Los cuadernos de *Mathematica* le permiten crear documentos que pueden verse interactivamente en la pantalla o imprimirse en papel.

Particularmente, en los cuadernos extensos, es común tener los capítulos, secciones etc., representados cada uno en grupos de celdas. La extensión de estos grupos de celdas es indicada por el corchete que aparece a la derecha.

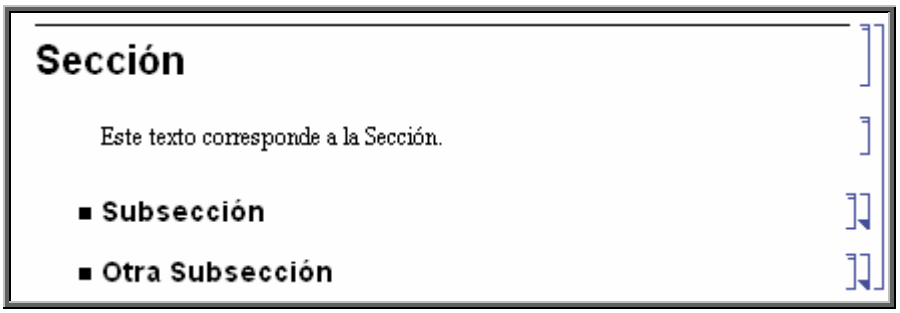
La agrupación de celdas en un cuaderno se indica con los corchetes anidados que aparecen en el lado derecho.



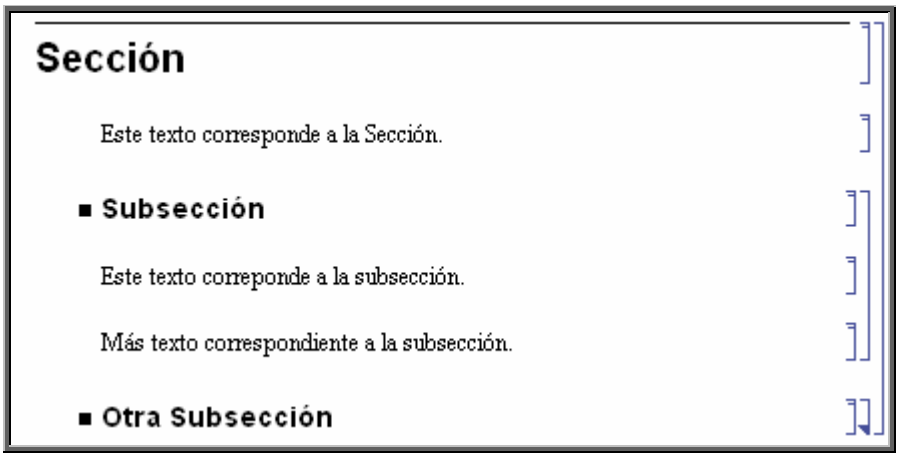
Un grupo de celdas puede estar *abierto* o *cerrado*. Cuando está abierto usted puede ver todas sus celdas explícitamente. Pero cuando está cerrado, usted sólo puede ver la celda que encabeza el grupo de celdas.

Los cuadernos extensos son a menudo distribuidos con muchos grupos de celdas cerradas, para que cuando usted vea por primera vez el cuaderno aprecie solamente una lista de su contenido. Usted puede abrir las partes en las que esté interesado haciendo doble clic sobre el corchete apropiado.

Haciendo doble clic sobre el corchete que abarca un grupo de celdas cerramos el grupo, dejando sólo la primera celda visible.



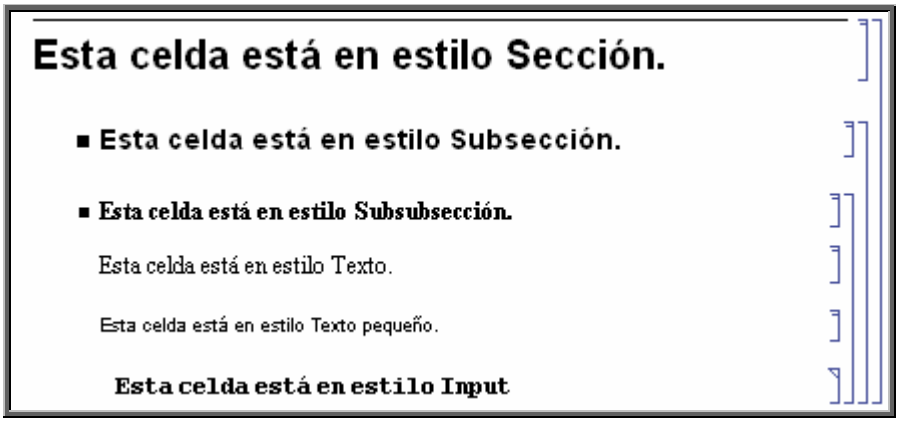
Cuando el grupo está cerrado, el corchete que le corresponde tiene una flecha en la parte inferior. Haciendo doble clic sobre esta flecha abrimos nuevamente el grupo.



A cada celda dentro de un cuaderno se le asigna un *estilo* en particular que indica su rol dentro del cuaderno. Así, por ejemplo, el material entendido como entrada para ser ejecutado por el núcleo de *Mathematica* está típicamente el estilo de `Input` (entrada), mientras que el que se entiende para ser leído como puramente de texto está típicamente en estilo `Text` (texto).

La interfase de *Mathematica* provee menús y métodos abreviados de teclas para crear celdas con diferentes estilos, y para cambiar estilos en las celdas existentes.

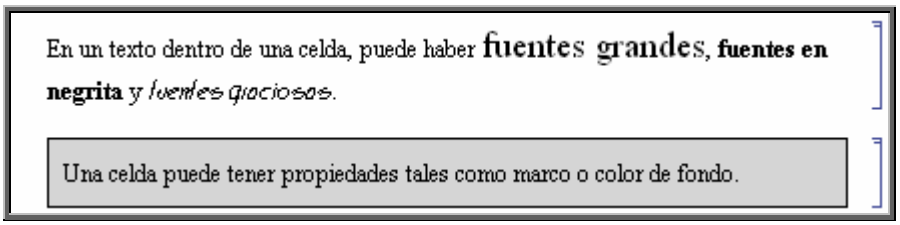
Esto muestra celdas en diferentes estilos. Los estilos no sólo definen el formato del contenido de las celdas, sino que también su ubicación y espaciado.



Poniendo una celda en un estilo en particular, usted especifica una colección entera de propiedades para celda, incluyendo por ejemplo el tamaño y la fuente del texto que se digitará.

La interfase de *Mathematica* permite modificar tales propiedades, bien sea para las celdas completas o para material específico dentro de las celdas.

Incluso dentro de una celda con un estilo en particular, la interfase de *Mathematica* permite modificar una amplia gama de propiedades en forma separada.



Vale mencionar que al hacer diversas clases de cosas con los cuadernos de *Mathematica*, usted está utilizando diversas partes del sistema de *Mathematica*. Las operaciones tales como apertura y cierre de grupos de celdas, realización de animaciones y reproducción de sonidos utilizan solamente una parte pequeña del front end de *Mathematica*, y estas operaciones tranquilamente se pueden realizar con un programa fácilmente disponible conocido como *MathReader*.

Para poder crear y corregir los cuadernos, usted necesita más que el front end de *Mathematica*. Y finalmente, para poder actualizar los cálculos dentro de un cuaderno de *Mathematica*, necesita un sistema completo de *Mathematica*, que contenga el front end y el núcleo.

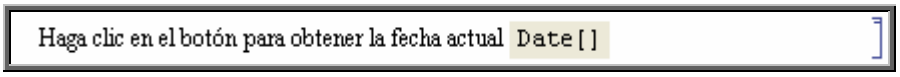
MathReader	lee cuadernos de *Mathematica*
front end de *Mathematica*	crea y edita cuadernos en *Mathematica*
núcleo de *Mathematica*	hace cálculos en cuadernos

Programas requeridos para diversas clases de operaciones en los cuadernos.

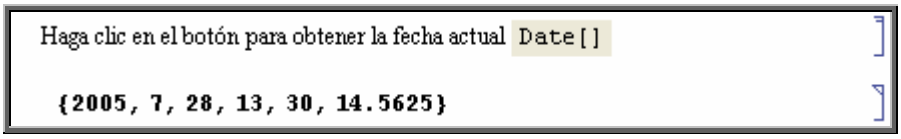
4.3. Elementos activos en cuadernos

Una de las características más potentes de los cuadernos de *Mathematica* es que sus acciones pueden ser programadas. Así, por ejemplo, usted puede establecer un botón en un cuaderno de *Mathematica* que permita realizar varias operaciones siempre que usted haga clic sobre él.

He aquí un cuaderno que contiene un botón.



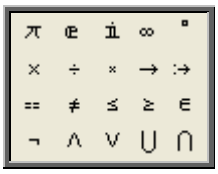
Haciendo clic en el botón conseguimos que se muestre la fecha actual.



Más adelante en este libro, discutiremos cómo usted puede crear botones y otros objetos similares en los cuadernos de *Mathematica*. Por ahora es suficiente decir que siempre que una celda se indique como activa, al hacer clic en los elementos activos dentro ella se producirá las acciones que han sido programadas para que estos elementos realicen.

Es común inicializar paletas que consisten en una serie de botones. A veces tales paletas aparecen como celdas dentro de un cuaderno. Aunque a menudo, se usa un cuaderno especial que aparece como ventana, que se puede poner convenientemente en algún lado de la pantalla de su computadora y utilizar conjuntamente con cualquier otro cuaderno.

Las paletas que consisten en una serie de botones a menudo son colocadas en cuadernos separados.

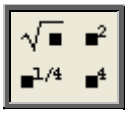


En los casos más simples, los botones en las paletas sirven como teclas adicionales a su teclado. Así, cuando usted presiona un botón, el carácter o el objeto mostrado en aquel botón es insertado en su cuaderno tal como si usted lo hubiera escrito a máquina.

Aquí está una paleta de letras griegas con botones que actúan como teclas adicionales a su teclado.

A menudo, sin embargo, un botón puede contener un placeholder indicado por ■. Esto significa que cuando usted presiona el botón, lo que se selecciona actualmente en su cuaderno será insertado en la posición del placeholder.

Estos botones contienen placeholders indicados por ■.



Este es un cuaderno con una expresión seleccionada.

Al presionar el botón superior izquierdo de la última paleta la expresión seleccionada queda envuelta en una raíz cuadrada.

$$a + \left(a + \sqrt{(a + (a + b^2)^2)^2}\right)^2$$

A veces los botones que contienen placeholders serán programados simplemente para insertar una cierta expresión en su cuaderno. Pero más a menudo, serán programados para evaluar el resultado, enviándolo como una entrada al núcleo de *Mathematica*.

Estos botones se inicializan para realizar operaciones.

```
Simplify[■]
FullSimplify[■]
```

He aquí un cuaderno con una expresión seleccionada.

$$\frac{2 + 2\,\mathrm{Cos}[2x]}{4} + \frac{1}{32}(12 - 16\,\mathrm{Cos}[2x] + 4\,\mathrm{Cos}[4x])$$

Al presionar el botón superior en la última paleta la expresión seleccionada es simplificada.

$$\frac{2 + 2\,\mathrm{Cos}[2x]}{4} + \mathrm{Sin}[x]^4$$

Hay algunas situaciones en las cuales es conveniente tener varios placeholders en un solo botón. Su actual selección se inserta en la posición del primer placeholder, indicada por ■. Los placeholder adicionales se indican por □, y usted puede trasladarse a través de los placeholders sucesivos usando la tecla <Tab>.

He aquí una paleta que contiene botones con varios placeholders.

$$\int ■\,d□ \quad \partial_□ ■$$

$$\int_□^□ ■\,d□ \quad \partial_{□,□} ■$$

He aquí una expresión en un cuaderno.

$$\frac{\text{Cos}[x]}{1+x}$$

Presionando el botón superior izquierdo en la última paleta insertamos la expresión en el lugar de ■.

$$\int \frac{\text{Cos}[x]}{1+x}\, d\blacksquare$$

Usted puede trasladarse a los otros placeholders usando la tecla <Tab>, y entonces los corrige para insertar lo que usted desea.

$$\int \frac{\text{Cos}[x]}{1+x}\, dx$$

4.4. Hipervínculos y texto activo

El front end de *Mathematica* proporciona una variedad de formas de buscar palabras o texto particulares en los cuadernos de *Mathematica*. Pero cuando los documentos son extensos o tenemos colecciones de documentos, conviene insertar hipervínculos que nos llevan inmediatamente a un punto específico en un cuaderno, tal como se hace en los websites.

Los hipervínculos se indican generalmente por palabras o frases subrayadas, y están a menudo en un color diferente. Al hacer clic en un hipervínculo le lleva inmediatamente dondequiera que el hipervínculo señale.

He aquí un texto. El texto puede contener un vínculo que lleva a cualquier otra parte.

Los hipervínculos en cuadernos trabajan bastante bien como los botones discutidos en la sección anterior. Y de nuevo, todos los aspectos de hipervínculos son programables.

De hecho, es posible crear texto activo en cuadernos que permita realizar casi cualquier tipo de acción.

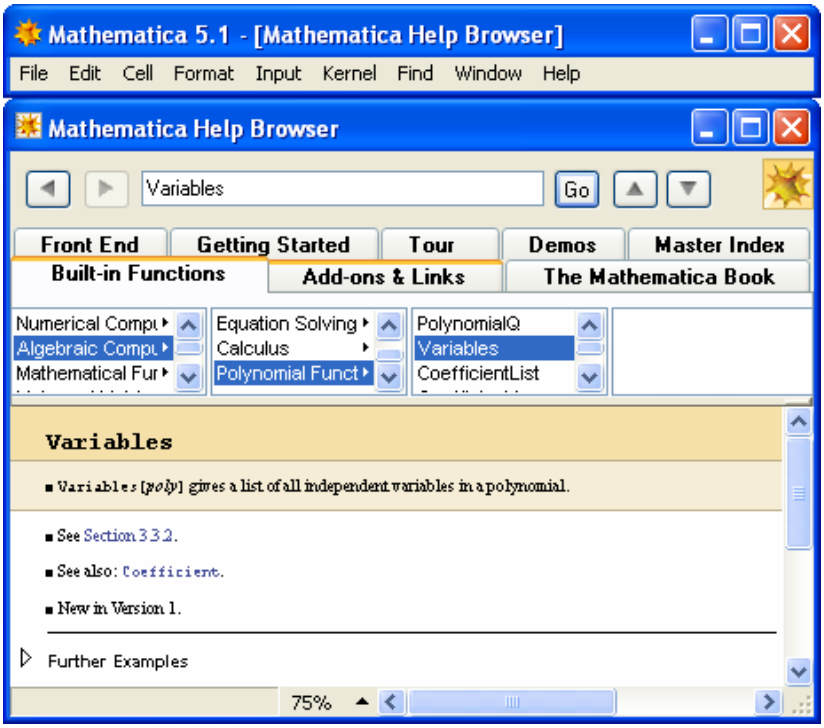
4.5. Acceso a la ayuda en una interfase de cuaderno

En la mayoría de las versiones de interfase de cuaderno de *Mathematica*, el menú Help (ayuda) le da acceso al Help Browser (navegador de ayuda), el cual sirve como un punto de entrada a la gran cantidad de documentación en línea para *Mathematica*.

Getting Started	un inicio rápido en el uso de *Mathematica*
Built-in Functions	información sobre todas las funciones incorporadas en *Mathematica*
The *Mathematica* Book	el libro completo en línea
Master Index	índice de todo el material documentado en línea

Típicos tipos de ayuda disponibles en la Interfase de notebook.

Un ejemplo de buscar la información básica sobre una función en el Help Browser de *Mathematica*.



4.6. Acceso a la ayuda en una interfase basada en texto

?*Name*	muestra información sobre *Name*
??*Name*	muestra información adicional sobre *Name*
?*Aaaa*∗	muestra información sobre todos los objetos cuyos nombres empiezan con *Aaaa*

Formas de recibir información directamente desde el kernel de *Mathematica*.

Esto da información de la función incorporada Log.

```
In[1]:= ?Log

      Log[z] gives the natural logarithm
         of z (logarithm to base e). Log[b, z]
         gives the logarithm to base b. More...
```

Usted puede preguntar por información acerca de cualquier objeto, si es incorporado en *Mathematica*, ha sido leído en un paquete de *Mathematica* o ha sido introducido por usted.

Cuando utiliza ? para conseguir la información, debe cerciorarse de que el signo de interrogación aparezca como el primer carácter en su línea de entrada. Debe hacer esto de modo que *Mathematica* sepa cuándo usted está solicitando información y no está dando una entrada ordinaria para ser evaluada.

Vale mencionar que junto con la información brindaba aparece un hipervínculo en color azul con la palabra More. Haciendo clic sobre este hipervínculo *Mathematica* nos lleva directamente a la información de la misma en el Help Browser.

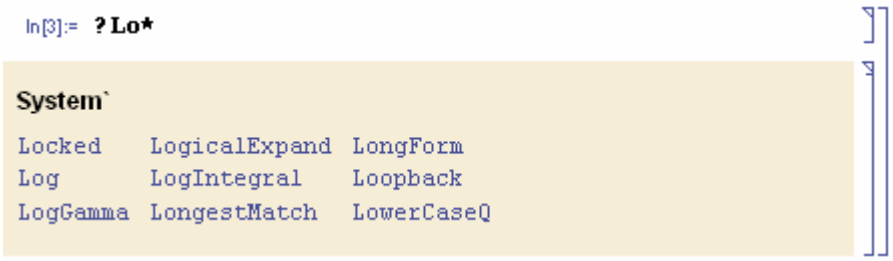
Puede conseguir información adicional usando ??.

```
In[2]:= ??Log

      Log[z] gives the natural logarithm
         of z (logarithm to base e). Log[b, z]
         gives the logarithm to base b. More...

      Attributes[Log] = {Listable, NumericFunction, Protected}
```

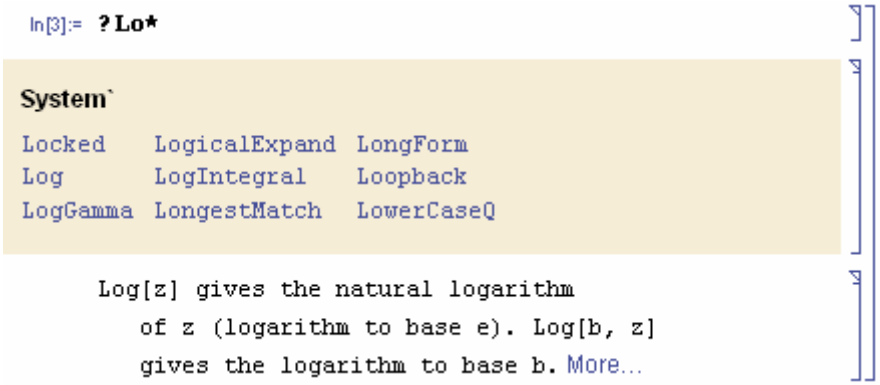
?Aaaa le dará la información de un objeto en particular cuyo nombre especifique. Usando el caracter *, sin embargo, usted puede obtener información sobre todos

los objetos con nombres similares. Así pues, por ejemplo `?Lo*` brinda información sobre todos los objetos cuyos nombres contengan las letras `Lo`, seguidas por cualquier cadena de caracteres.

Esto da información sobre todos los objetos de *Mathematica* cuyos nombres comiencen con `Lo`. Cuando hay más de un objeto, *Mathematica* enumera sus nombres como hipervínculos y haciendo clic sobre ellos es posible obtener información.



Haciendo clic en `Log` de la lista anterior se obtiene la respectiva información.



Usted puede poner `*` en cualquier lugar de la cadena que ingresa iniciada con `?`. Por ejemplo, `?*Expand` le daría todos los objetos cuyos nombres terminan en `Expand`. De manera similar `?P*d` le daría los objetos cuyos nombres comienzan con `P`, terminan con `d`, y tienen cualquier cadena de caracteres intermedios. (Note que la manera en que se utiliza `*` para especificar nombres en *Mathematica* es similar a la manera que se utiliza `*` en Unix y otros sistemas operativos para especificar nombres del archivos.)

Usted puede pedir la información sobre la mayor parte de las formas especiales de entrada que *Mathematica* utiliza. Esto pide la información acerca del operador `:=`.

```
In[4]:= ?:=

      lhs := rhs assigns rhs to be the delayed value
         of lhs. rhs is maintained in an unevaluated
         form. When lhs appears, it is replaced
         by rhs, evaluated afresh each time. More...
```

4.7. Paquetes en *Mathematica*

Una de las características más importantes de *Mathematica* es que es un sistema extensible. Hay una cierta cantidad de funciones incorporadas en *Mathematica*. Pero usando el lenguaje de programación de *Mathematica*, siempre es posible añadir más funciones.

Para muchos tipos de cálculos, lo incorporado en la versión estándar de *Mathematica* será suficiente. Sin embargo, si usted trabaja en particular en un área especializada, puede encontrarse en la necesidad de utilizar ciertas funciones no incorporadas en *Mathematica*.

En tales casos, usted podría encontrar un *package* (paquete) de *Mathematica* que contiene las funciones que usted necesita. Los paquetes de *Mathematica* son archivos escritos en el lenguaje de programación de *Mathematica*. Los mismos consisten en colecciones de definiciones hechas en *Mathematica* las cuales se aplican a áreas particulares.

<< *paquete*	lee un paquete de *Mathematica*

Leyendo paquetes de *Mathematica*.

Si usted quiere usar las funciones de un paquete en particular, primero debe leer el paquete en *Mathematica*. Los detalles de como hacer esto se discuten en la sección 12. Hay varias convenciones que gobiernan los nombres que usted usará para referirse a los paquetes.

Este comando lee un paquete en particular de *Mathematica*.

```
<<LinearAlgebra`Orthogonalization`
```

La función `GramSchmidt` está definida en el paquete.

```
GramSchmidt[{{1, 2}, {-7, 0}}]
```

$$\left\{\left\{\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}}\right\}, \left\{-\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}}\right\}\right\}$$

Hay varias sutilezas asociadas a estas cosas como los conflictos entre los nombres de las funciones en los diferentes paquetes. Un punto importante que no debe olvidar es que usted no debe referirse a una función que leerá desde un paquete antes de leerla realmente en el paquete. Si usted hace esto por equivocación, deberá ejecutar el comando `Remove["name"]` para librarse de la definición de la función que usted hizo antes de leer el paquete. Si usted no usa `Remove`, *Mathematica* usará "su" versión de la función, en lugar de la del paquete.

`Remove[`*name*`]`	remueve una función que ha sido introducida por error

Asegurándose que *Mathematica* use las definiciones correctas de los paquetes.

El hecho de que *Mathematica* pueda extenderse usando paquetes significa que las "partes de *Mathematica*" son ilimitadas. En lo que al uso concierne, no hay en realidad ninguna diferencia entre las funciones definidas en paquetes y las funciones incorporadas en *Mathematica*.

De hecho, un número determinado de las funciones descritas en este libro se ejecutan como paquetes de *Mathematica*. Sin embargo, en la mayoría de los sistemas de *Mathematica*, se han cargado los paquetes necesarios, de modo que las funciones que ellos definen estén siempre presentes.

Usted puede utilizar el Help Browser para obtener información sobre los paquetes estándares de *Mathematica*. Para ello seleccione la tarjeta Add-ons & Links del mismo.

4.8. Advertencias y mensajes

Mathematica sigue su trabajo silenciosamente, dando salida solamente cuando ha acabado de hacer de los cálculos que usted pidió.

Sin embargo, si *Mathematica* se percata de algo que usted esta haciendo y que definitivamente no entiende, imprimirá un mensaje para advertirle.

La función de la raíz cuadrada debe tener solamente un argumento. *Mathematica* imprime un mensaje para advertirle que usted ha dado dos argumento aquí.

```
Sqrt[4, 5]
```

```
Sqrt::argx :  Sqrt called with 2 arguments; 1 argument is expected.
```

```
Sqrt[4, 5]
```

Cada mensaje tiene un nombre. Usted puede apagar mensajes usando `Off`.

```
Off[Sqrt::argx]
```

El mensaje `Sqrt::argx` ahora se ha apagado, y no volverá a aparecer.

```
Off[Sqrt::argx]
```

Esto vuelve a encender `Sqrt::argx` otra vez.

```
On[Sqrt::argx]
```

`Off[`*Función*`::`*etiqueta*`]`	apaga (suprime) un mensaje
`On[`*Función*`::`*etiqueta*`]`	enciende un mensaje

Funciones para controla la salida de un mensaje.

4.9. Interrupción de cálculos

Probablemente habrá veces en que desee detener *Mathematica* en medio de un cálculo. Tal vez usted se da cuenta que pidió a *Mathematica* hacer un cálculo incorrecto. O quizás el cálculo tarda demasiado, y usted quiere saber que es lo que pasa.

La forma en que usted interrumpe un cálculo en *Mathematica* depende de qué clase de interfase está utilizando.

<Alt>+<,>	interfase de cuaderno
<Control>+<c>	interfase basada en texto

Combinaciones de teclas para interrumpir cálculos en *Mathematica*.

En algunos sistemas informáticos, puede tomar a *Mathematica* un cierto tiempo para responder a su interrupción. Cuando *Mathematica* responde, le dará un menú de cosas posibles para hacer.

`continue`	continuar el cálculo
`show`	mostrar que está haciendo *Mathematica*
`inspect`	examinar el estado actual de su cálculo
`abort`	abortar este cálculo en particular
`exit`	salir de *Mathematica* completamente

Algunas opciones disponibles cuando usted interrumpe un cálculo en *Mathematica*.

5. Cálculos algebraicos

5.1. Cálculo simbólico

Una de las características importantes de *Mathematica* es que puede hacer cálculos simbólicos y numéricos. Esto significa que puede manejar fórmulas algebraicas así como números.

He aquí un típico cálculo numérico.

```
4 + 36 - 1
39
```

Este es un cálculo simbólico.

```
7 x – 3 x + 6
6 + 4x
```

Cálculo numérico	`4 + 36 - 1` → `39`
Cálculo simbólico	`7x - 3x + 6` → `6 + 4x`

Cálculo simbólico y numérico.

Usted puede digitar cualquier expresión algebraica en *Mathematica*.

```
-1 + 2 x + x^3
-1 + 2x + x^3
```

Mathematica realiza automáticamente simplificaciones algebraicas básicas. Aquí combina a x^2 y a $-4x^2$ para dar $-3x^2$.

```
x^2 + x - 4x^2
x – 3x^2
```

Usted puede digitar cualquier expresión algebraica, usando a los operadores enumerados en la sección 2.1. Puede utilizar espacios para denotar la multiplicación. Tenga cuidado de no olvidarse del espacio en `x y`. Si usted digita `xy` sin espacio, *Mathematica* interpretará esto como solo símbolo, con el nombre `xy`, no como un producto de dos símbolos `x` y `y`.

Mathematica reordena y combina términos usando las reglas estándares del álgebra.

```
x y + 2 x^2 y + y^2 x^2 - 2 y x
```
$-xy + 2x^2y + x^2y^2$

He aquí otra expresión algebraica.

```
(x + 2 y + 1)(x - 2)^2
```
$(-2 + x)^2(1 + x + 2y)$

La función Expand amplía productos y potencias.

```
Expand[%]
```
$4 - 3x^2 + x^3 + 8y - 8xy + 2x^2y$

`Factor` hace lo inverso de `Expand`.

```
Factor[%]
```
$4 - 3x^2 + x^3 + 8y - 8xy + 2x^2y$

Cuando digita expresiones complicadas, es importante que ponga los paréntesis en los lugares correctos. Así, por ejemplo, debe dar la expresión x^{4y} en la forma `x^(4y)`. Si deja abiertos los paréntesis, se interpretará como $x^4 y$.

He aquí una fórmula más complicada, que requiere de varios paréntesis.

```
Sqrt[2]/9801 (4n)! (1103 + 26390 n) / (n!^4 + 1)
```
$$\frac{\sqrt{2}\ (1103 + 26390\,n)\ (4n)!}{9801\ (1 + (n!)^4)}$$

Cuando usted digita una expresión, *Mathematica* aplica automáticamente su gran repertorio de reglas para transformar las expresiones. Estas reglas incluyen las reglas estándares del álgebra, tales como $x - x = 0$, junto con reglas mucho más sofisticadas.

Mathematica utiliza reglas estándares del álgebra para sustituir $(\sqrt{1+x})^4$ por $(1+x)^2$.

```
Sqrt[1 + x]^4
(1 + x)^2
```

Mathematica no conoce ninguna regla para esta expresión, así que deja expresión en la forma original que usted le dio.

```
Log[1 + Cos[x]]
Log[1 + Cos[x]]
```

La noción de reglas de transformación es muy general. De hecho, usted puede pensar en *Mathematica* como un simple sistema que aplica una colección de reglas de transformación a diferentes clases de expresiones.

El principio general que *Mathematica* sigue es simple de explicar. Toma cualquier expresión que usted introduce, y obtiene resultados aplicando una sucesión de reglas de transformación, deteniéndose cuando no hay más reglas de transformación que aplicar.

- Tome cualquier expresión, y aplique reglas de transformación hasta que el resultado no cambie más.

El principio fundamental *Mathematica*.

5.2. Valores para símbolos

Cuando *Mathematica* transforma una expresión por ejemplo `x + x` en `2x`, está tratando la variable `x` en forma puramente simbólica o formal. En tales casos, `x` es un símbolo que puede representar cualquier expresión.

A menudo, sin embargo, usted necesita sustituir un símbolo como `x` por un "valor" determinado. Algunas veces este valor será un número; aunque frecuentemente será una expresión.

Para sustituir el símbolo `x`, que aparece en la expresión `1 + 2 x`, con un valor determinado; puede crear una regla de la transformación en *Mathematica*, y después aplicar esta regla a la expresión. Para sustituir `x` por el valor 3, usted crearía la regla de transformación `x->3`. Debe digitar `->` como un par de

caracteres, sin espacio entre ellos. Puede interpretar `x->3` como una regla en la cual "`x` será sustituido por `3`".

Para aplicar una regla de transformación a una expresión particular de *Mathematica*, usted digita `expr/.regla`. El "operador de reemplazo" `/.` se digita como un par de caracteres, sin espacio entre ellos.

Esto utiliza la regla de transformación `x->3` en la expresión `1 + 2 x`.

```
1 + 2x /. x -> 3
```
```
7
```

Usted puede sustituir `x` por cualquier expresión. Aquí cada ocurrencia de `x` es sustituida por `2 - y`.

```
1 + x + x^2 /. x -> 2 - y
```
```
3 + (2 - y)^2 - y
```

He aquí una regla de transformación. *Mathematica* la trata como cualquier otra expresión simbólica.

```
x -> 3 + y
```
```
x → 3 + y
```

Esto aplica la regla de transformación última a la expresión `x ^ 2 - 9`.

```
x^2 - 9 /. %
```
```
- 9 + (3 + y)^2
```

expr `/.` *x* `->` *valor*	reemplaza *x* por *valor* en la expresión *exp*
expr `/.` `{`*x* `->` *xval*`,` *y* `->` *yval*`}`	realiza varios reemplazos

Sustitución de símbolos por valores en expresiones.

Usted puede aplicar reglas juntas poniendo las reglas en una lista.

```
(x + y) (x - y)^2 /. {x -> 3, y -> 1 - a}
```
```
(4 - a) (2 + a)^2
```

El operador de reemplazo `/.` le permite aplicar reglas de la transformación a una expresión particular. A veces, sin embargo, usted querrá definir reglas de transformación que se apliquen siempre. Por ejemplo, puede ser que desee sustituir `x` por `3` siempre que aparezca `x`.

Según lo discutido en la sección 3.2, usted puede hacer esto asignando el valor 3 a x usando x = 3. Una vez que haya hecho la asignación x = 3, x siempre será sustituido siempre por 3, cuando aparezca.

Esto asigna el valor de 3 a x.

```
x = 3
3
```

Ahora x será sustituido automáticamente por 3 dondequiera que aparezca.

```
x^2 - 1
8
```

Esto asigna la expresión 1 + a a x.

```
x = 1 + a
1 + a
```

Ahora x es reemplazado por 1 + a.

```
x^2 - 1
-1 + (1 + a)^2
```

Usted puede definir como valor de un símbolo a cualquier expresión, no solamente a un número. Recuerde que una vez que haya dado tal definición, ésta continuará siendo utilizada siempre que aparezca el símbolo, hasta que la cambie o quite explícitamente. Para la mayoría de usuarios, el olvidarse quitar valores que han asignado a los símbolos es la causa más común de errores al usar *Mathematica*.

x = *valor*	define un valor para *x* que será utilizado siempre
x = .	remueve cualquier valor definido para *x*

Asignando valores a símbolos.

El símbolo x todavía tiene el valor que le asignó arriba.

```
x + 5 - 2x
6 + a - 2(1 + a)
```

Esto quita el valor que asignó a x.

```
x =.
```

Ahora x no tiene ningún valor definido, así que puede ser utilizado como variable puramente simbólica.

```
x + 5 - 2x
5 - x
```

Los lenguajes de programación tradicionales que no soportan el cálculo simbólico permiten que las variables sean utilizadas solamente como nombres para objetos, típicamente números, que se han asignado como valores para ellos. En *Mathematica*, sin embargo, x se puede también tratar como variable puramente formal, a la cual se le puede aplicar varias reglas de transformación. Por supuesto, si usted da explícitamente una definición, por ejemplo `x = 3`, entonces x será sustituida siempre por 3, y no sirve más como variable formal.

Debe recordar que las definiciones explícitas por ejemplo `x = 3` tienen un efecto global. Por otra parte, un reemplazo tal como *exp* `/ . x - > 3` afecta solamente a la expresión específica *expr*.

Puede mezclar siempre reemplazos con asignaciones. Con asignaciones, puede dar nombres a las expresiones en las cuales desea hacer reemplazos, o a las reglas que usted desea utilizar para hacer los reemplazos.

Esto asigna un valor al símbolo t.

```
t = 1 + x^2
1 + x^2
```

Esto encuentra el valor de t, y después sustituye x por 2 en él.

```
t /. x -> 2
5
```

Esto encuentra el valor de t para un valor diferente de x.

```
t /. x -> 5a
1 + 25 a^2
```

Esto encuentra el valor de t cuando x es sustituido por Pi, y luego evalúa el resultado numéricamente.

```
t /. x -> Pi //N
10.8696
```

5.3. Transformación de expresiones algebraicas

A menudo hay muchas formas diferentes de escribir la misma expresión algebraica. Como un ejemplo, la expresión $(1+x)^2$ puede ser escrita como $1+2x+x^2$. *Mathematica* proporciona una gran colección de funciones para hacer conversiones entre las diferentes formas de expresiones algebraicas.

`Expand[`*expr*`]`	desarrolla productos y potencias, escribiendo el resultado como suma de términos
`Factor[`*expr*`]`	escribe *expr* como un producto de factores mínimos

Dos funciones comunes para transformar expresiones algebraicas.

`Expand` da la "forma expandida", con los productos y las potencias desarrolladas.

```
Expand[ (1 + x)^2 ]
```
$1 + 2\,x + x^2$

`Factor` recupera la forma original.

```
Factor[ % ]
```
$(1 + x)^2$

Es fácil generar expresiones complicadas con `Expand`.

```
Expand[ (1 + x + 3 y)^4 ]
```
$$1 + 4\,x + 6\,x^2 + 4\,x^3 + x^4 + 12\,y + 36\,x\,y + 36\,x^2\,y + 12\,x^3\,y + 54\,y^2 + 108\,x\,y^2 + 54\,x^2\,y^2 + 108\,y^3 + 108\,x\,y^3 + 81\,y^4$$

`Factor` a menudo le da expresiones más simples.

```
Factor[ % ]
```
$(1 + x + 3\,y)^4$

Hay algunos casos, no obstante, donde `Factor` puede darle expresiones más complicadas.

```
Factor[ x^10 - 1 ]
```
$$(-1 + x)\,(1 + x)\,(1 - x + x^2 - x^3 + x^4)\,(1 + x + x^2 + x^3 + x^4)$$

En este caso, `Expand` da la forma “más simple”.

```
Expand[ % ]
-1 + x^10
```

5.4. Simplificación de expresiones algebraicas

En muchas situaciones usted desea escribir una expresión algebraica en la forma más simple posible. Aunque sea difícil saber exactamente lo que se entiende por la “forma más simple”, un procedimiento práctico que vale la pena seguir es analizar varias formas diferentes de una expresión, y elegir la de menor número de partes.

`Simplify[`*expr*`]`	intenta encontrar la forma más simple de *expr* aplicando varias transformaciones algebraicas estándares
`FullSimplify[`*expr*`]`	intente encontrar la forma más simple aplicando una amplia gama de transformaciones

Simplificación de expresiones algebraicas.

`Simplify` escribe x^2+2x+1 en forma factorizada.

```
Simplify[x^2 + 2x + 1]
(1 + x)^2
```

`Simplify` deja a $x^{10}-1$ en forma expandida, puesto que para esta expresión, la forma descompuesta en factores es más grande.

```
Simplify[x^10 - 1]
-1 + x^10
```

Usted puede utilizar a menudo `Simplify` para “mejorar” expresiones complicadas que obtiene como resultado de cálculos.

He aquí la integral de $1/(x^4-1)$.

```
Integrate[1/(x^4 - 1), x]
```

$$-\frac{\text{ArcTan}[x]}{2}+\frac{1}{4}\text{Log}[-1+x]-\frac{1}{4}\text{Log}[1+x]$$

Al derivar el último resultado deberíamos volver a la expresión original. En este caso, como es común, se obtiene una versión más complicada de la expresión original.

```
D[%, x]
```

$$\frac{1}{4(-1+x)} - \frac{1}{4(1+x)} - \frac{1}{2(1+x^2)}$$

`Simplify` permite volver a la forma original de la expresión

```
Simplify[%]
```

$$\frac{1}{-1+x^4}$$

`Simplify` se usa para aplicar varias transformaciones algebraicas estándares a las expresiones que usted ingresa. A veces, sin embargo, puede usar transformaciones más sofisticadas para poder encontrar la forma más simple de una expresión.

`FullSimplify` aplica una gama mucho más amplia de transformaciones, incluyendo no solamente funciones algebraicas, sino también muchas otras clases de funciones.

`Simplify` no altera esta expresión.

```
Simplify[Gamma[x] Gamma[1 - x]]
Gamma[1 – x] Gamma[x]
```

`FullSimplify`, sin embargo, consigue simplificarla.

```
FullSimplify[Gamma[x] Gamma[1 - x]]
πCsc[πx]
```

Para expresiones muy pequeñas, `FullSimplify` a menudo permitirá obtener algunas simplificaciones notables. Pero para expresiones grandes, puede llegar a ser considerablemente lento. La razón de esto es que para hacer su trabajo, FullSimplify tiene que tratar de combinar cada parte de una expresión con otra, y para expresiones grandes el número de casos que debe considerar puede ser astronómicamente grande.

`Simplify` también tiene una tarea difícil de hacer, pero evita algunas de las transformaciones que demandan más tiempo y que son utilizadas por `FullSimplify`. Para los cálculos algebraicos simples, por lo tanto, resulta conveniente aplicar `Simplify` a sus resultados.

En cálculos más complicados, sin embargo, incluso `Simplify` puede ser que necesite probar con un número muy grande de diversas formas, y por lo tanto tome un largo tiempo. En tales casos, necesita hacer una simplificación controlada, y utilizar su conocimiento de manera que usted guíe el proceso.

5.5. Puesta de expresiones en diferentes formas

Las expresiones algebraicas complicadas se pueden escribir generalmente en varias maneras. *Mathematica* proporciona una variedad de funciones para convertir expresiones de una forma a otra. Las más comunes de estas funciones son `Expand`, `Factor` y `Simplify`. Sin embargo, cuando usted tiene expresiones racionales que contienen cocientes, puede ser que necesite utilizar otras funciones.

`Expand[`*expr*`]`	desarrolla productos y potencias, escribiendo
`ExpandAll[`*expr*`]`	aplica `Expand` por todas partes
`Factor[`*expr*`]`	reduce a producto de factores
`Together[`*expr*`]`	pone todos los términos sobre un común denominador
`Apart[`*expr*`]`	separa en términos con denominadores simples
`Cancel[`*expr*`]`	cancela factores comunes entre numeradores y denominadores
`Simplify[`*expr*`]`	aplica un conjunto de transformaciones algebraicas y da la forma más pequeña de *expr* encontrada

Funciones para transformar expresiones algebraicas.

He aquí una expresión racional que puede ser escrita en varias formas diferentes.

```
e = (x - 1)^2 (2 + x) / ((1 + x) (x - 3)^2)
```

$$\frac{(-1+x)^2\,(2+x)}{(-3+x)^2\,(1+x)}$$

`Expand` expande el numerador, pero deja el denominador en forma factorizada.

```
Expand[e]
```

$$\frac{2}{(-3+x)^2\,(1+x)} - \frac{3x}{(-3+x)^2\,(1+x)} + \frac{x^3}{(-3+x)^2\,(1+x)}$$

`ExpandAll` expande todo, incluyendo el denominador.

`ExpandAll[e]`

$$\frac{2}{9+3x-5x^2+x^3}-\frac{3x}{9+3x-5x^2+x^3}+\frac{x^3}{9+3x-5x^2+x^3}$$

`Apart` separa la expresión en términos con denominadores simples.

`Apart[%]`

$$1+\frac{5}{(-3+x)^2}+\frac{19}{4(-3+x)}+\frac{1}{4(1+x)}$$

`Factor` factoriza todo, en este caso reproduce la forma original.

`Factor[%]`

$$\frac{(-1+x)^2(2+x)}{(-3+x)^2(1+x)}$$

Según `Simplify`, esta es la forma más simple de escribir la expresión original.

`Simplify[e]`

$$\frac{(-1+x)^2(2+x)}{(-3+x)^2(1+x)}$$

Conseguir expresiones en la forma que uno desea se convierte en todo un arte. En la mayoría de los casos, es mejor simplemente experimentar, intentando diversas transformaciones hasta que se consiga lo que se desea.

Cuando tenga una expresión con una sola variable, puede elegir escribirla como una suma de términos, un producto, etcétera. Si tiene una expresión con varias variables, hay una colección más amplia de posibles formas. Usted puede, por ejemplo, elegir agrupar términos en una expresión de modo que una u otra de las variables sea “dominante”.

`Collect[`*expr*`, `*x*`]`	agrupa juntas todos las potencias de *x*
`FactorTerms[`*expr*`, `*x*`]`	extrae los factores que no dependen de *x*

Reordenamiento de expresiones en varias variables.

He aquí una expresión algebraica en dos variables.

`v = Expand[(3 + 2 x)^2 (x + 2 y)^2]`

$$9x^2+12x^3+4x^4+36xy+48x^2y+16x^3y+36y^2+48xy^2+16x^2y^2$$

Esto agrupa los términos de v afectados por la misma potencia de x.

```
Collect[v, x]
```

$$4x^4 + 36y^2 + x^3(12 + 16y) + x^2(9 + 48y + 16y^2) + x(36y + 48y^2)$$

Esto agrupa juntas las potencias de y.

```
Collect[v, y]
```

$$9x^2 + 12x^3 + 4x^4 + (36x + 48x^2 + 16x^3)y + (36 + 48x + 16x^2)y^2$$

Uno de estos factores no depende de y.

```
FactorTerms[v, y]
```

$$(9 + 12x + 4x^2)(x^2 + 4xy + 4y^2)$$

Como hemos visto, cuando usted se limita a expresiones polinómicas racionales, hay muchas formas de escribir cualquier expresión particular. Si considera expresiones más complicadas, que incluyan, por ejemplo, funciones matemáticas trascendentes, la variedad de formas posibles llega a ser mayor. Por consiguiente, es imposible tener una función incorporada específica en *Mathematica* para producir cada forma posible. Más bien *Mathematica* le permite construir sistemas arbitrarios de reglas de transformación para hacer diversas conversiones. Sin embargo hay algunas funciones incorporadas adicionales de *Mathematica* para transformar expresiones.

`TrigExpand[`*expr*`]`	expande expresiones trigonométricas en una suma de términos
`TrigFactor[`*expr*`]`	factoriza expresiones trigonométricas en productos de términos
`TrigReduce[`*expr*`]`	reduce expresiones trigonométricas usando ángulos múltiples
`TrigToExp[`*expr*`]`	convierte funciones trigonométricas a exponenciales
`ExpToTrig [`*expr*`]`	convierte funciones exponenciales a trigonométricas
`FunctionExpand[`*expr*`]`	expande funciones especiales y otras
`ComplexExpand[`*expr*`]`	realiza expansiones asumiendo que todas las variables son reales
`PowerExpand[`*expr*`]`	transforma $(xy)^p$ en $x^p y^p$, etc.

Algunas funciones más para transformar expresiones.

Esto expande la expresión trigonométrica, escribiéndola de modo que todas las funciones tengan argumento x.

```
TrigExpand[Tan[x] Cos[2x]]
```

$$\frac{3}{2}\,\text{Cos}[x]\,\text{Sin}[x] - \frac{\text{Tan}[x]}{2} - \frac{1}{2}\,\text{Sin}[x]^2\,\text{Tan}[x]$$

Esto utiliza identidades trigonométricas para factorizar la expresión.

```
TrigFactor[%]
```

```
(Cos[x] - Sin[x]) (Cos[x] + Sin[x]) Tan[x]
```

Esto reduce la expresión usando ángulos múltiples.

```
TrigReduce[%]
```

$$-\frac{1}{2}\,\text{Sec}[x]\,(\text{Sin}[x] - \text{Sin}[3x])$$

Esto expande el seno asumiendo que x e y son reales.

```
ComplexExpand[ Sin[x + I y] ]
```

```
Cosh[y] Sin[x] + i Cos[x] Sinh[y]
```

Esto hace la extensión asumiendo que x e y son complejos.

```
ComplexExpand[ Sin[x + I y], {x, y} ]
```

```
-Cosh[Im[x] + Re[y]] Sin[Im[y] - Re[x]] +
  i Cos[Im[y] - Re[x]] Sinh[Im[x] + Re[y]]
```

Las transformaciones a expresiones hechas por funciones como `Expand` y `Factor` siempre son correctas, independientemente del valor que puedan tener las variables simbólicas en las expresiones. A veces, sin embargo, es útil realizar transformaciones que sólo son correctas para algunos posibles valores de las variables simbólicas. Una transformación como esta la realiza `PowerExpand`.

Mathematica no expande automáticamente potencias no enteras de productos.

```
Sqrt[x y]
```

$$\sqrt{x\,y}$$

`PowerExpand` hace la expansión.

```
PowerExpand[%]
```

$$\sqrt{x}\,\sqrt{y}$$

He aquí otro ejemplo.

```
Sqrt[a^2]//PowerExpand
```

a

5.6. Simplificación con asunciones

`Simplify[`*expr*`,` *asun*`]` simplifica *exp* son asunciones

Simplificación con asunciones.

Mathematica no simplifica esto automáticamente, puesto que es solamente verdad para algunos valores de `x`.

```
Simplify[Sqrt[x^2]]
```

$\sqrt{x^2}$

$\sqrt{x^2}$ es igual a x para $x \geq 0$, pero no en otro caso.

```
{Sqrt[4^2], Sqrt[(-4)^2]}
```

{4, 4}

Esto le dice a `Simplify` que haga la asunción `x>0`, de modo que la simplificación pueda proceder.

```
Simplify[Sqrt[x^2], x > 0]
```

x

Ninguna simplificación automática se puede hacer en esta expresión.

```
2 a + 2 Sqrt[a - Sqrt[-b]] Sqrt[a + Sqrt[-b]]
```

$2\,a + 2\sqrt{a - \sqrt{-b}}\,\sqrt{a + \sqrt{-b}}$

Si *a* y *b* son asumidos positivos, la expresión puede sin embargo ser simplificada.

```
Simplify[%, a > 0 && b > 0]
```

$2\left(a + \sqrt{a^2 + b}\right)$

He aquí un ejemplo simple que involucra funciones trigonométricas.

```
Simplify[ArcSin[Sin[x]], -Pi/2 < x < Pi/2]
```

x

`Element[`x`,`dom`]`	indica que x es un elemento del dominio dom
`Element[{`x_1`,`x_2`, ...},`dom`]`	indica que todos los x_i son elementos del dominio dom
`Reals`	números reales
`Integers`	enteros
`Primes`	números primos

Algunos dominios usados en asunciones.

Esto simplifica $\sqrt{x^2}$ asumiendo que x es un número real.

```
Simplify[Sqrt[x^2],
Element[x, Reals]]
Abs[x]
```

Esto simplifica el seno asumiendo que n es un entero.

```
Simplify[Sin[x + 2 n Pi],
Element[n, Integers]]
Sin[x]
```

Con las asunciones dadas, puede ser utilizado el Pequeño Teorema de Fermat.

```
Simplify[Mod[a^p, p], Element[a, Integers] &&
Element[p, Primes]]
Mod[a,p]
```

Esto utiliza el hecho de que $\sin(x)$, pero no $\arcsin(x)$, es real cuando x es real.

```
Simplify[Re[{Sin[x], ArcSin[x]}],
Element[x, Reals]]
{Sin[x],Re[ArcSin[x]]}
```

5.7. Seleccionar partes de Expresiones Algebraicas

`Coefficient[`*expr*`,`*form*`]`	coeficiente de *form* en *expr*
`Exponent[`*expr*`,`*form*`]`	máxima potencia de *form* en *expr*
`Part[`*expr*`,`n`]` or *exp*`[[`n`]]`	n-ésimo término de *expr*

Funciones para seleccionar partes de polinomios.

He aquí una expresión algebraica.

```
e = Expand[(1 + 3x + 4y^2)^2]
```

$$1 + 6x + 9x^2 + 8y^2 + 24xy^2 + 16y^4$$

Esto da el coeficiente de `x` en `e`.

```
Coefficient[e, x]
```

$$6 + 24y^2$$

`Exponent[`*expr*`,`*y*`]` da la mayor potencia de de *y* que aparece en *expr*.

```
Exponent[e, y]
```

```
4
```

Esto da término en `e`.

```
Part[e, 4]
```

$$8y^2$$

Usted puede notar que la función `Part[`*expr*, *n*`]` usada para seleccionar el *n*-ésimo término en una suma es igual que la función descrita en la sección 3.4 para seleccionar elementos en listas. Ésta no es ninguna coincidencia. En efecto, según lo discutido en la sección 3.5, cada expresión de *Mathematica* se puede manipular estructuralmente como una lista. Sin embargo, según lo discutido en la sección 3.5, debe tener cuidado, porque *Mathematica* muestra a menudo expresiones algebraicas en una forma diferente a como las trata internamente.

`Coefficient` trabaja incluso con polinomios que no están explícitamente expandidos.

```
Coefficient[(1 + 3x + 4y^2)^2, x]
```

$$6 + 24y^2$$

`Numerator[`*expr*`]`	numerador de *expr*
`Denominator[`*expr*`]`	denominador de *expr*

Funciones para seleccionar partes de expresiones racionales.

He aquí una expresión racional.

```
r = (1 + x)/(2 (2 - y))
```

$$\frac{1 + x}{2\,(2 - y)}$$

`Denominator` selecciona el denominador.

```
Denominator[%]
```
$2\,(2-y)$

`Denominator` da 1 para las expresiones que no son cocientes.

```
Denominator[1/x + 2/y]
1
```

5.8. Controlar la presentación de expresiones grandes

Cuando usted hace cálculos simbólicos, es muy fácil encontrarse con expresiones extremadamente complicadas. A menudo, incluso no deseará ver el resultado completo de un cálculo.

Si termina su entrada con un punto y coma, *Mathematica* hará el cálculo que pidió, pero no mostrará el resultado. Puede sin embargo utilizar % o `Out[`*n*`]` para referirse al resultado.

Aunque no desee ver el resultado completo de un cálculo, a menudo necesita ver su forma básica. Puede utilizar `Short` para presentar la salida de una expresión, omitiendo algunos de los términos.

Terminar su entrada con `;` impide a *Mathematica* mostrar el resultado complicado de un cálculo.

```
Expand[(x + 5 y + 10)^8] ;
```

Usted puede referirse al resultado como %. `//Short` muestra el resultado en una línea. <<*n*>> indica los *n* términos que no se han mostrado.

```
% //Short
```
$100000000 + 80000000\,x + \text{«42»} + 390625\,y^8$

Esto muestra una versión de tres líneas de la expresión. Más partes son visibles ahora.

```
Short[%, 3]
```
$$100000000 + 80000000\,x + 28000000\,x^2 + \\ 5600000\,x^3 + 700000\,x^4 + \text{«35»} + 8750000\,x\,y^6 + \\ 437500\,x^2\,y^6 + 6250000\,y^7 + 625000\,x\,y^7 + 390625\,y^8$$

Esto da el número total de términos en la suma.

```
Length[%]
45
```

comando`;`	ejecuta *comando* pero no imprime el resultado
expr`//Short`	muestra *expr* en una sola línea
`Short[`*expr*`,`*n*`]`	muestra *expr* en *n* líneas

Algunas formas de acortar su salida.

5.9. Las limitaciones de *Mathematica*

En sólo un comando de *Mathematica*, usted puede especificar fácilmente un cálculo que sea demasiado complicado de hacer para cualquier ordenador. Por ejemplo, usted podría pedir `Expand[(1+x)^(10^100)]`. El resultado de este cálculo tendría más términos que el número total de partículas en el universo.

Usted no tendría ningún problema para resolver `Expand[(1+x)^100]` en cualquier ordenador que pueda utilizar *Mathematica*. Pero cuando usted aumenta el exponente de `(1+x)`, los resultados que consigue tarde o temprano se harán demasiado grandes de almacenar para la memoria de su ordenador. Exactamente en qué punto sucede esto depende no solamente de la cantidad total de memoria que su computadora tiene, sino a menudo también de detalles tales como qué otros trabajos esta realizando su computadora cuando usted intenta hacer su cálculo.

Si su ordenador realmente se queda sin memoria en medio de un cálculo, la mayor parte de versiones de *Mathematica* no tienen ninguna otra opción, más que pararse inmediatamente. Por consiguiente, es importante planificar sus cálculos de modo que ellos nunca necesiten más memoria de la que su ordenador tiene.

Incluso si el resultado de un cálculo algebraico es absolutamente simple, las expresiones intermedias que usted genera en el curso del cálculo pueden ser muy complicadas. Esto significa que incluso si el resultado final es pequeño, las partes intermedias de un cálculo pueden ser demasiado grandes de manejar para su ordenador. Si esto sucede, puede particionar su cálculo, y resolver exitosamente cada parte del mismo. Usted debe saber que el esquema interno que *Mathematica* usa para direccionar la memoria es tal que cuando la parte de un cálculo es terminada, la memoria que se usaba para almacenar las expresiones intermedias que surgen inmediatamente es hecha disponible para nuevas expresiones.

El espacio de memoria es el factor restrictivo más común en los cálculos con *Mathematica*. El tiempo también, sin embargo, puede ser un factor restrictivo. Usted por lo general esta preparado para esperar un segundo, o aún un minuto, para obtener el resultado de un cálculo. Pero no lo esta para esperar una hora o un día, y casi nunca será capaz de esperar un año.

El código interno de *Mathematica* usa algoritmos sumamente eficientes y optimizados. Pero hay algunas tareas para las cuales los mejores algoritmos conocidos siempre tarde o temprano toman una gran cantidad de tiempo. Una cuestión típica es que el tiempo requerido por el algoritmo puede aumentar casi exponencialmente con el tamaño de la entrada. Un caso clásico es la factorización de números enteros–donde los mejores algoritmos conocidos requieren tiempos que crecen casi de manera exponencial con el número de dígitos. En la práctica, usted encontrará que `FactorInteger[`*k*`]` dará un resultado casi inmediato cuándo *k* tiene menos de aproximadamente 40 dígitos. Pero si *k* tiene 60 dígitos, `FactorInteger[`*k*`]` puede comenzar a tomar un tiempo inmanejablemente largo.

En algunos casos, hay mejora progresiva de los algoritmos que se conocen, de modo que las versiones sucesivas de *Mathematica* puedan realizar cálculos particulares cada vez más rápidos. Pero las ideas de la teoría de cómputo sugieren que muchos cómputos siempre requerirán una cantidad irreducible de trabajo computacional–de modo que ningún algoritmo más rápido sea encontrado alguna vez para ellos.

- Hacer aritmética con los números que contienen algunos cientos millones de dígitos.
- Generar un millón de dígitos de números como π y e.
- Expandir un polinomio que da un millón de términos.
- Factorizar un polinomio en cuatro variables con unos cien mil términos.
- Reducir un sistema de desigualdades cuadráticas para unos miles de componentes independientes.
- Encontrar las raíces enteras de un polinomio con grado un millón.
- Aplicación de una regla recurrente un millón de veces.
- Cálculo de todos los primos hasta diez millones.
- Encontrar la inversa numérica de una matriz de 1000×1000.
- Solución de un sistema lineal de un millón de variables con cien mil coeficientes no ceros.
- Encontrar el determinante de una matriz entera de 250×250.
- Encontrar el determinante de una matriz simbólica de 20×20.

- Encontrar las raíces numéricas de un polinomio de grado 200.
- Solución de un problema de programación lineal con cien mil variables.
- Encontrar la trasformada de Fourier de una lista con cien millones de elementos.
- Representación de millón de gráficos primitivos.
- Clasificar una lista de diez millones de elementos.
- Buscar una cadena que tiene diez millones de caracteres.
- Importación de unos diez megabytes de datos numéricos.
- Formatear unas cien de páginas con salida `TraditionalForm`.

Algunas operaciones que típicamente toman algunos segundos en una PC 2004.

5.10. Uso de símbolos para etiquetar objetos

Hay muchas maneras de utilizar símbolos en *Mathematica*. Hasta ahora, nos hemos concentrado en usar símbolos para almacenar valores y para representar variables matemáticas. Esta sección describe otra manera de utilizar símbolos en *Mathematica*.

La idea es utilizar símbolos como "etiquetas" para diversos tipos de objetos.

Trabajar con unidades físicas da un ejemplo simple. Cuando usted especifica la longitud de un objeto, desea dar no solamente un número, sino también las unidades en las cuales se mide la longitud. En la notación estándar, puede ser que escriba una longitud como 12 metros.

Usted puede imitar esta notación casi directamente en *Mathematica*. Puede por ejemplo utilizar simplemente un símbolo `metros` para indicar las unidades de nuestra medida.

El símbolo `metros` aquí actúa como etiqueta, la cual indica las unidades usadas.

```
12 metros
12 metros
```

Usted puede añadir longitudes como en este caso.

```
% + 5.3 metros
17.3 metros
```

Esto da una velocidad.

```
% / (25 segundos)
```

$$\frac{0.692\ \text{metros}}{\text{segundos}}$$

Esto convierte a una velocidad en pies por segundo.

```
% /. metros -> 3.28084 pies
```

$$\frac{2.27034\ \text{pies}}{\text{segundos}}$$

Hay de hecho un paquete estándar de *Mathematica* que le permite para trabajar con unidades. El paquete define muchos símbolos que representan los tipos estándar de unidades

Cargue el paquete de *Mathematica* para manejar unidades.

```
<<Miscellaneous`Units`
```

EL paquete usa nombres estandarizados para unidades.

```
12 Meter/Second
```

$$\frac{12\ \text{Meter}}{\text{Second}}$$

La función `Convert[`*expr*`,`*units*`]` convierte a unidades especificadas.

```
Convert[ %, Mile/Hour ]
```

$$\frac{37500\ \text{Mile}}{1397\ \text{Hour}}$$

Por lo general usted tiene que dar los prefijos para unidades como palabras separadas.

```
Convert[ 3 Kilo Meter / Hour, Inch / Minute ]
```

$$\frac{250000\ \text{Inch}}{127\ \text{Minute}}$$

Esto convierte la temperatura. `Convert` sólo convierte las unidades de temperatura.

```
ConvertTemperature[20, Fahrenheit, Centigrade]
```

$$-\frac{20}{3}$$

6. Matemáticas simbólicas

6.1. Operaciones básicas

La capacidad de *Mathematica* de tratar con expresiones simbólicas, así como números, le permite usarlo para muchas clases de matemáticas.

El cálculo es un ejemplo. Con *Mathematica*, usted puede diferenciar una expresión simbólicamente, y conseguir una fórmula para el resultado.

Esto encuentra la derivada de x^n.

```
D[ x^n, x ]
```

$$n\,x^{-1+n}$$

He aquí un ejemplo ligeramente más complicado.

```
D[x^2 Log[x + a], x]
```

$$\frac{x^2}{a+x} + 2\,x\,\text{Log}[a+x]$$

`D[`f`,`x`]`	la derivada (parcial) $\frac{\partial f}{\partial x}$
`Integrate[`f`,`x`]`	la integral indefinida $\int f dx$
`Sum[`f`,{`i`,`i_{min}`,`i_{max}`}]`	la suma $\sum_{i=i_{min}}^{i_{max}} f$
`Solve[`*lhs*`==`*rhs*`,`x`]`	solución para una ecuación en x
`Series[`f`,{`x`,`x_0`,`*orden*`}]`	una expansión de f en series de potencies alrededor del punto $x = x_0$
`Limit[`f`,`x`->`x_0`]`	el límite $\lim_{x \to x_0} f$
`Minimize[`f`,`x`]`	minimización de f con respecto a x

Algunas operaciones matemáticas simbólicas.

Obtener fórmulas como resultado de cómputos es generalmente deseable cuando es posible. Hay sin embargo muchas circunstancias donde es matemáticamente imposible conseguir una fórmula explícita como resultado de un cómputo. Esto sucede, por ejemplo, cuando intenta solucionar una ecuación para la cual no hay solución de "forma cerrada". En tales casos, debe recurrir a métodos y aproximaciones numéricas. Éstos se discuten en la sección 7.

6.2. Diferenciación

He aquí la derivada x^n con respecto a x.

```
D[ x^n, x ]
```

$$n\,x^{-1+n}$$

Mathematica conoce las derivadas de todas las funciones matemáticas estándar.

```
D[ ArcTan[x], x ]
```

$$\frac{1}{1+x^2}$$

La tercera derivada con respecto a x.

```
D[ x^n, {x, 3} ]
```

$$(-2+n)\,(-1+n)\,n\,x^{-3+n}$$

La función `D[x^n,x]` realmente da la derivada *parcial*, en la cual se asume que `n` no depende `x`. *Mathematica* tiene otra función, llamada `Dt`, que encuentra derivadas *totales*, en la cual todas las derivadas se asumen relacionadas. En notación matemática, `D[`f`,`x`]` es como $\frac{\partial f}{\partial x}$, mientras `Dt[`$f$`,`$x$`]` es como $\frac{df}{dx}$. Puede entenderse `Dt` como la "derivada total".

`Dt` da una derivada total, asumiendo que `n` puede depender de `x`. `Dt[n,x]` representa $\frac{dn}{dx}$.

```
Dt[ x^n, x ]
```

$$x^n\left(\frac{n}{x}+\texttt{Dt[n, x]}\;\texttt{Log[x]}\right)$$

Esto da la diferencial total $d(x^n)$. `Dt[x]` es el diferencial dx.

```
Dt[ x^n ]
```

$$x^n\left(\frac{n\,\texttt{Dt[x]}}{x}+\texttt{Dt[n]}\;\texttt{Log[x]}\right)$$

`D[f, x]`	derivada parcial $\frac{\partial}{\partial x} f$
`D[f, x1, x2, ...]`	derivada múltiple $\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_2}\ldots f$
`D[f, {x, n}]`	derivada repetida $\frac{\partial^n f}{\partial x^n}$
`Dt[f]`	diferencial total $d\,f$
`Dt[f, x]`	derivada total $\frac{d}{dx} f$

Algunas funciones de diferenciación.

Así como trata variables simbólicamente, usted también puede tratar funciones simbólicamente en *Mathematica*. Así, por ejemplo, puede encontrar fórmulas para las derivadas de `f[x]`, sin especificar una forma explícita para la función `f`.

Mathematica no sabe como diferenciar `f`, así que le devuelve un resultado simbólico en términos de `f'`.

```
D[ f[x], x ]
```
```
f'[x]
```

Mathematica utiliza la regla de cadena para simplificar derivadas.

```
D[ 2 x f[x^2], x ]
```
```
2 f[x^2] + 4 x^2 f'[x^2]
```

6.3. Integración

He aquí la integral $\int x^n dx$ en *Mathematica*.

```
Integrate[x^n, x]
```
```
x^(1+n)
-------
1 + n
```

He aquí un ejemplo ligeramente más complicado.

```
Integrate[1/(x^4 - a^4), x]
```
```
-ArcTan[x/a]/(2 a^3) + Log[a - x]/(4 a^3) - Log[a + x]/(4 a^3)
```

Mathematica sabe resolver casi cualquier integral que puede ser expresada en términos de funciones matemáticas estándares. Pero debe comprender que aun cuando un integrando pueda contener sólo funciones simples, su integral puede implicar funciones mucho más complicadas–o no puede ser expresable en absoluto en términos de funciones matemáticas estándares.

He aquí una integral simple.

```
Integrate[Log[1 - x^2], x ]
```

$$-2\,x - \text{Log}[-1+x] + \text{Log}[1+x] + x\,\text{Log}[1-x^2]$$

Esta integral puede ser expresada sólo en términos de una función de dilogarítmica.

```
Integrate[Log[1 - x^2]/x, x]
```

$$-\frac{1}{2}\,\text{PolyLog}[2, x^2]$$

Esta integral involucra `Erf`.

```
Integrate[Exp[1 - x^2], x]
```

$$\frac{1}{2}\,e\,\sqrt{\pi}\,\text{Erf}[x]$$

Y esta otra involucra una función de Fresnel.

```
Integrate[Sin[x^2], x]
```

$$\sqrt{\frac{\pi}{2}}\,\text{FresnelS}\left[\sqrt{\frac{2}{\pi}}\,x\right]$$

Esta integral incluso requiere una función hipergeométrica.

```
Integrate[(1 - x^2)^n, x]
```

$$x\,\text{Hypergeometric2F1}\left[\frac{1}{2}, -n, \frac{3}{2}, x^2\right]$$

Esta integral simplemente no puede ser expresada en términos de funciones matemáticas estándares. Por consiguiente, *Mathematica* la deja como está.

```
Integrate[ x^x, x ]
```

$$\int x^x\,dx$$

`Integrate[f, x]`	la integral indefinida $\int f\, dx$
`Integrate[f, x, y]`	la integral múltiple $\int dx\, dy\, f$
`Integrate[f, {x, xmin, xmax}]`	la integral definida $\int_{x_{min}}^{x_{max}} f\, dx$
`Integrate[f, {x, xmin, xmax}, {y, ymin, ymax}]`	la integral múltiple $\int_{x_{min}}^{x_{max}} dx \int_{y_{min}}^{y_{max}} dy\, f$

Integración.

He aquí la integral definida $\int_a^b \sin^2(x)\, dx$.

```
Integrate[Sin[x]^2, {x, a, b} ]
```

$$\frac{1}{2}\,(-a + b + \text{Cos[a] Sin[a]} - \text{Cos[b] Sin[b]})$$

He aquí otra integral definida.

```
Integrate[Exp[-x^2], {x, 0, Infinity}]
```

$$\frac{\sqrt{\pi}}{2}$$

Mathematica no puede darle una fórmula para esta integral definida.

```
Integrate[ x^x, {x, 0, 1} ]
```

$$\int_0^1 x^x\, dx$$

Aunque, puede conseguir un resultado numérico.

```
N[ % ]
0.783431
```

Esto evalúa la integral múltiple $\int_0^1 dx \int_0^x dy\,(x^2 + y^2)$. El rango de la variable de integración exterior aparece primero.

```
Integrate[ x^2 + y^2, {x, 0, 1}, {y, 0, x} ]
```

$$\frac{1}{3}$$

Esto integra x^{10} sobre una región circular.

```
Integrate[ x^10 Boole[x^2 + y^2 <= 1], {x, -1, 1},
{y, -1 ,1}]
```

$$\frac{21\pi}{512}$$

6.4. Sumas y productos

Esto construye la suma $\sum_{i=1}^{7} \frac{x^i}{i}$.

```
Sum[x^i/i, {i, 1, 7}]
```

$$x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \frac{x^5}{5} + \frac{x^6}{6} + \frac{x^7}{7}$$

Puede obviarse el límite inferior si éste es igual a 1.

```
Sum[x^i/i, {i, 7}]
```

$$x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \frac{x^5}{5} + \frac{x^6}{6} + \frac{x^7}{7}$$

Los productos se obtienen en forma similar a las sumas.

```
Product[x + i, {i, 1, 4}]
```

$$(1+x)\,(2+x)\,(3+x)\,(4+x)$$

`Sum[`f`, {`i`, `i_{min}`, `i_{max}`}]`	la suma $\sum_{i=i_{min}}^{i_{max}} f$
`Sum[`f`, {`i`, `i_{min}`, `i_{max}`, `di`}]`	la suma con i incrementándose en un paso de di.
`Sum[`f`, {`i`, `i_{min}`, `i_{max}`}, {`j`, `j_{min}`, `j_{max}`}]`	la suma anidada $\sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} f$
`Product[`f`, {`i`, `i_{min}`, `i_{max}`}]`	el producto $\prod_{i=i_{min}}^{i_{max}} f$

Sumas y productos.

La suma es calculada simbólicamente como una función de *n*.

```
Sum[i^2, {i, 1, n}]
```

$$\frac{1}{6} n (1+n) (1+2n)$$

Mathematica también puede dar un resultado exacto para esta suma infinita.

```
Sum[1/i^4, {i, 1, Infinity}]
```

$$\frac{\pi^4}{90}$$

Como en el caso de las integrales, sumas simples pueden conducir a resultados complicados.

```
Sum[x^(i (i + 1)), {i, 1, Infinity}]
```

$$\frac{-2 x^{1/4} + \text{EllipticTheta}[2, 0, x]}{2 x^{1/4}}$$

Esta suma no se puede evaluar exactamente usando funciones matemáticas estándares.

```
Sum[1/(i! + (2i)!), {i, 1, Infinity}]
```

$$\sum_{i=1}^{\infty} \frac{1}{i! + (2i)!}$$

Usted puede sin embargo encontrar una aproximación numérica para el resultado.

```
N[%]
0.373197
```

Mathematica también tiene una notación para sumas y productos múltiples. `Sum[`f`,{`i`,`i_{min}`,`i_{max}`},{`j`,`j_{min}`,`j_{max}`}]` representa una suma en i y j, que será escrita en notación matemática estándar como $\sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} f$. Note que en la notación de *Mathematica*, como en la notación matemática estándar, el rango de la variable exterior se da primero.

Esta es la suma múltiple $\sum_{i=1}^{3} \sum_{j=1}^{i} x^i y^j$. Note que la suma exterior en i se da primero, tal como en la notación matemática.

```
Sum[x^i y^j, {i, 1, 3}, {j, 1, i}]
```
$$x\,y+x^2\,y+x^3\,y+x^2\,y^2+x^3\,y^2+x^3\,y^3$$

La forma en que se especifica los rangos de las variables en `Sum` y `Product` es un ejemplo algo general de la notación de iteradores que *Mathematica* utiliza. Usted verá esta notación otra vez cuando discutamos la generación de tablas y listas usando `Table` (sección 9.2), y cuando describamos los lazos `Do` (sección 8.3).

$\{i_{max}\}$	iterador i_{max} sin incrementar variable alguna
$\{i, i_{max}\}$	i varía de 1 a i_{max} en pasos de 1
$\{i, i_{min}, i_{max}\}$	i varía de i_{min} a i_{max} en pasos de 1
$\{i, i_{min}, i_{max}, di\}$	i varía de i_{min} a i_{max} en pasos de di
$\{i, i_{min}, i_{max}\}$, $\{j, j_{min}, j_{max}\}$, ...	i varía de i_{min} a i_{max}, y para cada valor, j varía de j_{min} a j_{max}, etc.

Notación de iteradores en *Mathematica*.

6.5. Ecuaciones

En la sección 3.2 se discutió asignaciones tales como `x = y` que asigna a `x` el valor de `y`. Esta sección habla de ecuaciones, que prueban una igualdad. La ecuación `x == y` , prueba si `x` es igual a `y`.

Esto prueba si `2 + 2` y `4` son iguales. El resultado es el símbolo `True`.

```
2 + 2 == 4
True
```

Es muy importante que usted no confunda `x = y` con `x == y` . Mientras que `x = y` es una declaración imperativa que en realidad origina una asignación, `x == y` simplemente prueba si `x` y `y` son iguales, y no causa ninguna acción explícita.

x=y	asigna el valor de y a x
x==y	prueba si x e y son iguales

Asignaciones y pruebas.

Esto asigna a `x` el valor `4`.

```
x = 4
4
```

Si pregunta por x, obtiene 4.

```
x
4
```

Esto prueba si x es igual a 4. En este caso si es.

```
x == 4
True
```

x es igual a 4 no a 6.

```
x == 6
False
```

Esto remueve el valor asignado a x.

```
x = .
```

Las pruebas que hemos utilizado hasta ahora involucran solamente números, y dan siempre una respuesta definida, `True` o `False`. Usted también puede hacer pruebas en expresiones simbólicas.

Mathematica no puede obtener un resultado definido para esta comprobación sin que usted especifique un valor numérico para x.

```
x == 5
x == 5
```

Si reemplaza x por el valor numérico específico 4, la prueba da `False`.

```
% / x -> 4
False
```

Incluso cuando hace pruebas en expresiones simbólicas, hay algunos casos donde puede conseguir resultados definidos. Algo importante es cuando usted prueba la igualdad de dos expresiones que son *idénticas*. Independientemente de los valores numéricos que puedan tener las variables en estas expresiones, *Mathematica* sabe que las expresiones siempre deben ser iguales.

Las dos expresiones son *idénticas*, por eso el resultado es True, independientemente del valor que pueda tener x.

```
2 x + x^2 == 2 x + x^2
True
```

Mathematica no trata de averiguar si estas expresiones son iguales. En este caso, usar `Expand` haría que tuvieran la misma forma.

```
2 x + x^2 == x (2 + x)
```

$$2x + x^2 == x\,(2 + x)$$

Las expresiones como `x == 4` representan ecuaciones en *Mathematica*. Hay muchas funciones en *Mathematica* para manipular y solucionar ecuaciones.

Esta es una ecuación en *Mathematica*. La sección 6.7 discutirá cómo solucionarla para `x`.

```
x^2 + 2 x - 7 == 0
```

$$-7 + 2x + x^2 == 0$$

Puede asignar un nombre a la ecuación.

```
eqn = %
```

$$-7 + 2x + x^2 == 0$$

Si pregunta por `eqn`, obtiene la ecuación.

```
eqn
```

$$-7 + 2x + x^2 == 0$$

6.6. Operadores relacionales y lógicos

x==*y*	igual (también se ingresa como $x == y$)
x!=*y*	desigual (también se ingresa como $x \neq y$)
x>*y*	mayor que
x>=*y*	mayor o igual que (también se ingresa como $x \geq y$)
x<*y*	menor que
x<=*y*	mayor o igual que (también se ingresa como $x \leq y$)
x==*y*==*z*	todos iguales
x!=*y*!=*z*	todos desiguales (distintos)
x>*y*>*z*, etc.	estrictamente decreciente, etc.

Operadores relacionales.

Esto prueba si `10` es menor que `7`. El resultado es `False`.

```
10 < 7
False
```

No todos estos números son desiguales, por esta razón se obtiene `False`.

```
3 != 2 != 3
False
```

Puede mezclar < y <=.

```
3 < 5 <= 6
True
```

Ya que las cantidades involucradas son numéricas, *Mathematica* puede determinar que esto es verdadero.

```
Pi^E < E^Pi
True
```

Mathematica no sabe si esto es verdadero o falso.

```
x > y
x > y
```

!*p*	no (también se ingresa como ¬ *p*)
p&&*q*&&...	y (también se ingresa como *p* ∧ *q* ∧...)
p||*q*||...	o (también se ingresa como *p* ∨ *q* ∨...)
Xor[*p*,*q*,...]	o exclusiva (también se ingresa como *p* ⊻ *q* ⊻...)
Nand[*p*,*q*,...] y Nor[*p*,*q*,...]	nand y nor (también se ingresan como ⊼ y ⊽)
If[*p*,*then*,*else*]	da *then* si *p* es `True`, y *else* si *p* es `False`
LogicalExpand[*expr*]	expande expresiones lógicas

Operadores lógicos.

Ambas pruebas dan `True`, entonces el resultado es `True`.

```
7 > 4 && 2 != 3
True
```

Usted debe recordar que las operaciones lógicas ==, && y || tienen todas *doble caracter* en *Mathematica*. Si ha usado un lenguaje de programación como C, estará familiarizado con esta notación.

Mathematica no sabe si esto es verdadero o falso.

```
p && q
```
```
p && q
```

Mathematica deja esta expresión inalterada.

```
(p || q) && !(r || s)
```
```
(p || q) &&! (r || s)
```

Puede usar `LogicalExpand` para expandir los términos.

```
LogicalExpand[ % ]
```
```
(p && !r && !s) || (q && !r && !s)
```

6.7. Solución de Ecuaciones

Una expresión como `x^2 + 2 x - 7 == 0` representa una ecuación en *Mathematica*. A menudo tendrá que solucionar ecuaciones como esta, para averiguar para qué valores de `x` son verdaderas.

Esto da las dos soluciones de la ecuación cuadrática $x^2 + 2x - 7 = 0$. Las soluciones se dan como reemplazos para `x`.

```
Solve[x^2 + 2x - 7 == 0, x]
```
```
{{x → -1 - 2 √2}, {x → -1 + 2 √2}}
```

He aquí los valores numéricos de las soluciones.

```
N[ % ]
```
```
{{x → -3.82843}, {x → 1.82843}}
```

Usted puede obtener una lista de las soluciones actuales para `x` aplicando las reglas generadas por `Solve` a `x` mediante el operador de reemplazo.

```
x /. %
```
```
{-3.82843, 1.82843}
```

Igualmente puede aplicar los reemplazos a cualquier otra expresión que involucra a x.

```
x^2 + 3 x /. %%
{3.17157, 8.82843}
```

`Solve[`*lhs*`==`*rhs*`,`*x*`]`	resuelve una ecuación, dando una lista de reglas para *x*
x`/.`*solución*	use la lista de reglas para obtener valores para *x*
expr`/.`*solución*	use la lista de reglas para obtener valores para cualquier expresión

Encuentro y uso de soluciones de ecuaciones.

`Solve` siempre trata de darle fórmulas explícitas para las soluciones de ecuaciones. Sin embargo, es un resultado básico matemático que, para ecuaciones suficientemente complicadas, no pueden darse fórmulas algebraicas explícitas. Si usted tiene una ecuación algebraica en una variable, y la potencia más alta de la variable es menor que cinco, entonces *Mathematica* siempre puede darle fórmulas para las soluciones. Sin embargo, si la potencia más alta es cinco o más, puede ser matemáticamente imposible dar fórmulas algebraicas explícitas para todas las soluciones.

Mathematica siempre puede solucionar ecuaciones algebraicas en una variable cuando la potencia más alta es menor que cinco.

```
Solve[x^4 - 5 x^2 - 3 == 0, x]
```

$$\left\{\left\{x \to -\sqrt{\frac{5}{2}+\frac{\sqrt{37}}{2}}\right\}, \left\{x \to \sqrt{\frac{5}{2}+\frac{\sqrt{37}}{2}}\right\},\right.$$
$$\left.\left\{x \to -i\sqrt{\frac{1}{2}\left(-5+\sqrt{37}\right)}\right\}, \left\{x \to i\sqrt{\frac{1}{2}\left(-5+\sqrt{37}\right)}\right\}\right\}$$

Esto puede solucionar algunas ecuaciones que involucran potencias más altas.

```
Solve[x^6 == 1, x]
{{x → -1}, {x → 1}, {x → -(-1)^(1/3)},
 {x → (-1)^(1/3)}, {x → -(-1)^(2/3)}, {x → (-1)^(2/3)}}
```

Hay algunas ecuaciones, sin embargo, para las cuales es matemáticamente imposible encontrar fórmulas explícitas para las soluciones. *Mathematica* usa objetos `Root` para representar las soluciones en este caso.

```
Solve[2 - 4 x + x^5 == 0, x]
```

```
{{x → Root[2 - 4 #1 + #1^5 &, 1]},
 {x → Root[2 - 4 #1 + #1^5 &, 2]}, {x → Root[2 - 4 #1 + #1^5 &, 3]},
 {x → Root[2 - 4 #1 + #1^5 &, 4]}, {x → Root[2 - 4 #1 + #1^5 &, 5]}}
```

Incluso aunque usted no pueda conseguir fórmulas explícitas, puede encontrar las soluciones numéricamente.

```
N[ % ]
```

```
{{x → -1.51851}, {x → 0.508499}, {x → 1.2436},
 {x → -0.116792 - 1.43845 i}, {x → -0.116792 + 1.43845 i}}
```

Además de la capacidad de solucionar ecuaciones puramente algebraicas, *Mathematica* también puede solucionar algunas ecuaciones que implican otras funciones.

Después de la impresión de una advertencia, *Mathematica* devuelve una solución para esta ecuación.

```
Solve[ Sin[x] == a, x ]
```

```
Solve::ifun :
 Inverse functions are being used by Solve, so some solutions may
   not be found; use Reduce for complete solution information.
{{x → ArcSin[a]}}
```

Es importante comprender que una ecuación tal como $\sin(x) = a$ en realidad tiene un número infinito de soluciones posibles, en este caso que se diferencian por múltiplos de 2π. Sin embargo, `Solve` por defecto da sólo una solución, pero imprime un mensaje que le dice que pueden existir otras soluciones. Usted puede usar `Reduce` para obtener más información.

No hay ninguna solución explícita para una ecuación trascendental como esta.

```
Solve[ Cos[x] == x, x ]
```

```
Solve::tdep : The equations appear to involve the variables
   to be solved for in an essentially non-algebraic way.
Solve[Cos[x] == x, x]
```

Puede encontrar una solución numérica aproximada usando `FindRoot`, y dando un valor inicial para `x`.

```
FindRoot[ Cos[x] == x, {x, 0} ]
```

$\{x \to 0.739085\}$

`Solve` también puede manejar ecuaciones que involucran funciones simbólicas. En tales casos, otra vez imprime una advertencia, luego da resultados en términos de funciones formales inversas.

Mathematica devuelve un resultado en términos de la función formal inversa de `f`.

```
Solve[ f[x^2] == a, x ]
```

```
InverseFunction::ifun :  Inverse functions are
   being used. Values may be lost for multivalued inverses.
```

$$\left\{\left\{x \to -\sqrt{f^{(-1)}[a]}\right\}, \left\{x \to \sqrt{f^{(-1)}[a]}\right\}\right\}$$

`Solve[{`*lhs*$_1$`==`*rhs*$_1$`, `*lhs*$_2$`==`*rhs*$_2$`, ...}, {`*x*`, `*y*`, ...}]`	resuelve un conjunto de ecuaciones simultáneas para *x*, *y*, ...

Solución de conjuntos de ecuaciones simultáneas.

También puede usar *Mathematica* para solucionar conjuntos de ecuaciones simultáneas. Simplemente da la lista de ecuaciones, y especifica la lista de variables.

He aquí una lista de dos ecuaciones simultáneas, para que sean resueltas en las variables *x* e *y*.

```
Solve[{a x + y == 0, 2 x + (1-a) y == 1}, {x, y}]
```

$$\left\{\left\{x \to -\frac{1}{-2+a-a^2}, y \to -\frac{a}{2-a+a^2}\right\}\right\}$$

He aquí algunas ecuaciones simultáneas más complicadas. Las dos soluciones se dan como dos listas de reemplazos para `x`.

```
Solve[{x^2 + y^2 == 1, x + 3 y == 0}, {x, y}]
```

$$\left\{\left\{x \to -\frac{3}{\sqrt{10}}, y \to \frac{1}{\sqrt{10}}\right\}, \left\{x \to \frac{3}{\sqrt{10}}, y \to -\frac{1}{\sqrt{10}}\right\}\right\}$$

Esto usa las soluciones para evaluar la expresión `x + y`.

```
x + y /. %
```

$$\left\{-\sqrt{\frac{2}{5}}, \sqrt{\frac{2}{5}}\right\}$$

Mathematica puede solucionar cualquier sistema de ecuaciones simultáneas lineales. También puede solucionar gran número de ecuaciones polinómicas simultáneas. Incluso cuando no logra solucionar las ecuaciones explícitamente, *Mathematica* por lo general las reducirá a una forma mucho más simple.

Cuando usted trabaja con sistemas de ecuaciones en varias variables, es a menudo conveniente reorganizar las ecuaciones eliminando algunas variables entre ellos.

Esto elimina `y` entre las dos ecuaciones, dando una sola ecuación para `x`.

```
Eliminate[{a x + y == 0, 2 x + (1-a) y == 1}, y]
```

$$(2 - a + a^2)\, x == 1$$

Si usted tiene varias ecuaciones, no hay ninguna garantía que exista *cualquier* solución consistente para una variable particular.

No hay ninguna solución consistente para estas ecuaciones, así *Mathematica* devuelve `{}`, indicando que el conjunto de soluciones es vacío.

```
Solve[{x==1, x==2}, x]
{}
```

No hay tampoco ninguna solución consistente para estas ecuaciones para casi todos los valores de `a`.

```
Solve[{x==1, x==a}, x]
{}
```

La pregunta general de si un sistema de ecuaciones tiene cualquier solución consistente es bastante sutil. Por ejemplo, para la mayor parte de valores de `a`, las ecuaciones `{x==1, x==a}` son inconsistentes, así no hay ninguna solución posible para `x`. Sin embargo, si `a` es igual a `1`, entonces las ecuaciones tienen una solución. `Solve` da soluciones genéricas para ecuaciones. Desecha cualesquiera soluciones que existen sólo cuando los parámetros satisfacen condiciones especiales.

Si usa `Reduce` en vez de `Solve`, *Mathematica* mantendrá todas las posibles soluciones para un sistema de ecuaciones, incluyendo aquellas que requieren condiciones especiales en los parámetros.

Esto muestra que las ecuaciones tienen una solución sólo cuando `a==1`. La notación `a==1&&x==1` representa la exigencia que tanto `a==1` como `x==1` deben ser `True` (verdaderos).

```
Reduce[{x==a, x==1}, x]
```

```
a == 1 && x == 1
```

Esto da el conjunto completo de posibles soluciones para la ecuación. La respuesta es dada en términos de una combinación de ecuaciones más simples. `&&` indica que las ecuaciones deben ser simultáneamente verdaderas; `||` indica alternativas.

```
Reduce[a x - b == 0, x]
```

$$(b == 0 \&\& a == 0) \;||\; \left(a \neq 0 \&\& x == \frac{b}{a}\right)$$

Esto da una combinación más complicada de ecuaciones.

```
Reduce[a x^2 - b == 0, x]
```

$$(b == 0 \&\& a == 0) \;||\; \left(a \neq 0 \&\& \left(x == -\frac{\sqrt{b}}{\sqrt{a}} \;||\; x == \frac{\sqrt{b}}{\sqrt{a}}\right)\right)$$

Esto da una representación simbólica de todas las soluciones.

```
Reduce[Sin[x] == a, x]
```

```
C[1] ∈ Integers &&
  (x == π - ArcSin[a] + 2 π C[1] || x == ArcSin[a] + 2 π C[1])
```

`Solve[`*lhs*`==`*rhs*`,`*x*`]`	resuelve una ecuación en *x*
`Solve[{`lhs_1`==`rhs_1`, `lhs_2`==`rhs_2`, ...},{`*x*`,`*y*`, ...}]`	resuelve un conjunto de ecuaciones simultáneas en *x*, *y*, ...
`Eliminate[{`lhs_1`==`rhs_1`, `lhs_2`==`rhs_2`, ...},{`*x*`, ...}]`	elimina *x*, ... en un conjunto de ecuaciones simultáneas
`Reduce[{`lhs_1`==`rhs_1`, `lhs_2`==`rhs_2`, ...},{`*x*`,`*y*`, ...}]`	da un conjunto de ecuaciones simplificadas, incluyendo todas las posibles soluciones

Funciones para solucionar y manipular ecuaciones.

`Reduce` también tiene capacidades poderosas para manipular ecuaciones específicamente en números reales o enteros.

Esto reduce la ecuación asumiendo que x e y son complejos.

```
Reduce[x^2 + y^2 == 1, y]
```

$$y == -\sqrt{1 - x^2} \;||\; y == \sqrt{1 - x^2}$$

Esto incluye las condiciones de que x e y son reales.

```
Reduce[x^2 + y^2 == 1, y, Reals]
```

$$-1 \le x \le 1 \;\&\&\; \left(y == -\sqrt{1 - x^2} \;||\; y == \sqrt{1 - x^2}\right)$$

Esto da solamente soluciones enteras.

```
Reduce[x^2 + y^2 == 1, y, Integers]
```

```
(x == -1 && y == 0) || (x == 0 && y == -1) ||
  (x == 0 && y == 1) || (x == 1 && y == 0)
```

6.8. Desigualdades

`Reduce[`*ineqs*`, {`*x*`, `*y*`, ...}]`	reduce una colección de desigualdades
`FindInstance[`*ineqs*`, {`*x*`, `*y*`, ...}]`	encuentra un caso que satisface *ineqs*

Manejo de desigualdades.

Esto encuentra una forma reducida para las desigualdades.

```
Reduce[x + y < 1 && y > x > 0, {x, y}]
```

$$0 < x < \frac{1}{2} \;\&\&\; x < y < 1 - x$$

Estas desigualdades no pueden ser satisfechas.

```
Reduce[x + y < 1 && y > x > 1, {x, y}]
```

```
False
```

Es fácil encontrarse con resultados algo complicados.

```
Reduce[x + y < 1 && y^2 > x > 0, {x, y}]
```

$$\left(0 < x < \frac{1}{2}\left(3-\sqrt{5}\right) \&\& \left(y < -\sqrt{x} \;||\; \sqrt{x} < y < 1-x\right)\right) ||$$
$$\left(\frac{1}{2}\left(3-\sqrt{5}\right) \le x < \frac{1}{2}\left(3+\sqrt{5}\right) \&\& y < -\sqrt{x}\right) ||$$
$$\left(x \ge \frac{1}{2}\left(3+\sqrt{5}\right) \&\& y < 1-x\right)$$

Las ecuaciones a menudo pueden ser solucionadas para dar valores definidos de las variables. Pero las desigualdades solamente definen regiones que pueden ser especificadas por otras desigualdades. Usted puede usar `FindInstance` para encontrar valores definidos de las variables que satisfacen un conjunto particular de desigualdades.

Esto encuentra un punto en la región especificada por las desigualdades.

```
FindInstance[x + y < 1 && y^2 > x > 0, {x, y}]
```

$$\left\{\left\{x \to \frac{7}{2}, y \to -3\right\}\right\}$$

`Minimize[{`*expr*`,`*ineq*`}, {`*x*`, `*y*`, ...}]`	minimiza *expr* mientras satisfaga *ineq*
`Maximize[{`*expr*`,`*ineq*`}, {`*x*`, `*y*`, ...}]`	maximiza *expr* mientras satisfaga *ineq*

Minimización y maximización sujetas.

Esto da el máximo, junto con el valor donde ocurre.

```
Maximize[{x^2 + y, x^2 + y^2 <= 1}, {x, y}]
```

$$\left\{\frac{5}{4}, \left\{x \to -\frac{\sqrt{3}}{2}, y \to \frac{1}{2}\right\}\right\}$$

Esto da el mínimo, junto con el valor donde ocurre.

```
Minimize[{x^2 + y, x^2 + y^2 <= 1}, {x, y}]
```

$$\{-1, \{x \to 0, y \to -1\}\}$$

6.9. Ecuaciones diferenciales

`DSolve[`*eqns*`, y[x], x]`	resuelve una ecuación diferencial para `y[x]`, tomando a *x* como variable independiente
`DSolve[`*eqns*`, y, x]`	da una solución para *y* en forma de función pura

Solución de una ecuación diferencial ordinaria.

He aquí la solución de la ecuación diferencial $y'(x) = a\,y(x) + 1$. C[1] es un coeficiente que debe ser determinado a partir condiciones de frontera.

```
DSolve[ y'[x] == a y[x] + 1, y[x], x ]
```

$$\left\{\left\{y[x] \to -\frac{1}{a} + e^{a x}\, C[1]\right\}\right\}$$

Si usted incluye una condición inicial apropiada, no hay ningún coeficiente indeterminado en la solución.

```
DSolve[ {y'[x] == a y[x] + 1, y[0] == 0}, y[x], x ]
```

$$\left\{\left\{y[x] \to \frac{-1 + e^{a x}}{a}\right\}\right\}$$

Mientras que ecuaciones algebraicas tales como $x^2 + x = 1$ son ecuaciones para *variables*, ecuaciones diferenciales tales como $y''(x) + y'(x) = y(x)$ son ecuaciones para *funciones*. En *Mathematica*, siempre debe dar las ecuaciones diferenciales explícitamente en términos de funciones como `y[x]`, y debe especificar las variables como `x` de las cuales dependen las funciones. Por consiguiente, debe escribir una ecuación tal como $y''(x) + y'(x) = y(x)$ en la forma `y''[x]+y'[x]==y[x]`. Usted no debe escribirla como `y''+y'==y`.

Mathematica puede solucionar ecuaciones diferenciales ordinarias tanto lineales como no lineales, así como listas de ecuaciones simultáneas. Si no especifica las suficientes condiciones iniciales o de frontera, *Mathematica* dará las soluciones con un número apropiado de coeficientes indeterminados. Cada vez que usted usa `DSolve`, éste representa a los coeficientes indeterminados por `C[1]`, `C[2]`, etc.

He aquí un par de ecuaciones diferenciales simultáneas, sin condiciones iniciales o de frontera. La solución que se obtiene tiene dos coeficientes indeterminados.

```
DSolve[ {x'[t] == y[t], y'[t] == x[t]}, {x[t],
y[t]}, t ]
```

$$\left\{\left\{x[t] \to \frac{1}{2} e^{-t} \left(1 + e^{2t}\right) C[1] + \frac{1}{2} e^{-t} \left(-1 + e^{2t}\right) C[2],\right.\right.$$
$$\left.\left. y[t] \to \frac{1}{2} e^{-t} \left(-1 + e^{2t}\right) C[1] + \frac{1}{2} e^{-t} \left(1 + e^{2t}\right) C[2]\right\}\right\}$$

Cuando usted le pide a `DSolve` que encuentre una solución para `y[x]`, las reglas que devuelve especifican como sustituir `y[x]` en cualquier expresión. Sin embargo, estas reglas no especifican como sustituir objetos como `y'[x]`. Si usted quiere manipular soluciones obtenidas con `DSolve`, a menudo encontrará mejor pedir soluciones para `y`, antes que para `y[x]`.

Esto da la solución para `y` como una "función pura".

```
DSolve[ y'[x] == x + y[x], y, x ]
```

$$\{\{y \to \text{Function}[\{x\}, -1 - x + e^{x} C[1]]\}\}$$

Usted puede ahora usar al operador de reemplazo para aplicar esta solución a expresiones que involucran a `y`.

```
y''[x] + y[x] /. %
```

$$\{-1 - x + 2 e^{x} C[1]\}$$

Note que `DSolve` puede manejar combinaciones de ecuaciones diferenciales algebraicas. También puede manejar ecuaciones diferenciales parciales, en las cuales hay más de una variable independiente.

6.10. Series de potencias

Las operaciones matemáticas de las que hemos hablado hasta ahora son exactas. Considerando la entrada exacta, sus resultados son fórmulas exactas.

En muchas situaciones, sin embargo, usted no necesita un resultado exacto. Puede ser suficiente, por ejemplo, encontrar un fórmula aproximada que es válida, digamos, cuando la cantidad `x` es pequeña.

Esto da una aproximación en serie de potencias para $(1+x)^n$ alrededor de 0, hasta los términos de orden 3.

```
Series[(1 + x)^n, {x, 0, 3}]
```

$$1 + n x + \frac{1}{2} (-1 + n) n x^2 + \frac{1}{6} (-2 + n) (-1 + n) n x^3 + O[x]^4$$

Mathematica conoce las expansiones en serie de potencias para muchas funciones matemáticas.

```
Series[Exp[-a t] (1 + Sin[2 t]), {t, 0, 4}]
```

$$1 + (2 - a)\,t + \left(-2a + \frac{a^2}{2}\right) t^2 + \left(-\frac{4}{3} + a^2 - \frac{a^3}{6}\right) t^3 + \left(\frac{4a}{3} - \frac{a^3}{3} + \frac{a^4}{24}\right) t^4 + O[t]^5$$

Si usted da una función que no conoce, `Series` escribe la serie de potencias en términos de derivadas.

```
Series[1 + f[t], {t, 0, 3}]
```

$$1 + f[0] + f'[0]\,t + \frac{1}{2} f''[0]\,t^2 + \frac{1}{6} f^{(3)}[0]\,t^3 + O[t]^4$$

Las series de potencias son fórmulas aproximadas que juegan el mismo papel con respecto a las expresiones algebraicas como los números aproximados con las expresiones numéricas. *Mathematica* le permite realizar operaciones en series potencias, en todos los casos mantiene el orden apropiado o el "grado de precisión" para las series de potencias resultantes.

He aquí una serie de potencias simple, de orden 3.

```
Series[Exp[x], {x, 0, 3}]
```

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + O[x]^4$$

Cuando usted hace las operaciones en una serie de potencias, el resultado es calculado sólo en el orden apropiado en `x`.

```
%^2 (1 + %)
```

$$2 + 5x + \frac{13x^2}{2} + \frac{35x^3}{6} + O[x]^4$$

Esto cambia la serie de potencias en una expresión ordinaria.

```
Normal[%]
```

$$2 + 5x + \frac{13x^2}{2} + \frac{35x^3}{6}$$

Ahora el cuadrado es calculado *exactamente*.

```
%^2
```

$$\left(2+5x+\frac{13x^2}{2}+\frac{35x^3}{6}\right)^2$$

Al aplicar `Expand` se obtiene un resultado con once términos.

```
Expand[%]
```

$$4+20x+51x^2+\frac{265x^3}{3}+\frac{1207x^4}{12}+\frac{455x^5}{6}+\frac{1225x^6}{36}$$

`Series[`*expr*`, {`*x*`, `x_0`, `*n*`}]`	encuentra la expansión en serie de potencias de *expr* alrededor del punto $x = x_0$ de *n*-ésimo orden
`Normal[`*series*`]`	trunca una serie de potencias para dar una expresión ordinaria

Operaciones en series de potencias.

6.11. Límites

He aquí la expresión $\sin(x)/x$.

```
t = Sin[x]/x
```

$$\frac{\text{Sin[x]}}{x}$$

Si sustituye `x` por `0`, la expresión se hace `0/0`, y obtiene un resultado indeterminado.

```
t /. x->0
```

```
Power::infy : Infinite expression 1/0 encountered.
∞::indet : Indeterminate expression 0 ComplexInfinity encountered.
Indeterminate
```

Si encuentra el valor numérico de $\sin(x)/x$ para un x próximo a `0`, usted consigue un resultado próximo a `1`.

```
t /. x->0.01
0.999983
```

Esto encuentra el *límite* de $\sin(x)/x$ cuando x tiende a `0`. El resultado es ciertamente `1`.

```
Limit[t, x->0]
1
```

`Limit[`*expr*`, `x`->`x_0`]`	el límite de *expr* cuando x tiende a x_0

Límites.

6.12. Transformadas integrales

`LaplaceTransform[`*expr*`, `t`, `s`]`	encuentra la transformada de Laplace de *expr*
`InverseLaplaceTransform[`*expr*`, `t`, `s`]`	encuentra la transformada inversa de Laplace de *expr*

Transformadas de Laplace.

Esto calcula la transformada de Laplace.

```
LaplaceTransform[t^3 Exp[a t], t, s]
```

$$\frac{6}{(a-s)^4}$$

He aquí la transformada inversa.

```
InverseLaplaceTransform[%, s, t]
```

$$e^{at} t^3$$

`FourierTransform[`*expr*`, `t`, `w`]`	encuentra la transformada simbólica de Fourier de *expr*
`InverseFourierTransform[`*expr*`, `t`, `w`]`	encuentra la transformada inversa de Fourier de *expr*

Transformadas de Fourier.

Esto calcula la transformada de Fourier.

```
FourierTransform[t^4 Exp[-t^2], t, w]
```

$$\frac{e^{-\frac{w^2}{4}} \left(12 - 12\, w^2 + w^4\right)}{16\sqrt{2}}$$

He aquí la transformada inversa.

```
InverseFourierTransform[%, w, t]
```

$$e^{-t^2}\, t^4$$

Note que en la literatura científica y técnica muchas convenciones diferentes son usadas para definir transformadas de Fourier.

6.13. Ecuaciones recurrentes

`RSolve[`*eqns*`, a[n], n]`	resuelve las ecuaciones recurrentes *eqns* para `a[n]`

Solución de ecuaciones recurrentes.

Esto resuelve una ecuación recurrente simple.

```
RSolve[{a[n] == 3 a[n-1]+1, a[1]==1},
a[n], n]
```

$$\left\{\left\{a[n] \to \frac{1}{2}\left(-1+3^n\right)\right\}\right\}$$

He aquí una solución más complicada de otra ecuación recurrente.

```
RSolve[{a[n+1] == (a[n]+1)/(n+1), a[1]==0},
a[n], n]
```

$$\left\{\left\{a[n] \to \frac{\text{Gamma}[-1,-1]+(-1)^n\,\text{Gamma}[1+n]\,\text{Gamma}[-n,-1]}{e\,\text{Gamma}[1+n]}\right\}\right\}$$

He aquí la solución de una ecuación recurrente con dos condiciones iniciales.

```
RSolve[{a[n+2] == (3 a[n+1]-a[n])/2, a[1]==0,
a[2]==1}, a[n], n]
```

$$\{\{a[n] \to 2^{1-n}\left(-2+2^n\right)\}\}$$

6.14. Paquetes para matemáticas simbólicas

Hay muchos paquetes de *Mathematica* que implementan operaciones matemáticas simbólicas. Esta sección da algunos ejemplos del conjunto paquetes estándares distribuidos con *Mathematica*. Sin embargo, debe tener en cuenta que algunas

copias de *Mathematica* pueden inicializarse de modo que las funciones aquí descritas sean cargadas automáticamente si son alguna vez necesarias.

Análisis vectorial

`<<Calculus`VectorAnalysis``	carga el paquete de análisis vectorial
`SetCoordinates[`*system*`[`*names*`]]`	especifica el sistema de coordenadas a usarse (`Cartesian`, `Cylindrical`, `Spherical`, etc.), dando los nombres de las coordenadas en ese sistema
`Grad[`*f*`]`	evalúa el gradiente ∇f de f en el sistema de coordenadas escogido
`Div[`*f*`]`	evalúa la divergencia $\nabla \cdot f$ de la lista f
`Curl[`*f*`]`	evalúa el rotacional $\nabla \times f$ de la lista f
`Laplacian[`*f*`]`	evalúa el laplaciano $\nabla^2 f$ de f

Análisis vectorial.

Esto carga el paquete de análisis vectorial. En algunas versiones de *Mathematica*, puede que no tenga que cargar el paquete explícitamente.

```
<<Calculus`VectorAnalysis`
```

Esto especifica que usted desea usar un sistema de coordenadas esférico con las coordenadas `r`, `theta` y `phi`.

```
SetCoordinates[Spherical[r, theta, phi]]
```

```
Spherical[r, theta, phi]
```

Esto evalúa el gradiente de $r^2 \sin(\theta)$ en el sistema de coordenadas esféricas.

```
Grad[r^2 Sin[theta]]
```

```
{2 r Sin[theta], r Cos[theta], 0}
```

Métodos variacionales

`<<Calculus`VariationalMethods``	carga el paquete de métodos variacionales
`VariationalD[f, y[x], x]`	encuentra la derivada variacional de *f*

Métodos variacionales.

Esto carga el paquete de métodos variacionales.

```
<<Calculus`VariationalMethods`
```

Esto encuentra la derivada funcional de $y(x)\sqrt{y'(x)}$.

```
VariationalD[y[x] Sqrt[y'[x]], y[x], x]
```

$$\frac{2\,y'[x]^2 + y[x]\,y''[x]}{4\,y'[x]^{3/2}}$$

Cuaterniones

`<<Algebra`Quaternions``	carga el paquete de cuaterniones
`Quaternion[a, b, c, d]`	el cuaternión $a + bi + cj + dk$

Cuaterniones.

Esto carga el paquete de cuaterniones.

```
<<Algebra`Quaternions`
```

Esto encuentra la principal raíz cuadrada de un cuaternión.

```
Sqrt[Quaternion[1, 1, 1, 0]]
```

$$\text{Quaternion}\left[3^{1/4}\cos\left[\frac{\text{ArcTan}\left[\sqrt{2}\right]}{2}\right],\ \frac{3^{1/4}\sin\left[\frac{\text{ArcTan}\left[\sqrt{2}\right]}{2}\right]}{\sqrt{2}},\ \frac{3^{1/4}\sin\left[\frac{\text{ArcTan}\left[\sqrt{2}\right]}{2}\right]}{\sqrt{2}},\ 0\right]$$

6.15. Casos genéricos y no genéricos

Esto da un resultado para la integral de x^n que es válida para casi todos los valores de *n*.

```
Integrate[x^n, x]
```

$$\frac{x^{1+n}}{1+n}$$

Para el caso especial de x^{-1}, sin embargo, el resultado correcto es diferente.

```
Integrate[x^-1, x]
Log[x]
```

El objetivo general del cálculo simbólico es conseguir fórmulas que son válidas para muchos valores posibles de las variables que aparecen en ellas. No es a menudo práctico tratar de obtener fórmulas que son válidas para absolutamente todo valor posible de cada variable.

Mathematica siempre reemplaza $0/x$ por 0.

```
0 / x
0
```

Si *x* es igual 0, sin embargo, el resultado no es 0.

```
0 / 0
```

Power::infy : Infinite expression $\frac{1}{0}$ encountered.

∞::indet : Indeterminate expression 0 ComplexInfinity encountered.

```
Indeterminate
```

Esta construcción trata ambos casos, pero sería poco utilizable.

```
If[x != 0, 0, Indeterminate]
If[x ≠ 0, 0, Indeterminate]
```

Si *Mathematica* no sustituyera automáticamente $0/x$ por 0, entonces pocos cálculos simbólicos serían posibles. Pero debe comprender que la necesidad práctica de hacer tales reemplazos puede hacer que se obtengan resultados absurdos cuando se usan valores específicos de los parámetros.

Las operaciones básicas de *Mathematica* sin embargo son instaladas con cuidado de modo que siempre sea posible que los resultados obtenidos sean válidos para casi todos los valores de cada variable.

$\sqrt{x^2}$ no es automáticamente reemplazada por *x*.

```
Sqrt[x^2]
```

$\sqrt{x^2}$

Si así fuera, entonces este resultado sería `-2`, que es incorrecto.

```
% /. x -> -2
```

2

Esto hace que *x* se asuma como una variable real positiva, y la reemplaza.

```
Simplify[Sqrt[x^2], x > 0]
```

x

Esto hace que *x* se asuma como una variable real negativa, y la reemplaza.

```
Simplify[Sqrt[x^2], x < 0]
```

x

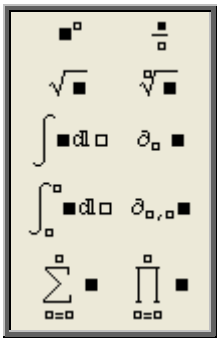
6.16. Notación matemática en cuadernos

Si usted usa una interfase de cuaderno para *Mathematica* (sección 4.1), entonces usted puede ingresar algunas operaciones mencionadas en esta sección de modo especial.

$\sum_{i=i_{min}}^{i_{max}} f$	`Sum[`*f*`, {`*i*`, `i_{min}`, `i_{max}`}]`	suma
$\prod_{i=i_{min}}^{i_{max}} f$	`Product[`*f*`, {`*i*`, `i_{min}`, `i_{max}`}]`	producto
$\int f \, dx$	`Integrate[`*f*`, `*x*`]`	integral indefinida
$\int_{x_{min}}^{x_{max}} f \, dx$	`Integrate[`*f*`, {`*x*`, `x_{min}`, `x_{max}`}]`	integral definida
$\partial_x f$	`D[`*f*`, `*x*`]`	derivada parcial
$\partial_{x,y} f$	`D[`*f*`, `*x*`, `*y*`]`	derivada parcial múltiple

Formas especiales y ordinarias de ingresar operaciones matemáticas en cuadernos.

Esto muestra parte de la paleta estándar para ingresar operaciones matemáticas. Cuando usted presiona un botón en la paleta, la forma mostrada en el botón se inserta en su cuaderno, con el cuadrado negro substituido por lo que usted había seleccionado en el cuaderno.



Esc sum Esc	signo de sumatoria $\sum$
Esc prod Esc	signo de producto $\prod$
Esc int Esc	signo de integral $\int$
Esc dd Esc	diferencial especial d para uso en integrales
Esc pd Esc	derivada parcial ∂
Ctrl-[_] ó Ctrl-[-]	mueve a la posición de subíndice o límite inferior de un integral
Ctrl-[^] ó Ctrl-[6]	mueve a la posición de superíndice o límite superior de un integral
Ctrl-[+] ó Ctrl-[=]	mueve a la posición subescritura o límite inferior de una suma o producto
Ctrl-[&] ó Ctrl-[7]	mueve a la posición de sobreescritura o límite superior de una suma o producto
Ctrl-[%] ó Ctrl-[5]	cambia entre posiciones superiores e inferiores
Ctrl-[␣] (Control-Espacio)	retorna de posiciones superiores o inferiores

Formas de ingresar notaciones especiales en un teclado estándar de lenguaje Inglés.

Usted puede ingresar un integral como esta. Asegúrese de usar la diferencial especial d ingresada como Esc dd Esc, no sólo una d ordinaria.

$$\int x^n \, \mathrm{d}x$$

$$\frac{x^{1+n}}{1+n}$$

Aquí está la secuencia real clave que usted escribe para conseguir la entrada.

```
esc int esc x ctrl [^] n ctrl [_] esc dd esc x
```

$$\frac{x^{1+n}}{1+n}$$

7. Matemáticas numéricas

7.1. Operaciones básicas

Los resultados exactos simbólicos son por lo general muy deseados cuando pueden ser obtenidos. En muchos cálculos, sin embargo, no es posible conseguir resultados simbólicos. En tales casos, usted debe recurrir a métodos numéricos.

`N[`*expr*`]`	valor numérico de una expresión (ver sección 2)
`NIntegrate[`*f*`, {`*x*`, `x_{min}`, `x_{max}`}]`	aproximación numérica para $\int_{x_{min}}^{x_{max}} f\,dx$
`NSum[`*f*`, {`*i*`, `i_{min}`, Infinity}]`	aproximación numérica para $\sum_{i_{min}}^{\infty} f$
`FindRoot[`*lhs*`==`*rhs*`, {`*x*`, `x_0`}]`	encuentra la solución numérica para una ecuación, empezando con $x = x_0$
`NSolve[`*lhs*`==`*rhs*`, `*x*`]`	aproximaciones numéricas de todas las soluciones de una ecuación
`FindMinimum[`*f*`, {`*x*`, `x_0`}]`	encuentra un mínimo de *f*, empezando con $x = x_0$
`NMinimize[`*f*`, `*x*`]`	intenta encontrar el mínimo global de *f*

Operaciones numéricas básicas.

Mathematica mantiene esta expresión en forma simbólica exacta.

```
(3 + Sqrt[2])^3
```

$$(3+\sqrt{2})^3$$

Usted puede incluso usar operaciones estándares simbólicas en ella.

```
Expand[ % ]
```

$$45 + 29\sqrt{2}$$

`N[`*expr*`]` da una aproximación numérica.

```
N[ % ]
86.0122
```

Funciones como `Integrate` siempre procuran obtener resultados exactos para los cálculos. Cuando no pueden obtenerlos, devuelven los resultados sin evaluar. Usted entonces puede encontrar aproximaciones numéricas explícitamente aplicando `N`. Funciones como `NIntegrate` hacen los cálculos numéricamente desde el principio, sin ninguna tentativa de obtener un resultado exacto.

No hay una fórmula exacta para esta integral, por lo cual *Mathematica* la devuelve sin evaluar.

```
Integrate[Sin[Sin[x]], {x, 1, 2}]
```

$$\int_1^2 \text{Sin[Sin[x]]}\, dx$$

Puede usar `N` para obtener un resultado numérico aproximado.

```
N[ % ]
0.81645
```

`NIntegrate` resuelve la integral en forma numérica desde el inicio.

```
NIntegrate[Sin[Sin[x]], {x, 1, 2}]
0.81645
```

7.2. Sumas, productos e integrales numéricos

`NSum[`f`, {`i`, `i_{min}`, Infinity}]`	aproximación numérica para $\sum_{i_{min}}^{\infty} f$
`NProduct[`f`, {`i`, `i_{min}`, Infinity}]`	aproximación numérica para $\prod_{i_{min}}^{\infty} f$
`NIntegrate[`f`, {`x`, `x_{min}`, `x_{max}`}]`	aproximación numérica para $\int_{x_{min}}^{x_{max}} f\, dx$
`NIntegrate[`f`, {`x`, `x_{min}`, `x_{max}`}, {`y`, `y_{min}`, `y_{max}`}]`	la integral múltiple $\int_{x_{min}}^{x_{max}} dx \int_{y_{min}}^{y_{max}} dy\, f$

Sumas, productos e integrales numéricos.

He aquí una aproximación numérica $\sum_{i=1}^{\infty} \frac{1}{i^3}$.

```
NSum[1/i^3, {i, 1, Infinity}]
1.20206
```

`NIntegrate` puede manejar singularidades en los puntos finales de la región de integración.

```
NIntegrate[1/Sqrt[x (1-x)], {x, 0, 1}]
3.14159
```

Puede resolver integrales numéricas sobre regiones infinitas.

```
NIntegrate[Exp[-x^2], {x, -Infinity, Infinity}]
1.77245
```

He aquí una integral doble sobre un dominio triangular. Note el orden en el cual se ingresan las variables.

```
NIntegrate[ Sin[x y], {x, 0, 1}, {y, 0, x} ]
0.119906
```

He aquí una integral doble sobre un dominio limitado por dos parábolas.

```
NIntegrate[ x^2 + y^2, {x, -1, 1}, {y, x^2 - 1,
-x^2 + 1}]
1.14286
```

7.3. Solución numérica de ecuaciones

`NSolve[`*lhs*`==`*rhs*`, `*x*`]`	soluciona numéricamente una ecuación polinomial
`NSolve[{`lhs_1`==`rhs_1`, `lhs_2`==`rhs_2`, ...},{`*x*`, `*y*`, ...}]`	soluciona numéricamente un sistema de ecuaciones polinomiales
`FindRoot[`*lhs*`==`*rhs*`, {`*x*`, `x_0`}]`	encuentra una solución numérica para una ecuación, empezando en $x = x_0$
`FindRoot[{`lhs_1`==`rhs_1`, `lhs_2`==`rhs_2`, ...},{{`*x*`, `x_0`}, {`*x*`, `x_0`},...}]`	encuentra soluciones numéricas para ecuaciones simultáneas

Búsqueda de raíces numéricas.

`NSolve` le da aproximaciones numéricas a todas las raíces de una ecuación polinomial.

```
NSolve[ x^5 + x + 1 == 0, x ]
{{x → -0.754878}, {x → -0.5 - 0.866025 i},
 {x → -0.5 + 0.866025 i}, {x → 0.877439 - 0.744862 i},
 {x → 0.877439 + 0.744862 i}}
```

También puede usar `NSolve` para resolver numéricamente sistemas simultáneos de ecuaciones.

```
NSolve[{x + y == 2, x - 3 y + z == 3,
x - y + z == 0}, {x, y, z}]
{{x → 3.5, y → -1.5, z → -5.}}
```

Si sus ecuaciones involucran sólo funciones lineales o polinómicas, entonces puede usar `NSolve` para obtener aproximaciones numéricas de todas las soluciones. Sin embargo, cuando sus ecuaciones involucran funciones más complicadas, no hay en general ningún procedimiento sistemático para obtener todas las soluciones, aún numéricamente. En tales casos, puede usar `FindRoot` para buscar soluciones. Tiene que dar a `FindRoot` un valor para que empiece la búsqueda.

Esto encuentra una solución numérica, empezando en $x = 1$.

```
FindRoot[ 3 Cos[x] == Log[x], {x, 1} ]
{x → 1.44726}
```

La ecuación tiene varias soluciones. Si usted comienza con un x diferente, `FindRoot` puede devolver una solución diferente.

```
FindRoot[ 3 Cos[x] == Log[x], {x, 10} ]
{x → 13.1064}
```

Usted puede buscar soluciones para sistemas de ecuaciones. Aquí la solución involucra números complejos.

```
FindRoot[{x == Log[y], y == Log[x]}, {{x, I},
{y, 2}}]
{x → 0.318132 + 1.33724 i, y → 0.318132 + 1.33724 i}
```

7.4. Ecuaciones diferenciales numéricas

`NDSolve[`*eqs*`, `*y*`, {`*x*`, `x_{min}`, `x_{max}`}]`	soluciona numéricamente para la función *y*, con la variable independiente *x* en el rango x_{min} a x_{max}
`NDSolve[`*eqs*`, {`y_1`, `y_2`, ...}, {`*x*`, `x_{min}`, `x_{max}`}]`	soluciona un sistema de ecuaciones para las y_i

Solución numérica de ecuaciones diferenciales.

Esto genera una solución numérica para la ecuación $y'(x) = y(x)$ con $0 < x < 2$. El resultado está dado en términos de una función interpolante.

```
NDSolve[{y'[x] == y[x], y[0] == 1}, y, {x, 0, 2}]
{{y → InterpolatingFunction[{{0., 2.}}, <>]}}
```

He aquí el valor de $y(1.5)$.

```
y[1.5] /. %
{4.48169}
```

Con una ecuación algebraica como $x^2 + 3x + 1$, cada solución para x es simplemente un número. Para una ecuación diferencial, sin embargo, la solución es una función. Por ejemplo, en la ecuación $y'(x) = y(x)$, usted obtiene una aproximación a la función $y(x)$ con la variable independiente x que varía sobre algún rango.

Mathematica representa aproximaciones numéricas a funciones como objetos. Estos objetos son las funciones que, cuando se aplican a un *x* particular, devuelven el valor aproximado de $y(x)$ en aquel punto. `InterpolatingFunction` en forma eficaz almacena una tabla de valores para $y(x_i)$, luego interpola esta tabla para encontrar una aproximación para $y(x)$ en el *x* particular que usted solicita.

`y[ `*x*`]/.`*solución*	usa la lista de reglas de la función *y* para obtener valores para `y[ `*x*`]`
`InterpolatingFunction[`*data*`, `*x*`]`	evalúa una función interpolada en el punto *x*
`Plot[Evaluate[y[ `*x*`]/.`*solución*`],{`*x*`, `x_{min}`, `x_{max}`}]`	grafica la solución de la ecuación diferencial

Uso de los resultados de `NDSolve`.

Esto soluciona un sistema de dos ecuaciones diferenciales simultáneas.

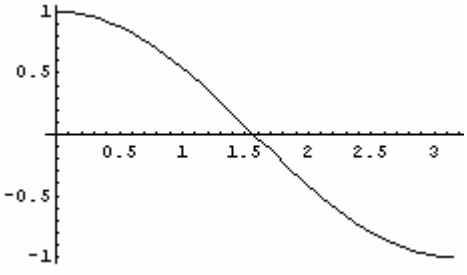
```
NDSolve[ {y'[x] == z[x], z'[x] == -y[x], y[0] == 0,
z[0] == 1}, {y, z}, {x, 0, Pi} ]
{{y → InterpolatingFunction[{{0., 3.14159}}, <>],
  z → InterpolatingFunction[{{0., 3.14159}}, <>]}}
```

He aquí el valor de `z[2]` hallado a partir de la solución.

```
z[2] /. %
{-0.416147}
```

He aquí el gráfico de la solución para `z[x]` encontrada anteriormente. `Plot` se discute en la sección 10.1.

```
Plot[Evaluate[z[x] /. %%], {x, 0, Pi}]
```



- Graphics -

`NDSolve[`*eqn*`, `*u*`, {`*x*`, `x_{min}`, `x_{max}`}, {`*t*`, `t_{min}`, `t_{max}`}, ...]`	resuelve una ecuación diferencial parcial

Solución numérica de ecuaciones diferenciales parciales.

7.5. Optimización numérica

`NMinimize[`*f*`, {`*x*`, `*y*`, ...}]`	minimiza *f*
`NMaximize[`*f*`, {`*x*`, `*y*`, ...}]`	maximiza *f*
`NMinimize[{`*f*`, `*ineqs*`}, {`*x*`, `*y*`, ...}]`	minimiza *f* sujeto a las restricciones *ineqs*
`NMaximize[{`*f*`, `*ineqs*`}, {`*x*`, `*y*`, ...}]`	maximiza *f* sujeto a las restricciones *ineqs*

Obtención de máximos y mínimos globales.

Esto da el valor máximo, y dónde es que ocurre.

```
NMaximize[x/(1 + Exp[x]), x]
{0.278465, {x → 1.27846}}
```

Esto minimiza la función dentro del círculo unitario.

```
NMinimize[{Cos[x] - Exp[x y], x^2 + y^2 < 1},
{x, y}]
{-0.919441, {x → 0.795976, y → 0.605328}}
```

`FindMinimum[`*f*`, {`*x*`, `x_0`}]`	busca un mínimo local de *f*, comenzando en $x = x_0$
`FindMinimum[`*f*`, {{`*x*`, `x_0`}, {`*y*`, `y_0`}, ...}]`	busca un mínimo local en varias variables
`FindMaximum[`*f*`, {`*x*`, `x_0`}]`	busca un máximo local de *f*, comenzando en $x = x_0$
`FindMaximum[`*f*`, {{`*x*`, `x_0`}, {`*y*`, `y_0`}, ...}]`	busca un máximo local en varias variables

Búsqueda de máximos y mínimos locales.

`NMinimize` y `NMaximize` pueden encontrar los mínimos y máximos absolutos de muchas funciones. Pero en algunos casos no es realista hacer esto. Usted puede buscar mínimos y máximos locales usando `FindMinimum` y `FindMaximum`.

Esto busca un mínimo local de $x\cos(x)$, empezando en $x = 2$.

```
FindMinimum[x Cos[x], {x, 2}]
{-3.28837, {x → 3.42562}}
```

Con un punto de partida diferente, usted puede obtener un mínimo local diferente.

```
FindMinimum[x Cos[x], {x, 10}]
{-9.47729, {x → 9.52933}}
```

Esto encuentra un mínimo local de $\sin(x\,y)$.

```
FindMinimum[Sin[x y], {{x, 2}, {y, 2}}]
{-1., {x → 2.1708, y → 2.1708}}
```

7.6. Manipulación de datos numéricos

Cuando usted tiene datos numéricos, es a menudo conveniente encontrar una fórmula simple que los aproxime. Por ejemplo, usted puede tratar de "ajustar" ("fit") una línea o curva tomando los puntos de sus datos.

`Fit[{`y_1`, `y_2`, ...}, {`f_1`, `f_2`, ...}, `x`]`	ajusta los valores y_n para una combinación lineal de funciones f_i
`Fit[{{`x_1`, `y_1`}, {`x_2`, `y_2`}, ...}, {`f_1`, `f_2`, ...}, `x`]`	ajusta los puntos (x_n, y_n) para una combinación lineal de las f_i

Ajuste de curvas para una combinación lineal de funciones.

Esto genera una tabla de valores numéricos de la función exponencial. `Table` se discutirá en la sección 9.2.

```
data = Table[ Exp[x/5.] , {x, 7}]
{1.2214, 1.49182, 1.82212,
 2.22554, 2.71828, 3.32012, 4.0552}
```

Esto encuentra un ajuste de mínimos cuadrados para `data` de la forma $c_1 + c_2 x + c_3 x^2$. Los elementos de `data` se hacen corresponder con valores 1, 2, ... de *x*.

```
Fit[data, {1, x, x^2}, x]
1.09428 + 0.0986337 x + 0.0459482 x^2
```

Esto encuentra un ajuste de la forma $c_1 + c_2 x + c_3 x^3 + c_4 x^5 + c_5 x^7$.

```
Fit[data, {1, x, x^3, x^5, x^7}, x]
0.978681 + 0.237121 x + 0.00516759 x^3 -
 0.0000373676 x^5 + 3.30667 × 10^-7 x^7
```

Esto da una tabla de pares x, y.

```
data = Table[ {x, Exp[Sin[x]]} , {x, 0., 1., 0.2}]
{{0., 1.}, {0.2, 1.21978}, {0.4, 1.47612},
 {0.6, 1.75882}, {0.8, 2.04901}, {1., 2.31978}}
```

Esto encuentra un ajuste para los nuevos datos, de la forma $c_1 + c_2 \sin(x) + c_3 \sin(2x)$.

```
Fit[%, {1, Sin[x], Sin[2x]}, x]
0.989559 + 2.04199 Sin[x] - 0.418176 Sin[2 x]
```

`FindFit[`*data*`,` *form*`,` `{`p_1`,` p_2`, ...}, `*x*`]`	encuentra un ajuste para *form* con parámetros p_i

Ajuste de datos en formas generales.

Esto encuentra los mejores parámetros para un ajuste general.

```
FindFit[data, a + b x + c x^2, {a, b, c}, x]
{a → 0.991251, b → 1.16421, c → 0.174256}
```

Este no es un ajuste lineal.

```
FindFit[data, a + b^(c + d x), {a, b, c, d}, x]
{a → -3.65199, b → 1.65839, c → 3.03491, d → 0.501061}
```

Una forma común de elegir "señales" en datos numéricos es encontrar la *transformada de Fourier*, o espectro de frecuencia, de los datos.

`Fourier[`*data*`]`	transformada numérica de Fourier
`InverseFourier[`*data*`]`	transformada inversa de Fourier

Transformadas de Fourier.

He aquí un pulso cuadrado simple.

```
data = {1, 1, 1, 1, -1, -1, -1, -1}
{1, 1, 1, 1, -1, -1, -1, -1}
```

Esto toma la transformada de Fourier del pulso.

```
Fourier[data]
{0. + 0. ⅈ, 0.707107 + 1.70711 ⅈ,
 0. + 0. ⅈ, 0.707107 + 0.292893 ⅈ, 0. + 0. ⅈ,
 0.707107 - 0.292893 ⅈ, 0. + 0. ⅈ, 0.707107 - 1.70711 ⅈ}
```

Note que la función `Fourier` en *Mathematica* es definida con la convención de signo típicamente usada en las ciencias físicas en contraste a la que a menudo es usada en ingeniería eléctrica.

7.7. Estadística

`Mean[`*data*`]`	media (promedia valores)
`Median[`*data*`]`	mediana (valor central)
`Variance[`*data*`]`	varianza
`StandardDeviation[`*data*`]`	desviación estándar
`Quantile[`*data*`, `*q*`]`	cuantil q-ésimo
`Total[`*data*`]`	total de valores

Estadística descriptiva básica.

He aquí algunos datos.

```
data = {4.3, 7.2, 8.4, 5.8, 9.2, 3.9}
```
```
{4.3, 7.2, 8.4, 5.8, 9.2, 3.9}
```

Esto da la media de sus datos.

```
Mean[data]
```
```
6.46667
```

He aquí la varianza.

```
Variance[data]
```
```
4.69467
```

El conjunto de paquetes estándares distribuidos con *Mathematica* incluyen varios comandos para hacer análisis estadísticos de datos más sofisticados.

`Statistics`DescriptiveStatistics``	funciones de estadística descriptiva
`Statistics`MultivariateDescriptiveStatistics``	funciones de estadística descriptiva multivariante
`Statistics`ContinuousDistributions``	propiedades de distribuciones estadísticas continuas
`Statistics`DiscreteDistributions``	propiedades de distribuciones estadísticas discretas
`Statistics`HypothesisTests``	contrastes de hipótesis basados en la distribución normal
`Statistics`ConfidenceIntervals``	intervalos de confianza sacados de la distribución normal

`Statistics`MultinormalDistribution``	propiedades de distribuciones basadas en la distribución normal multivariante
`Statistics`LinearRegression``	análisis de regresión lineal
`Statistics`NonlinearFit``	ajuste no lineal de datos
`Statistics`DataSmoothing``	alisamiento de datos
`Statistics`DataManipulation``	utilidades para manipulación de datos

Algunos paquetes estándares de análisis estadístico.

8. Funciones y programas

8.1. Definición de funciones

En esta parte del libro, hemos visto muchos ejemplos de funciones que son construidas en *Mathematica*. En esta sección, mostramos como usted puede añadir sus propias funciones simples a *Mathematica*. La parte 2 describirá en mayor detalle los mecanismos para añadir funciones a *Mathematica*.

Como un primer ejemplo, piense añadir una función llamada `f` que eleva al cuadrado su argumento. El comando en *Mathematica* para definir esta función es `f[x_]:=x^2`. El _ (subguión) del miembro de la izquierda es muy importante; lo que esto significa será mencionado debajo. Por ahora, solamente acuérdese de poner un _ en el miembro de la izquierda, pero no en el miembro de la derecha, de su definición.

Esto define la función `f`. Note el _ en el miembro de la izquierda.

```
f[x_] := x^2
```

`f` eleva al cuadrado su argumento.

```
f[a+1]
```

$$(1+a)^2$$

El argumento puede ser un número.

```
f[4]
```

```
16
```

O puede ser una expresión más complicada.

```
f[3x + x^2]
```

$$(3x + x^2)^2$$

Usted puede usar `f` en un cálculo.

```
Expand[f[(x+1+y)]]
```

$$1 + 2x + x^2 + 2y + 2xy + y^2$$

Esto muestra la definición que usted hizo para `f`.

```
?f
Global`f
f[x_] := x^2
```

`f[x_]:=x^2`	define la función `f`
`?`*f*	muestra la definición de *f*
`Clear[`*f*`]`	borra todas las definiciones de *f*

Definición de una función en *Mathematica*.

Los nombres como `f` que usa para funciones en *Mathematica* son solamente símbolos. A causa de esto, debería evitar usar nombres que comienzan con mayúsculas, para prevenir la confusión con las funciones predefinidas en *Mathematica*. También debería asegurarse que no ha usado los nombres anteriormente en su sesión.

Las funciones en *Mathematica* pueden tener cualquier número de argumentos.

```
hump[x_, xmax_] := (x - xmax)^2 / xmax
```

Usted puede usar la función `hump` tal como cualquiera de las funciones predefinidas.

```
2 + hump[x, 3.5]
```

$$2 + 0.285714\,(-3.5 + x)^2$$

Esto da una nueva definición para `hump`, que sobrescribe la anterior.

```
hump[x_, xmax_] := (x - xmax)^4
```

La nueva definición es mostrada.

```
?hump
```

```
Global`hump
hump[x_, xmax_] := (x - xmax)^4
```

Esto limpia todas las definiciones para `hump`.

```
Clear[hump]
```

Cuando ha terminado con una función particular, es siempre buena idea limpiar definiciones que haya hecho para ella. Si no hace esto, entonces podría incurrir en un problema si trata de usar la misma función para un propósito diferente más adelante en su sesión de *Mathematica*. Usted puede limpiar todas las definiciones que haya hecho para una función o símbolo *f* usando `Clear[`*f*`]`.

8.2. Funciones como procedimientos

En muchas clases de cálculos, usted puede encontrarse digitando la misma entrada a *Mathematica* muchas veces. Usted puede ahorrarse mucha digitación definiendo una *función* que contiene sus comandos de entrada.

Esto construye un producto de tres términos, y expande el resultado.

```
Expand[ Product[x + i, {i, 3}] ]
```

$$6 + 11\,x + 6\,x^2 + x^3$$

Esto hace lo mismo, pero con cuatro términos.

```
Expand[ Product[x + i, {i, 4}] ]
```

$$24 + 50\,x + 35\,x^2 + 10\,x^3 + x^4$$

Esto define una función `exprod` que construye un producto de *n* términos, luego lo expande.

```
exprod[n_] := Expand[ Product[ x + i, {i, 1, n} ] ]
```

Siempre que usa la función, ejecutará las operaciones `Product` y `Expand`.

```
exprod[5]
```

$$120 + 274\,x + 225\,x^2 + 85\,x^3 + 15\,x^4 + x^5$$

Las funciones que define en *Mathematica* son esencialmente procedimientos que ejecutan los comandos que usted da. Puede tener varios pasos en sus procedimientos, separados por puntos y coma.

El resultado que obtiene de la función es simplemente la última expresión en el procedimiento. Note que tiene que poner paréntesis alrededor del procedimiento cuando lo define.

```
cex[n_, i_] := ( t = exprod[n];
Coefficient[t, x^i] )
```

Esto "corre" el procedimiento.

```
cex[5, 3]
```
```
85
```

$expr_1$; $expr_2$; ...	una secuencia de expresiones para evaluar
Module[{*a*, *b*, ...}, *proc*]	un procedimiento con variables locales *a*, *b*, ...

Construcción de procedimientos.

Cuando usted escribe procedimientos en *Mathematica*, es por lo general una buena idea declarar como *locales* las variables que usa dentro de los procedimientos, de modo que no interfieran con cosas fuera de los procedimientos. Puede hacer esto estableciendo sus procedimientos como *módulos*, en los cuales da una lista de variables para ser tratadas como locales.

La función `cex` arriba definida no es un módulo, así que el valor de `t` "escapa", y existe incluso después de la evaluación de la función.

```
t
```
$$120 + 274\,x + 225\,x^2 + 85\,x^3 + 15\,x^4 + x^5$$

Esta función es definida como un módulo con variable local `u`.

```
ncex[n_, i_] := Module[{u}, u = exprod[n];
Coefficient[u, x^i]]
```

La función devuelve el mismo resultado que la anteriormente definida.

```
ncex[5, 3]
```
```
85
```

Ahora, sin embargo, el valor de u no se escapa de la función.

```
u
```
```
u
```

8.3. Operaciones repetitivas

En el uso de *Mathematica*, a veces tiene que repetir una operación muchas veces. Hay muchos modos de hacer esto. A menudo el más natural es de hecho establecer una estructura como una lista con muchos elementos, y luego aplicar su operación a cada uno de los elementos.

Otra forma es usar la función de *Mathematica* `Do`, que trabaja de manera muy parecida a los iteradores en lenguajes como C y Fortran. `Do` usa la notación estándar de *Mathematica* para iteradores introducida para `Sum` y `Product` en la sección 6.4.

`Do[`*expr*`, {`*i*`, `*imax*`}]`	evalúa *expr* con *i* variando de `1` a *imax*
`Do[`*expr*`, {`*i*`, `*imin*`, `*imax*`, `*di*`}]`	evalúa *expr* con *i* variando de *imin* a *imax* en pasos de *di*
`Print[`*expr*`]`	imprime *expr*
`Table[`*expr*`, {`*i*`, `*imax*`}]`	hace una lista de valores de *expr* con *i* variando de `1` a *imax*

Implementación de operaciones repetitivas.

Esto imprime las salidas de los valores de los cinco primeros factoriales.

```
Do[ Print[i!], {i, 5} ]
```
```
1
2
6
24
120
```

A menudo es más útil tener una lista de los resultados, que usted luego puede manipular posteriormente.

```
Table[ i!, {i, 5} ]
```
```
{1, 2, 6, 24, 120}
```

Si usted no da una variable de iteración, *Mathematica* simplemente repite la operación que usted ha especificado, sin cambiar algo.

```
r = 1; Do[ r = 1/(1 + r), {100} ]; r
```

$$\frac{573147844013817084101}{927372692193078999176}$$

8.4. Reglas de transformación para funciones

La sección 5.2 mencionó como usted puede usar las reglas de transformación de la forma `x->`*valor* para sustituir símbolos por valores. La noción de reglas de transformación en *Mathematica* es, sin embargo, bastante general. Usted puede establecer reglas de transformación no sólo para símbolos, sino para cualquier expresión de *Mathematica*.

Aplicación de la regla de transformación `x->3` que reemplaza `x` por `3`.

```
1 + f[x] + f[y] /. x -> 3
```

$$1 + f[3] + f[y]$$

Puede usar una regla de transformación para `f[x]`. Esta regla no afecta a `f[y]`.

```
1 + f[x] + f[y] /. f[x] -> p
```

$$1 + p + f[y]$$

`f[t_]` es un *patrón* que admite cualquier argumento para `f`.

```
1 + f[x] + f[y] /. f[t_] -> t^2
```

$$1 + x^2 + y^2$$

Probablemente el aspecto más potente de las reglas de transformación en *Mathematica* es que ellas pueden involucrar expresiones no sólo literales, sino también *patrones*. Un patrón es una expresión como `f[t_]` que contiene un _ (subguión). El subguión representa cualquier expresión. Así, una regla de transformación para `f[t_]` especifica cómo la función `f` con *cualquier* argumento debería ser transformada. Note que, en contraste, una regla de transformación para `f[x]` sin un subguión, especifica sólo como la expresión literal `f[x]` debería ser transformada, y por ejemplo, no dice nada sobre la transformación de `f[y]`.

Cuando usted da una definición de función como `f[t_]:=t^2`, todo lo que hace es decir a *Mathematica* que aplique automáticamente la regla de transformación `f[t_]->t^2` siempre que es posible.

Usted puede establecer reglas de transformación para expresiones de cualquier forma.

```
f[a b] + f[c d] /. f[x_ y_] -> f[x] + f[y]
```

```
f[a] + f[b] + f[c] + f[d]
```

Esto usa una regla de transformación para `x^p_`.

```
1 + x^2 + x^4 /. x^p_ -> f[p]
```

```
1 + f[2] + f[4]
```

Las secciones 2.3 y 2.5 explicarán detalladamente como establecer patrones y reglas de transformación para cualquier clase de expresión. Basta decir aquí que en *Mathematica* todas las expresiones tienen una estructura simbólica definida; las reglas de transformación le permiten transformar partes de aquella estructura.

9. Listas

9.1. Juntar objetos

Primero encontramos listas en la sección 3.3 como una forma de agrupar números. En esta sección, veremos diferentes formas de usar listas. Usted encontrará que las listas son algunos de los objetos más flexibles y poderosos en *Mathematica*. Verá que las listas en *Mathematica* representan las generalizaciones de varios conceptos estándar en matemáticas e informática.

En un nivel básico, lo que esencialmente hace una lista en *Mathematica* es proporcionar una forma de agrupar varias expresiones de cualquier clase.

He aquí una lista de números.

```
{2, 3, 4}
```

```
{2, 3, 4}
```

Esto da una lista de expresiones simbólicas.

```
x^% - 1
```

$$\{-1 + x^2, -1 + x^3, -1 + x^4\}$$

Usted puede derivar estas expresiones.

```
D[%, x]
```

$$\{2x, 3x^2, 4x^3\}$$

Y luego puede encontrar los valores cuando `x` es reemplazado por `3`.

```
% /. x -> 3
```

$$\{6, 27, 108\}$$

Las funciones matemáticas que son construidas en *Mathematica* sobre todo son hechas "listables" de modo que actúen separadamente sobre cada elemento de una lista. Esto no es, sin embargo, cierto para todas las funciones en *Mathematica*. A no ser que usted las establezca, como una nueva función `f` que le permita introducir listas como argumentos.

9.2. Fabricación de tablas de valores

Usted puede usar listas como tablas de valores. Puede generar las tablas, por ejemplo, evaluando una expresión para una secuencia de diferentes valores de un parámetro.

Esto da una tabla de los valores de i^2, con i variando de 1 a 6.

```
Table[i^2, {i, 6}]
```

$$\{1, 4, 9, 16, 25, 36\}$$

He aquí una tabla de $\sin(n/5)$ para n de 0 a 4.

```
Table[Sin[n/5], {n, 0, 4}]
```

$$\left\{0, \text{Sin}\left[\frac{1}{5}\right], \text{Sin}\left[\frac{2}{5}\right], \text{Sin}\left[\frac{3}{5}\right], \text{Sin}\left[\frac{4}{5}\right]\right\}$$

Esto da los valores numéricos.

```
N[%]
```

$$\{0., 0.198669, 0.389418, 0.564642, 0.717356\}$$

También puede hacer tablas de fórmulas.

```
Table[x^i + 2i, {i, 5}]
```

$$\{2 + x, 4 + x^2, 6 + x^3, 8 + x^4, 10 + x^5\}$$

`Table` usa exactamente la misma notación para el iterador que las funciones `Sum` y `Product`, que fueron mencionadas en la sección 6.4.

```
Product[x^i + 2i, {i, 5}]
```

$$(2+x)\ (4+x^2)\ (6+x^3)\ (8+x^4)\ (10+x^5)$$

Esto hace una tabla con valores de x variando desde 0 hasta `1` en pasos de `0.25`.

```
Table[Sqrt[x], {x, 0, 1, 0.25}]
```

```
{0, 0.5, 0.707107, 0.866025, 1.}
```

Usted puede realizar otras operaciones en las listas que obtiene con `Table`.

```
%^2 + 3
```

```
{3, 3.25, 3.5, 3.75, 4.}
```

`TableForm` muestra las listas en un formato "tabular". Note que ambas palabras en `TableForm` empiezan con mayúsculas.

```
% // TableForm
```

```
3
3.25
3.5
3.75
4.
```

Todos los ejemplos hasta ahora han sido de tablas obtenidas variando un sólo parámetro. Usted también puede hacer tablas que involucran varios parámetros. Estas tablas multidimensionales son especificadas usando las notaciones estándares para el iterador en *Mathematica*, mencionadas en la sección 6.4.

Esto hace una tabla de $x^i + y^j$ con i variando desde 1 hasta 3 y j variando desde 1 hasta 2.

```
Table[x^i + y^j, {i, 3}, {j, 2}]
```

$$\{\{x+y,\ x+y^2\},\ \{x^2+y,\ x^2+y^2\},\ \{x^3+y,\ x^3+y^2\}\}$$

La tabla en este ejemplo es una *lista de listas*. Los elementos de la lista externa corresponden a los valores sucesivos de i. Los elementos de cada lista interior corresponden a los valores sucesivos de j, con i fijo.

A veces puede querer generar una tabla evaluando una expresión particular muchas veces, sin incrementar alguna variable.

Esto crea una lista conteniendo cuatro copias del símbolo `x`.

```
Table[x, {4}]
{x, x, x, x}
```

Esto da una lista de cuatro números seudo aleatorios. `Table` reevalúa `Random[]` para cada elemento en al lista, de modo que usted consigue un número diferente pseudoarbitrario.

```
Table[Random[ ], {4}]
{0.0560708, 0.6303, 0.359894, 0.871377}
```

`Table[`*f*`, {`i_{max}`}]`	da una lista de i_{max} valores de *f*
`Table[`*f*`, {`*i*`, `i_{max}`}]`	da una lista de los valores de *f* con *i* variando de 1 a i_{max}
`Table[`*f*`, {`*i*`, `i_{min}`, `i_{max}`}]`	da una lista de valores con *i* variando de i_{min} a i_{max}
`Table[`*f*`, {`*i*`, `i_{min}`, `i_{max}`, `*di*`}]`	usa pasos de *di*
`Table[`*f*`, {`*i*`, `i_{min}`, `i_{max}`}, {`*j*`, `j_{min}`, `j_{max}`}]`	genera una tabla multidimensional
`TableForm[`*list*`]`	muestra una lista en forma tabular

Funciones para generar tablas.

Usted puede usar las operaciones mencionadas en la sección 3.4 para extraer los elementos de la tabla.

Esto crea una tabla de 2×2, y le da el nombre `m`.

```
m = Table[i - j, {i, 2}, {j, 2}]
{{0, -1}, {1, 0}}
```

Esto extrae la primera sublista de la lista de listas, `m`.

```
m[[1]]
{0, -1}
```

Esto extrae el segundo elemento de aquella sublista.

```
%[[2]]
-1
```

Esto hace las dos operaciones juntas.

```
m[[1,2]]
-1
```

Esto muestra `m` en forma "tabular".

```
TableForm[m]
0  -1
1  0
```

t`[[`i`]]` o `Part[`t`, `i`]`	da la i-ésima sublista en t (también acepta la forma $t[[i]]$)
t`[[{`i_1`, `i_2`, ...}]]` o `Part[`t`, {`i_1`, `i_2`, ...}]`	da una lista de las i_1, i_2, ... partes de t
t`[[`i`, `j`, ...]]` o `Part[`t`, `i`, `j`, ...]`	da la parte de t correspondiente a $t[[i]][[j]]$ ...

Implementación de operaciones repetitivas.

Como mencionamos en la sección 3.4, usted puede usar las listas en *Mathematica* de modo análogo a los "arrays". Las listas de listas se usan entonces como series bidimensionales. Cuando usted los presenta en una forma tabular, los dos índices de cada elemento son como sus coordenadas x e y.

Puede usar `Table` para generar arrays de cualquier dimensión.

Esto genera un array tridimensional de 2×2×2. Es una lista de listas de listas.

```
Table[i j^2 k^3, {i, 2}, {j, 2}, {k, 2}]
{{{1, 8}, {4, 32}}, {{2, 16}, {8, 64}}}
```

9.3. Vectores y matrices

Vectores y matrices en *Mathematica* simplemente son representados por listas y por listas de listas, respectivamente.

`{`a`, `b`, `c`}`	vector (a, b, c)
`{{`a`, `b`}, {`c`, `d`}}`	matriz $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$

Representación de vectores y matrices por listas.

Esta es una matriz 2×2.

```
m = {{a, b}, {c, d}}
```
```
{{a, b}, {c, d}}
```

He aquí la primera fila.

```
m[[1]]
```
```
{a, b}
```

He aquí el elemento m_{12} .

```
m[[1,2]]
```
```
b
```

Este es un vector de dos componentes.

```
v = {x, y}
```
```
{x, y}
```

Los objetos p y q son tratados como escalares.

```
p v + q
```
```
{q + p x, q + p y}
```

Los vectores son sumados componente más componente.

```
v + {xp, yp} + {xpp, ypp}
```
```
{x + xp + xpp, y + yp + ypp}
```

Esto calcula el producto punto (“escalar”) de dos vectores.

```
{x, y} . {xp, yp}
```
```
x xp + y yp
```

Usted también puede multiplicar una matriz por un vector.

```
m . v
```
```
{a x + b y, c x + d y}
```

O una matriz por una matriz.

```
m . m
```
```
{{a^2 + b c, a b + b d}, {a c + c d, b c + d^2}}
```

O un vector por una matriz.

```
v . m
{a x + c y, b x + d y}
```

Esta combinación genera un escalar.

```
v . m . v
x (a x + c y) + y (b x + d y)
```

Debido a la forma en que *Mathematica* usa listas para representar vectores y matrices, usted nunca tiene que distinguir entre los vectores de la "fila" y de la "columna".

`Table[`*f*`, {`*i*`, `*n*`}]`	construye un vector de longitud *n* evaluando *f* para *i* = 1, 2, ..., *n*
`Array[`*a*`, `*n*`]`	construye un vector de longitud *n* de la forma `{`*a*`[1], `*a*`[2], ...}`
`Range[`*n*`]`	crea la lista `{1, 2, 3 ... , `*n*`}`
`Range[`n_1`, `n_2`]`	crea la lista `{ `n_1`, `n_1` + 1, ... , `n_2`}`
`Range[`n_1`, `n_2`, `*dn*`]`	crea la lista `{ `n_1`, `n_1` + `*dn*`, ... , `n_2`}`
list`[[`*i*`]]` o `Part[`*list*`, `*i*`]`	da el *i*-ésimo elemento del vector *list*
`Length[`*list*`]`	da el número de elementos de *list*
`ColumnForm[`*list*`]`	muestra los elementos de *list* en una columna
c v	multiplica por un escalar
a . *b*	producto punto de vectores
`Cross[`*a*`, `*b*`]`	producto vectorial de vectores
`Norm[`*b*`]`	norma de un vector

Funciones para vectores.

`Table[`*f*`, {`*i*`, `*n*`}, {`*j*`, `*m*`}]`	construye una matriz *m*×*n* evaluando *f* para *i* variando desde 1 hasta *m* y *j* variando desde 1 hasta *n*
`Array[`*a*`, `*m*`, `*n*`]`	construye una matriz *m*×*n* con elementos *a*`[`*i*`, `*j*`]`
`IdentityMatrix[`*n*`]`	genera una matriz identidad *n*×*n*
`DiagonalMatrix[`*list*`]`	genera una matriz cuadrada con los elementos de *list* en la diagonal
list`[[`*i*`]]` o `Part[`*list*`, `*i*`]`	da la *i*-ésima fila de la matriz *list*

list[[All, *j*]] o Part[*list*, All, *j*]	da la *j*-ésima columna de la matriz *list*
list[[*i*, *j*]] o Part[*list*, *i*, *j*]	da el elemento *i*, *j* de la matriz *list*
Dimensions[*list*]	da las dimensiones de la matriz *list*
MatrixForm[*list*]	muestra *list* en forma matricial

Funciones para matrices.

Esto construye una matriz 3×3 con elementos $s_{i,j} = i + j$.

```
s = Table[i+j, {i, 3}, {j, 3}]
```
```
{{2, 3, 4}, {3, 4, 5}, {4, 5, 6}}
```

Esto muestra `s` en un formato estándar de matriz bidimensional.

```
MatrixForm[s]
```
$$\begin{pmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \end{pmatrix}$$

Esto da un vector con elementos simbólicos. Usted puede usar esto para obtener fórmulas generales válidas con cualquier tipo de componentes de un vector.

```
Array[a, 4]
```
```
{a[1], a[2], a[3], a[4]}
```

Esto da una matriz 3×2 con elementos simbólicos.

```
Array[p, {3, 2}]
```
```
{{p[1, 1], p[1, 2]}, {p[2, 1], p[2, 2]}, {p[3, 1], p[3, 2]}}
```

He aquí las dimensiones de la matriz previamente definida.

```
Dimensions[%]
```
```
{3, 2}
```

Esto genera una matriz diagonal 3×3.

```
DiagonalMatrix[{a, b, c}]
```
```
{{a, 0, 0}, {0, b, 0}, {0, 0, c}}
```

c m	multiplicación por un escalar
a . *b*	producto matricial
`Inverse[`*m*`]`	matriz inversa
`MatrixPower[`*m*`, `*n*`]`	*n*-ésima potencia de una matriz
`Det[`*m*`]`	determinante
`Tr[`*m*`]`	traza
`Transpose[`*m*`]`	transpuesta
`Eigenvalues[`*m*`]`	valores propios
`Eigenvectors[`*m*`]`	vectores propios

Algunas operaciones matemáticas en matrices.

He aquí la matriz 2×2 de variables simbólicas que fue definida anteriormente.

```
m
```

```
{{a, b}, {c, d}}
```

Esto da su determinante.

```
Det[m]
```

```
-b c + a d
```

He aquí la transpuesta de `m`.

```
Transpose[m]
```

```
{{a, c}, {b, d}}
```

Esto da la inversa de `m` en forma simbólica.

```
Inverse[m]
```

$$\left\{\left\{\frac{d}{-b\,c+a\,d}, -\frac{b}{-b\,c+a\,d}\right\}, \left\{-\frac{c}{-b\,c+a\,d}, \frac{a}{-b\,c+a\,d}\right\}\right\}$$

He aquí una matriz racional 3×3.

```
h = Table[1/(i+j-1), {i, 3}, {j, 3}]
```

$$\left\{\left\{1, \frac{1}{2}, \frac{1}{3}\right\}, \left\{\frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right\}, \left\{\frac{1}{3}, \frac{1}{4}, \frac{1}{5}\right\}\right\}$$

Esto da su inversa.

```
Inverse[h]
```

```
{{9, -36, 30}, {-36, 192, -180}, {30, -180, 180}}
```

Multiplicando la inversa con la matriz original da la matriz identidad.

```
% . h
{{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}
```

He aquí una matriz 3×3.

```
r = Table[i+j+1, {i, 3}, {j, 3}]
{{3, 4, 5}, {4, 5, 6}, {5, 6, 7}}
```

`Eigenvalues` da los valores propios de la matriz.

```
Eigenvalues[r]
```

$$\left\{\frac{1}{2}\left(15+\sqrt{249}\right),\ \frac{1}{2}\left(15-\sqrt{249}\right),\ 0\right\}$$

`Eigenvectors` da los vectores propios de la matriz.

```
Eigenvectors[r]
```

$$\left\{\left\{-\frac{5}{4}+\frac{1}{16}\left(15+\sqrt{249}\right),\ -\frac{1}{8}+\frac{1}{32}\left(15+\sqrt{249}\right),\ 1\right\},\right.$$
$$\left.\left\{-\frac{5}{4}+\frac{1}{16}\left(15-\sqrt{249}\right),\ -\frac{1}{8}+\frac{1}{32}\left(15-\sqrt{249}\right),\ 1\right\},\ \{1,\ -2,\ 1\}\right\}$$

Esto da una aproximación numérica de la matriz.

```
rn = N[r]
{{3., 4., 5.}, {4., 5., 6.}, {5., 6., 7.}}
```

He aquí las aproximaciones numéricas para los valores propios.

```
Eigenvalues[rn]
```

$$\{15.3899,\ -0.389867,\ -2.43881\times 10^{-16}\}$$

9.4. Elegir elementos de listas

`First[`*list*`]`	el primer elemento de la lista
`Last[`*list*`]`	el último elemento
`Part[`*list*`, `*n*`]` o *list*`[[`*n*`]]`	el *n*-ésimo elemento
`Part[`*list*`, –`*n*`]` o *list*`[[–`*n*`]]`	el *n*-ésimo elemento empezando del final
`Part[`*list*`, {`n_1`, `n_2`, ...}]` o *list*`[[{`n_1`, `n_2`, ...}]]`	la lista de elementos de las posiciones n_1, n_2, ...

Selección de elementos de listas.

Usaremos esta lista para los ejemplos.

```
t = {a,b,c,d,e,f,g}
{a, b, c, d, e, f, g}
```

He aquí el último elemento de `t`.

```
Last[t]
g
```

Esto da el tercer elemento.

```
t[[3]]
c
```

Esto da una lista del primero y cuarto elementos.

```
t[[ {1, 4} ]]
{a, d}
```

`Take[`*list*`, `*n*`]`	los *n* primeros elementos de *list*
`Take[`*list*`, –`*n*`]`	los *n* últimos elementos
`Take[`*list*`, {`*m*`, `*n*`}]`	elementos *m* hasta *n* (inclusive)
`Rest[`*list*`]`	*list* con su primer elemento quitado
`Drop[`*list*`, `*n*`]`	*list* con sus *n* primeros elemento quitados
`Most[`*list*`]`	*list* con su último elemento quitado
`Drop[`*list*`, –`*n*`]`	*list* con sus *n* últimos elemento quitados
`Drop[`*list*`, {`*m*`, `*n*`}]`	*list* con sus elementos *m* hasta *n* quitados

Selección de secuencias de listas.

Esto da los tres primeros elementos de la lista `t` arriba definida.

```
Take[t, 3]
{a, b, c}
```

Esto da los tres últimos elementos.

```
Take[t, -3]
{e, f, g}
```

Esto da los elementos 2 hasta 5 inclusive.

```
Take[t, {2, 5}]
{b, c, d, e}
```

Esto da los elementos 3 hasta 7 en pasos de 2.

```
Take[t, {3, 7, 2}]
{c, e, g}
```

Esto da `t` con el primer elemento quitado.

```
Rest[t]
{b, c, d, e, f, g}
```

Usando `Drop` usted puede realizar la misma operación que con `Rest`.

```
Drop[t, 1]
{b, c, d, e, f, g}
```

Esto da t con su primeros tres elementos quitados.

```
Drop[t, 3]
{d, e, f, g}
```

Esto da `t` con sólo su tercer elemento quitado.

```
Drop[t, {3, 4}]
{a, b, e, f, g}
```

Las funciones en esta sección le permiten elegir partes ubicadas en posiciones particulares de las listas.

9.5. Prueba y búsqueda de elementos de una lista

`Position[`*list*`, `*form*`]`	posiciones en las cuales *form* ocurre en *list*
`Count[`*list*`, `*form*`]`	número de veces que *form* aparece como un elemento de *list*
`MemberQ[`*list*`, `*form*`]`	prueba si *form* es un elemento de *list*
`FreeQ[`*list*`, `*form*`]`	prueba si *form* no ocurre en ninguna parte *list*

Prueba y búsqueda de elementos de una lista.

La sección anterior mostró como extraer partes de listas basándose en sus posiciones o índices. *Mathematica* también tiene funciones que buscan y prueban elementos de listas, basándose en los valores de aquellos elementos.

Esto da una lista de las posiciones en las cuales `a` aparece en la lista.

```
Position[{a, b, c, a, b}, a]
{{1}, {4}}
```

`Count` cuenta el número de ocurrencias de `a`.

```
Count[{a, b, c, a, b}, a]
2
```

Esto muestra que `a` es un elemento de `{a,b,c}`.

```
MemberQ[{a, b, c}, a]
True
```

Por otra parte `d`, no lo es.

```
MemberQ[{a, b, c}, d]
False
```

Esto asigna a `m` una matriz identidad 3×3.

```
m = IdentityMatrix[3]
{{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}
```

Esto muestra que `0` ocurre en *alguna parte* de `m`.

```
FreeQ[m, 0]
False
```

Esto da una lista de las posiciones en las cuales ocurre `0` en `m`.

```
Position[m, 0]
{{1, 2}, {1, 3}, {2, 1}, {2, 3}, {3, 1}, {3, 2}}
```

Esto da una lista de las posiciones en las cuales ocurre `1` en `m`.

```
Position[m, 1]
{{1, 1}, {2, 2}, {3, 3}}
```

9.6. Agregar, quitar y modificar elementos de una lista

`Prepend[`*list*`,` *element*`]`	agrega *element* al inicio de *list*
`Append[`*list*`,` *element*`]`	agrega *element* al final de *list*
`Insert[`*list*`,` *element*`,` *i*`]`	inserta *element* en la posición *i* de *list*
`Insert[`*list*`,` *element*`, –`*i*`]`	inserta en la posición *i* contando desde el final de *list*
`Delete[`*list*`,` *i*`]`	borra el elemento de la posición *i* de *list*
`ReplacePart[`*list*`,` *new*`,` *i*`]`	reemplaza el elemento de la posición *i* de *list* con *new*
`ReplacePart[`*list*`,` *new*`, {`*i*`,` *j*`}]`	reemplaza *list*`[[`*i*`,` *j*`]]` con *new*

Funciones para manipular elementos en listas explícitas.

Esto da una lista con `x` insertado.

```
Prepend[{a, b, c}, x]
{x, a, b, c}
```

Esto inserta `x` en la posición 2.

```
Insert[{a, b, c}, x, 2]
{a, x, b, c}
```

Esto reemplaza el tercer elemento en la lista con `x`.

```
ReplacePart[{a, b, c, d}, x, 3]
{a, b, x, d}
```

Esto reemplaza el elemento 1, 2 en una matriz 2×2.

```
ReplacePart[{{a, b}, {c, d}}, x, {1, 2}]
{{a, x}, {c, d}}
```

Las funciones como `ReplacePart` toman listas explícitas y dan nuevas listas. A veces, sin embargo, puede ser que desee modificar una lista, sin generar explícitamente una nueva lista.

v `= {` e_1`,` e_2`, ...}`	asigna una lista a una variable
v`[[`*i*`]]` `=` *new*	asigna un nuevo valor para el *i*-ésimo elemento

Reseteo de los elementos de una lista.

Esto define `v` como una lista.

```
v = {a, b, c, d}
```
```
{a, b, c, d}
```

Esto indica que el tercer elemento debe ser `x`.

```
v[[3]] = x
```
```
x
```

Ahora `v` ha sido cambiado.

```
v
```
```
{a, b, x, d}
```

m`[[`*i*`,`*j*`]]` = *new*	reemplaza el elemento *i*, *j* de una matriz
m`[[`*i*`]]` = *new*	reempaza la *i*-ésima fila
m`[[All,`*j*`]]` = *new*	reempaza la *j*-ésima columna

Reseteo de elementos de matrices.

Esto define `m` como una matriz.

```
m = {{a, b}, {c, d}}
```
```
{{a, b}, {c, d}}
```

Esto reemplaza los elementos de la primera columna de la matriz.

```
m[[All, 1]] = {x, y}; m
```
```
{{x, b}, {y, d}}
```

Esto hace que los elementos de la primera columna sean `0`.

```
m[[All, 1]] = 0; m
```
```
{{0, b}, {0, d}}
```

9.7. Combinación de listas

`Join[`*list*$_1$`,` *list*$_2$`, ...]`	concatena listas
`Union[`*list*$_1$`,` *list*$_2$`, ...]`	combina lista quitando los elementos repetidos y ordenando el resultado

Funciones para combinar listas.

`Join` concatena cualquier cantidad de listas.

```
Join[{a, b, c}, {x, y}, {t, u}]
{a, b, c, x, y, t, u}
```

`Union` combina listas, manteniendo sólo los elementos distintos.

```
Union[{a, b, c}, {c, a, d}, {a, d}]
{a, b, c, d}
```

9.8. Listas y conjuntos

Mathematica por lo general mantiene los elementos de una lista en el orden en que se ingresaron. Si quiere tratar una lista de *Mathematica* como un conjunto matemático, sin embargo, puede que desee ignorar el orden de los elementos en la lista.

`Union[`$list_1$`, `$list_2$`, ...]`	da una lista de los distintos elementos en las $list_i$
`Intersection[`$list_1$`, `$list_2$`, ...]`	da una lista de los elementos que son comunes a todas las $list_i$
`Complement[`*universo*`, `$list_1$`, ...]`	da una lista de los elementos que están en el *universo*, pero no en cualquiera de las $list_i$
`Subsets[`*list*`]`	da el conjunto potencia de *list*

Funciones de la teoría de conjuntos.

`Union` da los elementos que aparecen en *cualquiera* de las listas.

```
Union[{c, a, b}, {d, a, c}, {a, e}]
{a, b, c, d, e}
```

`Intersection` da sólo elementos que aparecen n en *todas* las listas.

```
Intersection[{a, c, b}, {b, a, d, a}]
{a, b}
```

`Complement` da elementos que aparecen en la primera lista, pero no en cualquiera de las otras.

```
Complement[{a, b, c, d}, {a, d}]
{b, c}
```

Esto da el conjunto potencia de la lista.

```
Subsets[{a, b, c}]
{{}, {a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c}}
```

9.9. Reordenamiento de listas

`Sort[`*list*`]`	clasifica los elementos de *list* en orden
`Union[`*list*`]`	da una lista de los elementos que son comunes a todas las $list_i$
`Reverse[`*list*`]`	invierte el orden de los elementos de *list*
`RotateLeft[`*list*`]`	rota los elementos de *list n* lugares a la izquierda
`RotateRight[`*list*`]`	rota *n* lugares a la derecha

Funciones para reordenar listas.

Esto ordena los elementos de una lista en forma estándar. En casos simples como éste el ordenamiento es alfabético o numérico.

```
Sort[{b, a, c, a, b}]
{a, a, b, b, c}
```

Esto ordenada los elementos, quitando los que están duplicados.

```
Union[{b, a, c, a, b}]
{a, b, c}
```

Esto rota los elementos de la lista dos lugares a la izquierda.

```
RotateLeft[{a, b, c, d, e}, 2]
{c, d, e, a, b}
```

Puede rotar a la derecha dando un valor negativo, o usando `RotateRight`.

```
RotateLeft[{a, b, c, d, e}, -2]
{d, e, a, b, c}
```

`PadLeft[`*list*`, `*len*`, `*x*`]`	completa *list* en la izquierda con *x* hasta lograr la longitud *len*
`PadRight[`*list*`, `*len*`, `*x*`]`	completa *list* en la derecha

Completar listas.

Esto completa una lista con varias x hasta lograr que tenga longitud 10.

```
PadLeft[{a, b, c}, 10, x]
{x, x, x, x, x, x, x, a, b, c}
```

9.10. Agrupación y combinación de elementos de listas

`Partition[`*list*`, `*n*`]`	particiona listas en piezas de *n* elementos
`Partition[`*list*`, `*n*`, `*d*`]`	use *d* para piezas sucesivas
`Split[`*list*`]`	da el conjunto potencia de *list*

Funciones para agrupar elementos de listas.

He aquí una lista.

```
t = {a, b, c, d, e, f, g}
{a, b, c, d, e, f, g}
```

Esto agrupa los elementos en pares, deshaciéndose del último elemento.

```
Partition[t, 2]
{{a, b}, {c, d}, {e, f}}
```

Esto agrupa elementos en triadas. No hay traslapamiento entre las triadas.

```
Partition[t, 3]
{{a, b, c}, {d, e, f}}
```

Esto hace triadas de elementos, con cada triada sucesiva compensada por sólo un elemento.

```
Partition[t, 3, 1]
{{a, b, c}, {b, c, d}, {c, d, e}, {d, e, f}, {e, f, g}}
```

Esto divide la lista en piezas de elementos idénticos.

```
Split[{a, a, b, b, b, a, a, a, b}]
{{a, a}, {b, b, b}, {a, a, a}, {b}}
```

`Tuples[`*list*`, `*n*`]`	genera todas las *n*-uplas posibles de elementos de *list*
`Tuples[{`$list_1$`, `$list_2$`, ...}]`	genera el producto cartesiano $list_1 \times list_2 \times ...$

Encuentra las uplas posibles de elementos en listas.

Esto da todas las formas posibles de escoger dos elementos de la lista.

```
Tuples[{a, b}, 2]
{{a, a}, {a, b}, {b, a}, {b, b}}
```

Esto da todas las formas posibles de escoger un elemento de cada lista.

```
Tuples[{{a, b}, {1, 2, 3}}]
{{a, 1}, {a, 2}, {a, 3}, {b, 1}, {b, 2}, {b, 3}}
```

9.11. Ordenamiento de listas

`Sort[`*list*`]`	clasifica los elementos de *list* en orden
`Min[`*list*`]`	el menor elemento en *list*
`Ordering[`*list*`, `*n*`]`	posiciones de *n* menores elementos en *list*
`Max[`*list*`]`	el mayor elemento en *list*
`Ordering[`*list*`, –`*n*`]`	posiciones de *n* mayores elementos en *list*
`Ordering[`*list*`]`	el orden de todos los elementos en *list*
`Permutations[`*list*`]`	todos los posibles ordenamientos de *list*

Ordenamiento de listas.

He aquí una lista.

```
t = {17, 21, 14, 9, 18}
{17, 21, 14, 9, 18}
```

Esto da el menor elemento de la lista.

```
Min[t]
9
```

Esto da las posiciones de los tres menores elementos.

```
Ordering[t, 3]
{4, 3, 1}
```

He aquí los elementos.

```
t[[%]]
{9, 14, 17}
```

9.12. Reorganización de listas anidadas

`Flatten[`*list*`]`	elimina todos los niveles en *list*
`Flatten[`*list*`, `*n*`]`	elimina los *n* más altos niveles en *list*
`Partition[`*list*`, {`n_1`, `n_2`, ...}]`	particiona en bloques de tamaño $n_1 \times n_2 \times ...$
`Transpose[`*list*`]`	intercambia los dos niveles superiores de *list*
`RotateLeft[`*list*`, {`n_1`, `n_2`, ...}]`	rota niveles sucesivos *n* lugares
`PadLeft[`*list*`, {`n_1`, `n_2`, ...}]`	completa niveles sucesivos hasta lograr la longitud n_i

Algunas funciones para reorganizar listas anidadas.

Esto "elimina" las sublistas. Usted tomarlo como tan solo un comando eficaz que quita todos las llaves interiores.

```
Flatten[{{a}, {b, {c}}, {d}}]
{a, b, c, d}
```

Esto elimina sólo un nivel de sublista.

```
Flatten[{{a}, {b, {c}}, {d}}, 1]
{a, b, {c}, d}
```

Hay muchas otras operaciones que usted puede realizar en listas anidadas.

10. Gráficos y Sonido

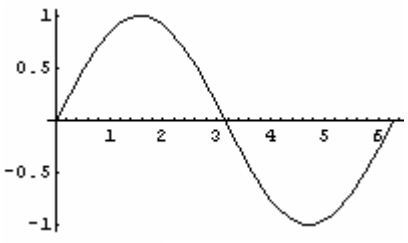
10.1. Gráficos básicos

`Plot[`*f*`, {`*x*`, `x_{min}`, `x_{max}`}]`	genera todas las *n*-uplas posibles de elementos de *list*
`Plot[{`f_1`, `f_2`, ...}, {`*x*`, `x_{min}`, `x_{max}`}]`	genera el producto cartesiano $list_1 \times list_2 \times ...$

Trazado básico de funciones.

Esto traza un gráfico de $\sin(x)$ como una función de x desde 0 hasta 2π.

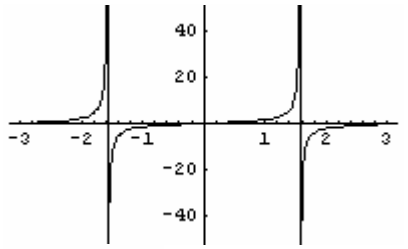
```
Plot[Sin[x], {x, 0, 2Pi}]
```



- Graphics -

Puede trazar funciones que tienen singularidades. *Mathematica* tratará de encontrar escalas apropiadas.

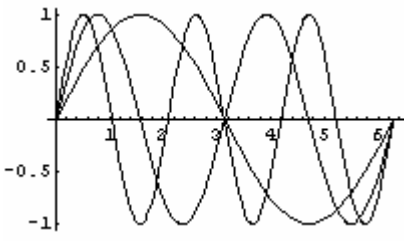
```
Plot[Tan[x], {x, -3, 3}]
```



- Graphics -

Puede dar una lista de funciones para que sean trazadas.

```
Plot[{Sin[x], Sin[2x], Sin[3x]}, {x, 0, 2Pi}]
```



- Graphics -

Para obtener curvas suaves, *Mathematica* tiene que evaluar las funciones que usted desea trazar en un gran número de puntos. Por consiguiente, es importante que establezca las cosas de modo que cada evaluación de las funciones sea tan rápida como se pueda.

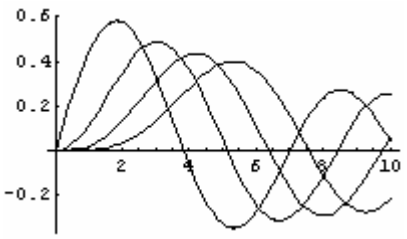
Cuando pide a *Mathematica* que trace un objeto, digamos *f*, como una función de *x, Mathematica* puede hacerlo de dos formas. La primera es tratar a *f* como una expresión simbólica en términos de *x*, y luego evaluar esta expresión numéricamente para los valores específicos de *x* necesarios en el trazado. La segunda es calcular primero que valores de *x* son necesarios, y sólo posteriormente evaluar *f* con aquellos valores de *x*.

Si digita `Plot[`*f*`, {`*x*`, `x_{min}`, `x_{max}`}]` es usada la segunda alternativa. Esto tiene la ventaja que *Mathematica* sólo trata de evaluar *f* para valores numéricos específicos de *x*; sin importar si *f* define valores sensibles cuando *x* es simbólico.

Hay, sin embargo, algunos casos en los cuales es mucho mejor tener *f* evaluada antes de empezar a hacer el trazado. Un caso típico es cuando *f* es en realidad un comando que genera una tabla de funciones. Usted puede hacer que *Mathematica* primero produzca la tabla, y en seguida evalúe las funciones, antes que producir la tabla de nuevo para cada valor de *x*. Puede hacer esto digitando `Plot[Evaluate[`*f*`], {`*x*`, `x_{min}`, `x_{max}`}]`.

Esto hace un trazado de las funciones Bessel $J_n(x)$ con *n* variando desde 1 hasta 4. `Evaluate` le dice a *Mathematica* que primero haga la tabla de funciones, y sólo entonces evalúe para valores particulares de `x`.

```
Plot[Evaluate[Table[BesselJ[n, x], {n, 4}]],
{x, 0, 10}]
```



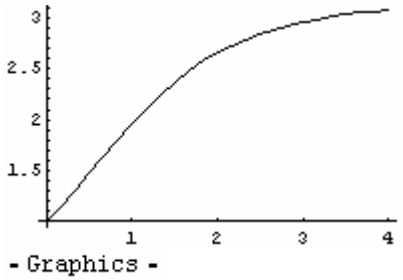

```
- Graphics -
```

Esto encuentra la solución numérica de una ecuación diferencial, como se mostró en la sección 7.4.

```
NDSolve[{y'[x] == Sin[y[x]], y[0] == 1}, y,
{x, 0, 4}]
{{y → InterpolatingFunction[{{0., 4.}}, <>]}}
```

He aquí un trazado de la solución. `Evaluate` le dice a *Mathematica* que primero obtenga un objeto con InterpolatingFunction, y luego lo evalúe en una secuencia de valores de `x`.

```
Plot[Evaluate[ y[x] /. % ], {x, 0, 4}]
```



- Graphics -

`Plot[`*f*`, {`*x*`, `x_{min}`, `x_{max}`}]`	primero obtiene valores numéricos para *x*, luego evalúa *f* para cada valor de *x*
`Plot[Evaluate[`*f*`], {`*x*`, `x_{min}`, `x_{max}`}]`	primero evalúa *f*, luego obtiene valores numéricos específicos de *f*
`Plot[Evaluate[Table[`*f*`,...]],{`*x*`, `x_{min}`, `x_{max}`}]`	genera una lista de funciones, y las traza
`Plot[Evaluate[`*y*`[`*x*`] /. `*solution*`]],{`*x*`, `x_{min}`, `x_{max}`}]`	traza una solución numérica para una ecuación diferencial obtenida con `NDSolve`

Métodos para establecer objetos a trazar.

10.2. Opciones

Cuando *Mathematica* traza un gráfico para usted, tiene que escoger muchas opciones. Tiene que resolverse lo que deben ser las escalas, donde se debe mostrar la función, cómo deben dibujarse los ejes, etcétera. La mayoría de las veces, *Mathematica* probablemente tomará opciones bastante buenas. Sin embargo, si usted quiere obtener los mejores dibujos posibles para sus objetivos particulares, debería ayudar a *Mathematica* en la elección de algunas de sus opciones.

Hay un mecanismo general para especificar “opciones” en las funciones de *Mathematica*. Cada opción tiene un nombre definido. Como últimos argumentos de una función como `Plot`, usted puede incluir una secuencia de reglas de la forma *nombre* `->` *valor*, para especificar los valores de varias opciones. A cualquier opción para la cual usted no da una regla explícita se le asigna su valor por “defecto”.

`Plot[`*f*`, {`*x*`,` x_{min}`,` x_{max}`},` *option* `->` *value*`]`	hace un trazado, especificando un valor particular para una opción

Elección de una opción para el trazado de un gráfico.

Una función como `Plot` tiene muchas opciones que usted puede ajustar. Por lo general tendrá que usar varias de ellas a la vez.

Si usted quiere optimizar un trazado particular, probablemente deberá experimentar, intentando una secuencia de diferentes ajustes para varias opciones.

Cada vez que usted produce un gráfico, puede especificar opciones para él. La sección 10.3 mencionará como usted puede cambiar algunas opciones, aún después de que ha producido el gráfico.

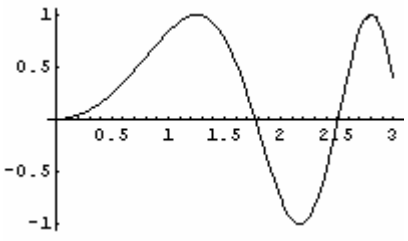
nombre de la opción	*valor por defecto*	
`AspectRatio`	`1/GoldeRatio`	proporción alto-ancho de el gráfico; `Automatic` se usa para escalas iguales de *x* e *y*
`Axes`	`Automatic`	si se desea incluir ejes
`AxesLabel`	`None`	etiquetas para colocarles a los ejes; `{`*xlabel*`, None}` especifica una etiqueta para el eje *x*; `{`*xlabel*`,` *ylabel*`}` especifica una etiqueta para ambos ejes
`AxesOrigin`	`Automatic`	el punto en el que se interceptan los ejes
`TextStyle`	`$TextStyle`	el estilo por defecto a usar para el texto del gráfico
`FormatType`	`StandardForm`	el tipo de formato por defecto a usar para el texto en el gráfico
`DisplayFunction`	`$DisplayFunction`	para mostrar los gráficos; `Identity` no los muestra
`Frame`	`False`	si desea dibujar un marco alrededor del gráfico

`FrameLabel`	`None`	etiquetas alrededor del marco; dé una lista ordenada en sentido horario empezando con el valor inferior del eje *x*
`FrameTicks`	`Automatic`	qué marcas dibujar si hay un marco; `None` no muestra ninguna marca
`GridLines`	`None`	qué líneas de cuadrícula incluir; `Automatic` incluye una línea para cada marca importante
`PlotLabel`	`None`	una expresión que se imprimirá como etiqueta para el gráfico
`PlotRange`	`Automatic`	el rango de las coordenadas a incluir en el gráfico; `All` incluye todos los puntos
`Ticks`	`Automatic`	indica qué marcas dibujar si hay ejes; None no muestra ninguna marca

Algunas de las opciones de `Plot`. Estas opciones también pueden usarse en `Show`.

He aquí un gráfico con todas las opciones que tienen sus valores por defecto.

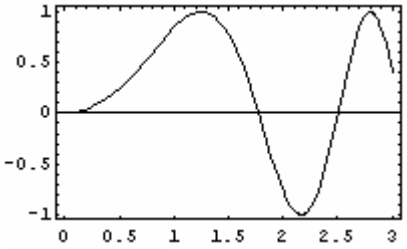
```
Plot[Sin[x^2], {x, 0, 3}]
```




```
- Graphics -
```

Esto dibuja ejes sobre un marco alrededor del gráfico.

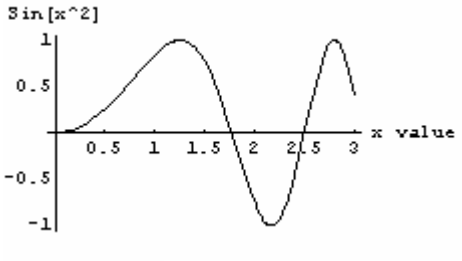
```
Plot[Sin[x^2], {x, 0, 3}, Frame->True]
```



- Graphics -

Esto especifica las etiquetas para los ejes *x* e *y*. Las expresiones que usted da como etiquetas son impresas como salidas de *Mathematica*. Puede dar cualquier porción de texto poniéndolo entre comillas.

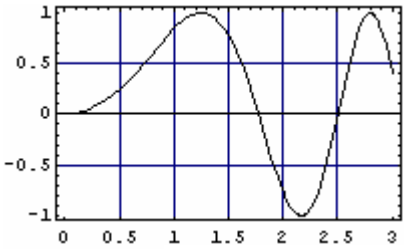
```
Plot[Sin[x^2], {x, 0, 3},
AxesLabel -> {"x value", "Sin[x^2]"} ]
```



- Graphics -

Usted puede dar varias opciones al mismo tiempo, en cualquier orden.

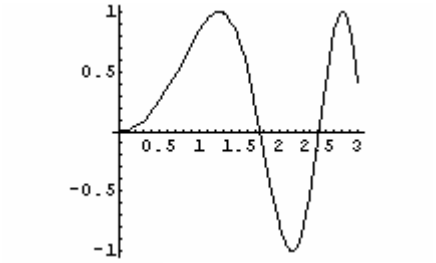
```
Plot[Sin[x^2], {x, 0, 3},
Frame -> True, GridLines -> Automatic]
```



- Graphics -

Ajustar la opción `AspectRatio` cambia la forma entera de su gráfico. `AspectRatio` da el cociente del ancho y alto. Su valor por defecto es el inverso de la proporción áurea–supuestamente la forma más agradable para un rectángulo.

```
Plot[Sin[x^2], {x, 0, 3}, AspectRatio -> 1]
```



- Graphics -

`Automatic`	usa algoritmos internos
`None`	no incluye esto
`All`	incluye todo
`True`	hace esto
`False`	no hace esto

Algunos ajustes comunes para varias opciones.

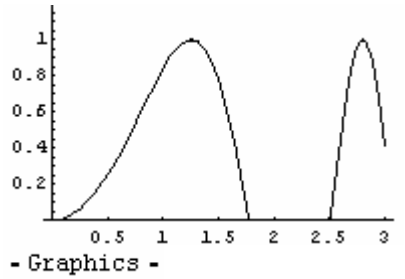
Cuando *Mathematica* traza un gráfico, intenta ajustar las escalas de *x* e *y* para incluir sólo las partes "interesantes" del gráfico. Si su función aumenta muy rápidamente, o tiene singularidades, las partes donde se torna demasiado grande serán cortadas. Especificando la opción `PlotRange`, usted puede controlar exactamente qué rangos de las coordenadas *x* e *y* se incluyen en su gráfico.

`Automatic`	muestra por lo menos una fracción grande de los puntos, incluyendo la región "interesante" (ajuste por defecto)
`All`	muestra todos los puntos
{y_{min}, y_{max}}	muestra un rango específico de los valores de *y*
{*xrange*, *yrange*}	muestra los rangos especificados de los valores de *x* e *y*

Ajustes para la opción `PlotRange`.

Ajustar la opción `PlotRange` da los límites explícitos de *y* para el gráfico. Con los límites especificados aquí, el fondo de la curva es cortado.

```
Plot[Sin[x^2], {x, 0, 3},
PlotRange -> {0, 1.2}]
```



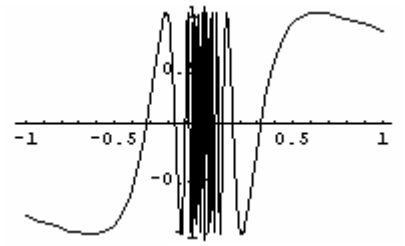
- Graphics -

Mathematica siempre trata de trazar funciones como curvas suaves. Por consiguiente, en lugares donde su función oscila mucho, *Mathematica* usará más puntos.

En general, *Mathematica* intenta *adaptar* el muestreo de su función a la forma de la función. Hay, sin embargo, un límite, que usted puede ajustar, para cómo finalmente *Mathematica* muestreará una función.

La función $\sin\left(\frac{1}{x}\right)$ oscila infinitamente a menudo cuando $x \to 0$. Mathematica intenta muestrear más puntos en la región donde la función oscila mucho, pero nunca puede muestrear el número infinito que usted necesitaría para reproducir la función exactamente. Por consiguiente, hay leves interferencias en el diagrama.

```
Plot[Sin[1/x], {x, -1, 1}]
```



- Graphics -

nombre de la opción	*valor por defecto*	
`PlotStyle`	`Automatic`	una lista de listas de directivas para gráficos a utilizar para cada curva
`PlotPoints`	`25`	el mínimo número de puntos para muestrear la función
`MaxBend`	`10.`	el ángulo máximo de flexión entre segmentos sucesivos de una curva
`PlotDivision`	`30.`	el factor máximo de subdivisión del muestreo de la función
`Compiled`	`True`	si desea compilar la función que es trazada

Más opciones para `Plot`. Estas opciones no pueden usarse en `Show`.

Es importante comprender que puesto que *Mathematica* puede muestrear su función solamente en un número limitado de puntos, siempre pueden faltar características de la función. Aumentando `PlotPoints`, usted puede hacer que *Mathematica* muestree su función en un mayor número de puntos. Por supuesto, cuanto mayor sea `PlotPoints`, más demora *Mathematica* en trazar cualquier función, incluso si ésta es suave.

Ya que `Plot` tiene que evaluar su función muchas veces, es importante hacer cada evaluación tan rápida como sea posible. Por consiguiente, *Mathematica* por lo general *compila* su función en un pseudocódigo de bajo nivel que puede ser ejecutado de manera muy eficiente. Un problema potencial con esto, sin embargo, es que el pseudocódigo permite sólo operaciones numéricas acordes a la precisión de máquina. Si la función que usted traza requiere operaciones de precisión más alta, debería apagar la compilación en `Plot`. Usted puede hacer esto poniendo la opción `Compile -> False`.

Note que *Mathematica* sólo puede compilar el “código en línea”; no puede, por ejemplo, compilar funciones que usted ha definido. Por consiguiente, cuando posible, debería usar `Evaluate` como se describe en la sección 10.1 para evaluar cualquier definición y conseguir una forma tal que el compilador de *Mathematica* puede manejar.

10.3. Rehacer y combinar gráficos

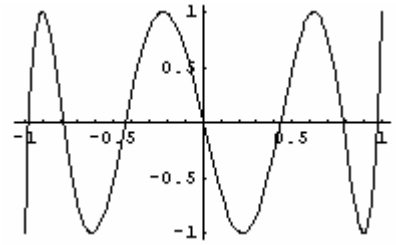
Mathematica guarda la información sobre cada gráfico que usted produce, de modo que usted más tarde pueda rehacerlo. Cuando usted rehace gráficos, puede cambiar algunas opciones que ha usado.

`Show[`*plot*`]`	rehace un gráfico
`Show[`*plot*`, `*option*` -> `*value*`]`	rehace con opciones cambiadas
`Show[`$plot_1$`, `$plot_2$`, ...]`	combina varios gráficos
`Show[GraphicsArray[` `{{`$plot_1$`, `$plot_2$`, ...} , ...}]]`	dibuja un array de gráficos
`InputForm[`*plot*`]`	muestra información guardada de un gráfico

Funciones para manipular gráficos.

He aquí un gráfico simple. `- Graphics -` por lo general es impresa en la línea de salida para apoyar la información que guarda *Mathematica* sobre el gráfico.

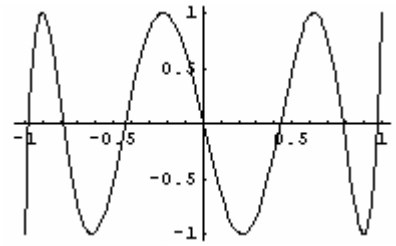
```
Plot[ChebyshevT[7, x], {x, -1, 1}]
```




```
- Graphics -
```

Esto rehace el gráfico anterior.

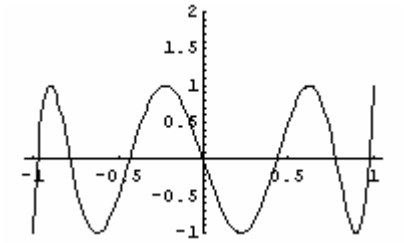
```
Show[%]
```




```
- Graphics -
```

Cuando usted redibuja el gráfico, puede cambiar algunas de las opciones. Esto cambia la opción del rango de *y*.

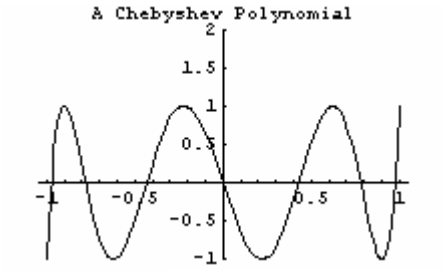
```
Show[%, PlotRange -> {-1, 2}]
```



- Graphics -

Esto toma el gráfico anterior, y cambia otra opción en él.

```
Show[%, PlotLabel -> "A Chebyshev Polynomial"]
```



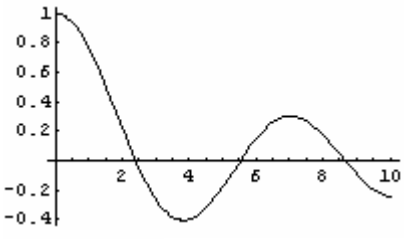
- Graphics -

Usando Show con una secuencia de opciones diferentes, usted puede ver el mismo gráfico de varias formas diferentes. Puede desear hacer esto, por ejemplo, si trata de encontrar el mejor ajuste posible de opciones.

También puede usar Show para combinar gráficos. No importa si los gráficos no tienen la misma escala: *Mathematica* siempre escogerá nuevas escalas para incluir los puntos que usted quiere.

Esto fija gj0 para que sea un gráfico de $J_0(x)$ desde $x = 0$ hasta 10.

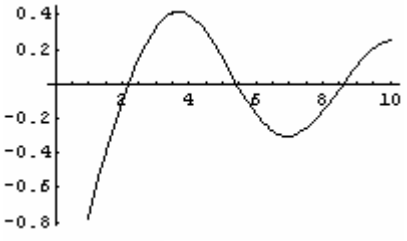
```
gj0 = Plot[BesselJ[0, x], {x, 0, 10}]
```



- Graphics -

He aquí el gráfico de $Y_1(x)$ desde $x = 1$ hasta 10.

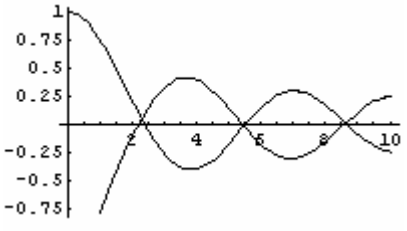
```
gy1 = Plot[BesselY[1, x], {x, 1, 10}]
```



- Graphics -

Esto muestra los dos gráficos anteriores combinadas en uno. Note que la escala es ajustada apropiadamente

```
gjy = Show[gj0, gy1]
```



- Graphics -

Usando `Show[`*plot*$_1$`,` *plot*$_2$`, …]` usted puede combinar varios gráficos en uno. `GraphicsArray` le permite dibujar varios gráficos en un array.

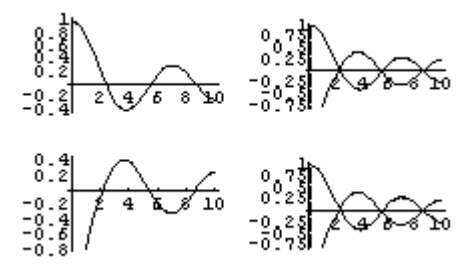
`Show[GraphicsArray[` `{`*plot*$_1$`,` *plot*$_2$`, ...}]]`	dibuja una fila de gráficos
`Show[GraphicsArray[` `{{`*plot*$_1$`}, {`*plot*$_2$`}, ...}]]`	dibuja una fila de gráficos

`Show[GraphicsArray[` {{$plot_{11}$, $plot_{12}$, ...} , ...}]]	dibuja un array rectangular de gráficos
`Show[GraphicsArray[` *plots*`, GraphicsSpacing ->` {h, v}]]	especifica el espaciado horizontal y vertical entre los gráficos

Dibujo de arrays de gráficos.

Esto muestra los últimos gráficos anteriores en un array.

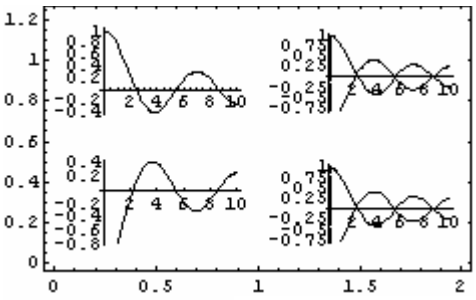
```
Show[GraphicsArray[{{gj0, gjy}, {gy1, gjy}}]]
```



```
- GraphicsArray -
```

Si muestra de nuevo un array de gráficos con `Show`, cualquier opción que especifica se usará para todo el array, antes que para gráficos individuales.

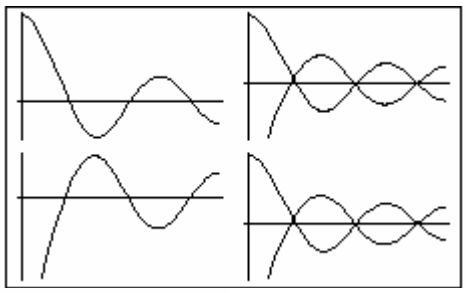
```
Show[%, Frame->True, FrameTicks->Automatic]
```



```
- GraphicsArray -
```

He aquí una forma de cambiar opciones para todos los gráficos del array.

```
Show[ % /. (Ticks -> Automatic) ->
(Ticks -> None) ]
```

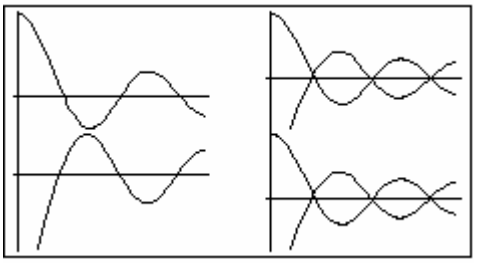


- GraphicsArray -

`GraphicsArray` por defecto pone un borde estrecho alrededor de cada uno de los gráficos en el array que produce. Usted puede cambiar este el tamaño de este borde poniendo la opción `GraphicsSpacing -> {`*h*, *v*`}`. Los parámetros *h* y *v* dan los espaciados horizontales y verticales a usarse.

Esto incrementa el espaciado horizontal, pero disminuye el espaciado vertical entre los gráficos en el array.

```
Show[%, GraphicsSpacing -> {0.3, 0}]
```

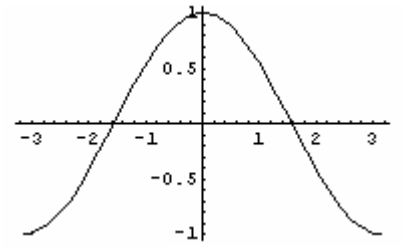


- GraphicsArray -

Cuando usted traza un gráfico, *Mathematica* guarda la lista de puntos que este usó, junto con alguna otra información. Usando lo guardado, usted puede rehacer gráficos de muchas formas diferentes con `Show`. Sin embargo, usted debería comprender que no importa que opciones especifique, `Show` todavía tiene el mismo conjunto básico de puntos para trabajar. Por ejemplo, si usa opciones de modo que *Mathematica* muestre una pequeña parte de su gráfico original ampliado, usted probablemente será capaz de ver los puntos individuales de la muestra que `Plot` ha usado. Las opciones como `PlotPoints` sólo pueden ajustarse en el comando `Plot` original.

He aquí un gráfico simple.

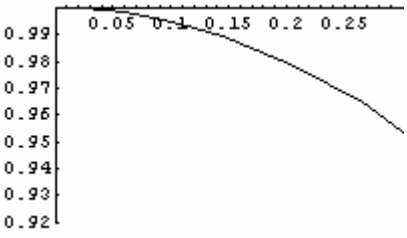
```
Plot[Cos[x], {x, -Pi, Pi}]
```




```
- Graphics -
```

Esto muestra una pequeña región del gráfico en una forma ampliada. En esta resolución los segmentos de recta individuales que fueron producidos por le comando `Plot` original.

```
Show[%, PlotRange -> {{0, .3}, {.92, 1}}]
```




```
- Graphics -
```

10.4. Manipulación de opciones

Hay un número de funciones incorporadas en *Mathematica* que, como `Plot`, tienen varias opciones que usted puede ajustar. *Mathematica* proporciona algunos mecanismos generales para manejar tales opciones.

Si usted no da un ajuste específico para una opción de una función como `Plot`, entonces *Mathematica* automáticamente usará un valor por defecto para la opción. La función `Options[`*function*`, `*option*`]` le permite averiguar el valor por defecto para una opción particular. Usted puede resetear el valor por defecto usando `SetOptions[`*function*`, `*option*` -> `*value*`]`. Note que si hace esto, el valor por defecto que usted ha dado quedará hasta que lo cambie explícitamente.

`Options[`*function*`]`	da una lista de los valores por defecto actuales para todas las opciones
`Options[`*function*`, `*option*`]`	da el valor por defecto para una opción particular
`SetOptions[`*function*`, `*option* `->` *value*`, ...]`	resetea el valor por defecto

Manipulación de valores por defecto para opciones.

He aquí el valor por defecto actual para la opción `PlotRange` de `Plot`.

```
Options[Plot, PlotRange]
```

```
{PlotRange → Automatic}
```

Esto resetea el valor por defecto para la opción `PlotRange`. El punto y coma impide que *Mathematica* imprima la salida de la lista bastante larga de opciones de `Plot`.

```
SetOptions[Plot, PlotRange->All] ;
```

Hasta que usted explícitamente lo resetee, el valor por defecto de la opción `PlotRange` ahora será `All`.

```
Options[Plot, PlotRange]
```

```
{PlotRange → All}
```

Esto vuelve a resetear el valor para la opción `PlotRange`.

```
SetOptions[Plot, PlotRange->{{-1, 1}, {-1, 1}}] ;
```

Hasta que usted explícitamente lo resetee, el valor por defecto de la opción `PlotRange` ahora será `{{-1, 1}, {-1, 1}}`.

```
Options[Plot, PlotRange]
```

```
{PlotRange → {{-1, 1}, {-1, 1}}}
```

Los objetos gráficos que obtiene con `Plot` o `Show` almacenan la información de las opciones que usan. Usted puede obtener esta información aplicando la función `Options` a estos objetos gráficos.

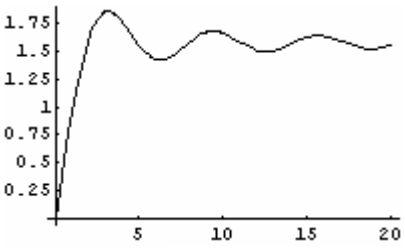
`Options[`*plot*`]`	muestra todas las opciones usadas por el gráfico *plot*
`Options[`*plot*`, `*option*`]`	muestra el ajuste para una opción específica

`AbsoluteOptions[`*plot*, *option*`]`	muestra la forma absoluta usada para una opción específica, incluso si el ajuste para la opción `Automatic` o `All`

Adquisición de información sobre opciones usadas en gráficos.

He aquí un gráfico, con ajustes por defecto para todas las opciones.

```
g = Plot[SinIntegral[x], {x, 0, 20}]
```




```
- Graphics -
```

El ajuste usado para la opción `PlotRange` fue `All`.

```
Options[g, PlotRange]
{PlotRange → All}
```

`AbsoluteOptions` da los valores *absolutos* escogidos automáticamente para `PlotRange`.

```
AbsoluteOptions[g, PlotRange]
{PlotRange →
  {{-0.499999, 20.5}, {-0.0462976, 1.89824}}}
```

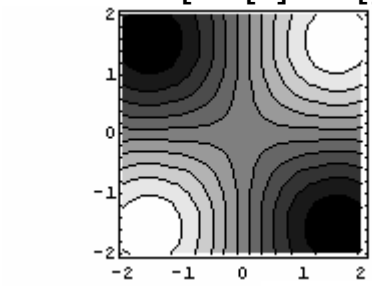
10.5. Contornos y gráficos de densidad

`ContourPlot[`*f*, {*x*, x_{min}, x_{max}}, {*y*, y_{min}, y_{max}}`]`	traza un gráfico de contorno de *f* como una función de *x* e *y*
`DensityPlot[`*f*, {*x*, x_{min}, x_{max}}, {*y*, y_{min}, y_{max}}`]`	traza un gráfico de densidad de *f*

Funciones para manipular gráficos.

Esto da un gráfico de contorno de la función $\sin(x)\sin(y)$.

```
ContourPlot[Sin[x] Sin[y], {x, -2, 2}, {y, -2, 2}]
```



- ContourGraphics -

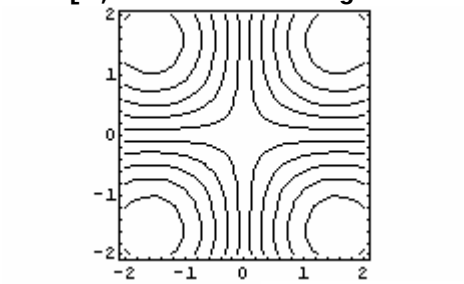
Un gráfico de contorno le da esencialmente un "mapa topográfico" de una función. Los contornos unen puntos sobre la superficie que tienen la misma altura. El valor por defecto permite obtener contornos correspondientes a una secuencia de valores de *z* igualmente espaciados. Los gráficos de contorno producidos por *Mathematica* por defecto son sombreados, de tal modo que las regiones con valores de *z* más altos sean más claras.

nombre de la opción	*valor por defecto*	
`ColorFunction`	`Automatic`	colores a usar para el sombreado; `Hue` usa una secuencia de matices
`Contours`	`10`	el total de contornos, o la lista de valores de *z* para contornos
`PlotRange`	`Automatic`	el rango de valores a ser incluidos, usted puede especificar {z_{min}, z_{max}}, `All` o `Automatic`
`ContourShading`	`True`	si hay que usar sombreado
`PlotPoints`	`30.`	puntos de evaluación en cada dirección
`Compiled`	`True`	si hay que compilar la función que es trazada

Algunas opciones para `ContourPlot`. El primer conjunto también puede usarse en `Show`.

Particularmente si usted utiliza una visualizadora o impresora que no maneja bien los niveles de gris, puede encontrar mejor prescindir del sombreado en gráficos de contorno.

```
Show[%, ContourShading -> False]
```

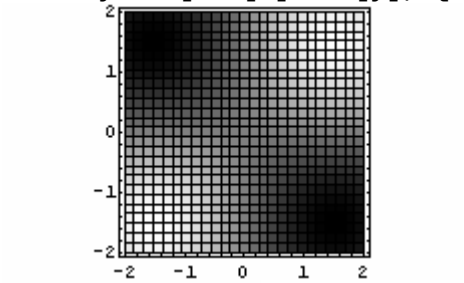


- ContourGraphics -

Usted debe comprender que si no evalúa su función sobre una cuadrícula bastante fina, puede haber inexactitudes en su gráfico de contorno. Un punto a notar es que mientras que una curva generada por `Plot` puede ser inexacta si su función varía demasiado rápido en una región particular, la forma de los contornos puede ser inexacta si su función varía demasiado despacio. Una función que varía rápidamente da un modelo regular de contornos, pero una función que es casi plana puede dar contornos irregulares. Usted puede solucionar tales problemas aumentando el valor de `PlotPoints`.

Los gráficos de densidad muestran los valores de su función como un array regular de puntos. Las regiones más claras son más altas.

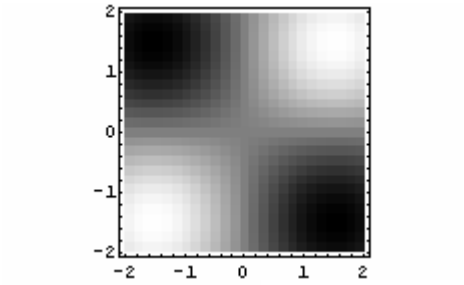
```
DensityPlot[Sin[x] Sin[y], {x, -2, 2}, {y, -2, 2}]
```



- DensityGraphics -

Usted puede deshacerse de la red como aquí.

```
Show[%, Mesh -> False]
```



- DensityGraphics -

nombre de la opción	*valor por defecto*	
`ColorFunction`	`Automatic`	colores para el sombreado
`Mesh`	`True`	si hay que dibujar una red
`PlotPoints`	`25`	puntos de evaluación en cada dirección
`Compiled`	`True`	si desea compilar la función que es trazada

Algunas opciones para `DensityPlot`. El primer conjunto también puede usarse en `Show`.

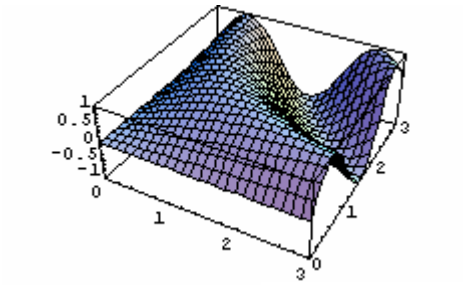
10.6. Gráficos de superficies tridimensionales

`Plot3D[`*f*`, {`*x*`, `x_{min}`, `x_{max}`},` `{`*y*`, `y_{min}`, `y_{max}`}]`	traza un gráfico tridimensional de *f* como una función de las variables *x* e *y*

Trazado básico de una función 3D.

Esto traza un gráfico tridimensional de la función $\sin(x\,y)$.

```
Plot3D[Sin[x y], {x, 0, 3}, {y, 0, 3}]
```



- SurfaceGraphics -

Hay muchas opciones para gráficos tridimensionales en *Mathematica*. Sólo algunos serán discutidos en esta sección.

El primer conjunto de opciones para gráficos tridimensionales es en gran parte análogo a aquellos proveídos en el caso bidimensional.

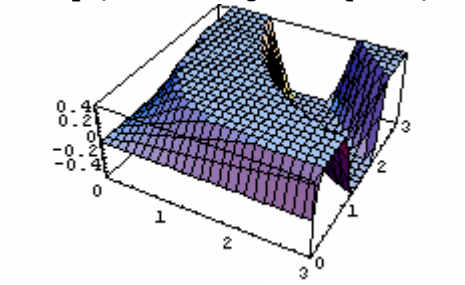
nombre de la opción	*valor por defecto*	
`Axes`	`Automatic`	si hay que incluir ejes
`AxesLabel`	`None`	etiquetas para colocarles a los ejes; {*xlabel*, `None`, `None`} especifica una etiqueta para el eje *x*, {*xlabel*, *ylabel*, *zlabel*} para todos los ejes
`Boxed`	`True`	si hay que dibujar una caja tridimensional alrededor de la superficie
`ColorFunction`	`Automatic`	colores a usar para el sombreado; `Hue` usa una secuencia de matices
`TextStyle`	`$TextStyle`	el estilo por defecto a usar para el texto del gráfico
`FormatType`	`StandardForm`	el tipo de formato por defecto a usar para el texto en el gráfico
`DisplayFunction`	`$DisplayFunction`	para mostrar los gráficos; `Identity` no los muestra
`FaceGrids`	`None`	para dibujar cuadrículas sobre los bordes de las caras de la caja; `All` dibuja una cuadrícula sobre cada cara
`HiddenSurface`	`True`	si se desea dibujar la superficie como un sólido
`Lighting`	`True`	si se desea colorear la superficie usando iluminación simulada
`Mesh`	`True`	si se desea dibujar una red en la superficie

PlotRange	None	el rango de las coordenadas a incluir en el gráfico; usted puede especificar All, {z_{min}, z_{max}} o {{x_{min}, x_{max}}, {y_{min}, y_{max}}, {z_{min}, z_{max}}}
Shading	Automatic	si se desea sombrear la superficie o no
ViewPoint	{1.3, -2.4, 2}	el punto en el espacio desde el cual se desea mirar la superficie
PlotPoints	25	el número de puntos para muestrear la función en cada dirección; {n_x, n_y} especifica diferentes números en las direcciones de x e y
Compiled	True	si desea compilar la función que es trazada

Algunas opciones para Plot3D. Estas opciones también pueden usarse en Show.

Esto vuelve a dibujar el gráfico anterior, con opciones cambiadas. Con este ajuste para PlotRange, sólo es mostrada la parte de la superficie en el rango $-0.5 \le z \le 0.5$.

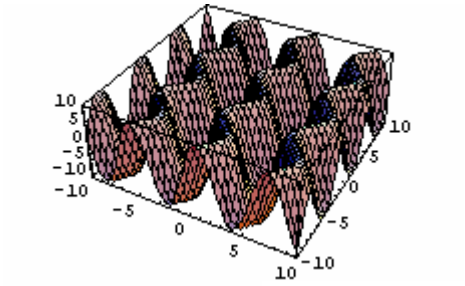
```
Show[%, PlotRange -> {-0.5, 0.5}]
```




```
- SurfaceGraphics -
```

Cuando usted hace el gráfico original, puede elegir muestrear más puntos. Necesitará hacer esto para conseguir buenos cuadros de las funciones que oscilan mucho.

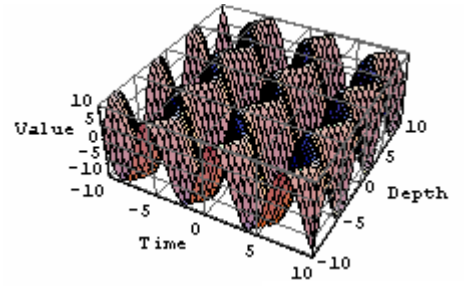
```
Plot3D[10 Sin[x + Sin[y]], {x, -10, 10},
{y, -10, 10}, PlotPoints -> 50]
```



- SurfaceGraphics -

He aquí el mismo gráfico, con etiquetas para los ejes, y cuadrículas añadidas a cada cara.

```
Show[%, AxesLabel -> {"Time", "Depth", "Value"},
FaceGrids -> All]
```

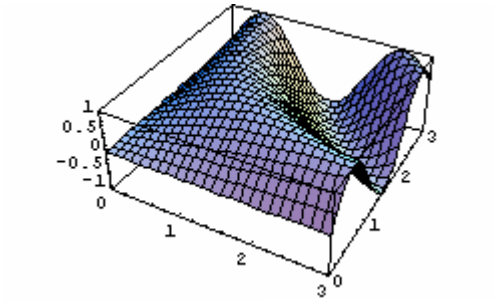


- SurfaceGraphics -

Probablemente la cuestión más importante en el trazado de una superficie tridimensional es especificar desde donde usted quiere mirar la superficie. La opción `ViewPoint` para `Plot3D` y `Show` le permite especificar el punto {*x*, *y*, *z*} del espacio desde el cual usted desea ver una superficie. En muchas versiones de *Mathematica*, hay formas de escoger puntos de vista tridimensionales interactivamente, luego obtener las coordenadas para ajustar la opción `ViewPoint`.

He aquí una superficie, vista desde el punto de vista por defecto `{1.3, -2.4, 2}`. Este punto de vista se elige por ser "genérico", de modo que las alineaciones coincidentes visualmente confusas entre diversas partes de su objeto sean imperceptibles.

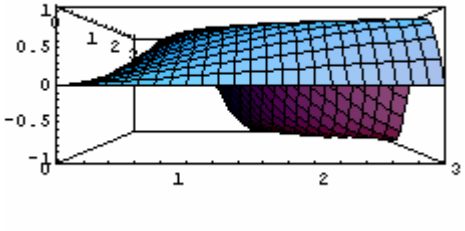
```
Plot3D[Sin[x y], {x, 0, 3}, {y, 0, 3}]
```



- SurfaceGraphics -

Esto vuelve a dibujar el gráfico, con el punto de vista directamente enfrente. Note el efecto de perspectiva que hace que la parte trasera de la caja parezca mucho más pequeña que la del frente.

```
Show[%, ViewPoint -> {0, -2, 0}]
```



- SurfaceGraphics -

`{1.3, -2.4, 2}`	punto de vista por defecto
`{0, -2, 0}`	directamente enfrente
`{0, -2, 2}`	enfrente y encima
`{0, -2, -2}`	enfrente y abajo
`{-2, -2, 0}`	esquina izquierda
`{2, -2, 0}`	esquina derecha
`{0, 0, 2}`	directamente encima

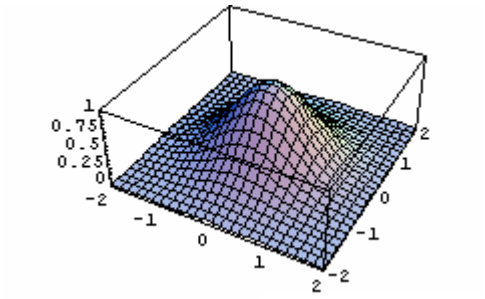
Opciones típicas para la opción `ViewPoint`.

La vista humana no es particularmente buena para entender superficies matemáticas complicadas. Por consiguiente, usted tiene que generar cuadros que contengan tantas pistas como sea posible sobre la forma de la superficie.

Los puntos de vista ligeramente encima de la superficie por lo general trabajan mejor. Es generalmente una buena idea mantener el punto vista suficientemente cerca a la superficie para que haya algún efecto de la perspectiva. Tener una caja dibujada explícitamente alrededor de la superficie es provechoso en el reconocimiento de la orientación de la superficie.

He aquí un gráfico con los ajustes por defecto para las opciones de la superficie representada.

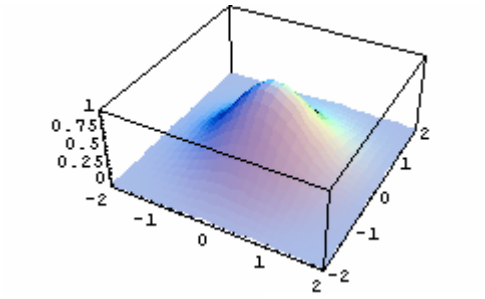
```
g = Plot3D[Exp[-(x^2+y^2)], {x, -2, 2}, {y, -2, 2}]
```



- SurfaceGraphics -

Esto muestra la superficie sin la red dibujada. Es por lo general mucho más difícil de ver la forma de la superficie si la red no está allí.

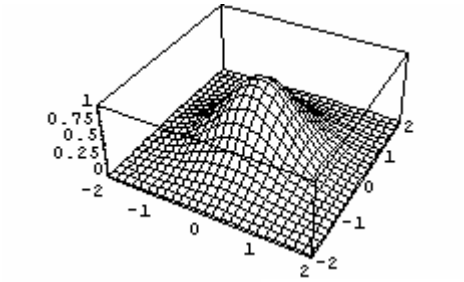
```
Show[g, Mesh -> False]
```



- SurfaceGraphics -

Esto muestra la superficie sin sombrear. Algunos dispositivos de visualización pueden no ser capaces de mostrar sombreado.

```
Show[g, Shading -> False]
```



- SurfaceGraphics -

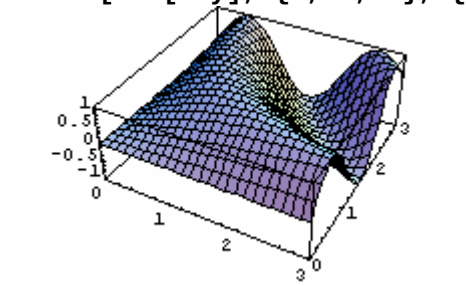
La inclusión de sombreado y red es por lo general de gran ayuda para entender la forma de una superficie. En algunos dispositivos de salida gráfica vectorial, sin embargo, puede que usted no sea capaz de obtener sombreado. También debe comprender que cuando el sombreado es incluido, puede tardar mucho la ejecución la superficie sobre su dispositivo de salida.

Para añadir un elemento suplementario de realismo a la gráfica tridimensional, *Mathematica* por defecto colorea superficies tridimensionales usando un modelo de iluminación simulado. En el caso por defecto, *Mathematica* asume que hay tres fuentes de iluminación que brillan sobre el objeto desde la parte superior derecha.

Mientras en la mayoría de casos, en particular con dispositivos de salida en color, la iluminación simulada es una ventaja, a veces puede ser confusa. Si ajusta la opción `Lighting -> False`, entonces *Mathematica* no usará la iluminación simulada, pero en cambio va a sombrear todas las superficies con niveles de gris determinados por su altura.

`Plot3D` por lo general colorea superficies usando un modelo de iluminación simulado.

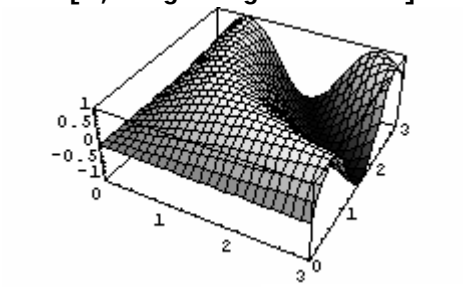
```
Plot3D[Sin[x y], {x, 0, 3}, {y, 0, 3}]
```



- SurfaceGraphics -

`Lighting -> False` apaga la iluminación simulada, y en lugar de esto sombrea superficies con niveles grises determinados por la altura.

```
Show[%, Lighting -> False]
```



```
- SurfaceGraphics -
```

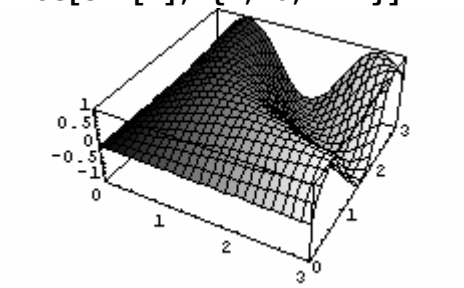
Con `Lighting -> False`, *Mathematica* sombrea superficies según la altura. Usted también puede decir a *Mathematica* explícitamente como sombrear cada elemento de una superficie.

`Plot3D[{f,` `GrayLevel[`*s*`]},` `{`*x*`, `x_{min}`, `x_{max}`},` `{`*x*`, `y_{min}`, `y_{max}`}]`	grafica una superficie correspondiente a *f*, sombreada en gris de acuerdo a la función *s*
`Plot3D[{f, Hue[`*s*`]},` `{`*x*`, `x_{min}`, `x_{max}`},` `{`*x*`, `y_{min}`, `y_{max}`}]`	sombra por variación de matices de color antes que por niveles de gris

Funciones para manipular gráficos.

Esto muestra una superficie cuya altura es determinada por la función `Sin[x y]`, pero cuyo sombreado es determinado por `GrayLevel[x/3]`.

```
Plot[Sin[x], {x, 0, 2Pi}]
```



```
- SurfaceGraphics -
```

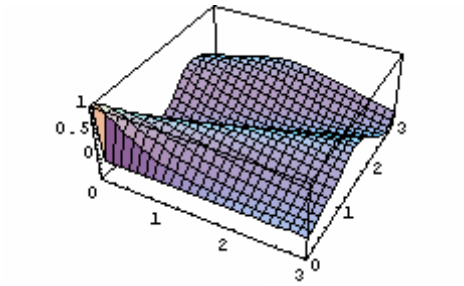
10.7. Conversión entre tipos de gráficos

Contornos, gráficos de densidad y gráficos de superficies son tres formas diferentes de mostrar esencialmente la misma información sobre una función. En todos los casos, usted necesita los valores de una función en una cuadrícula de puntos.

Las funciones `ContourPlot`, `DensityPlot` y `Plot3D` de *Mathematica* producen objetos gráficos de *Mathematica* que incluyen una lista de los valores de su función en una cuadrícula. Por consiguiente, habiendo usado cualesquiera de estas funciones, *Mathematica* fácilmente puede tomar su salida y usarla para producir otro tipo de gráfico.

He aquí el gráfico de una superficie.

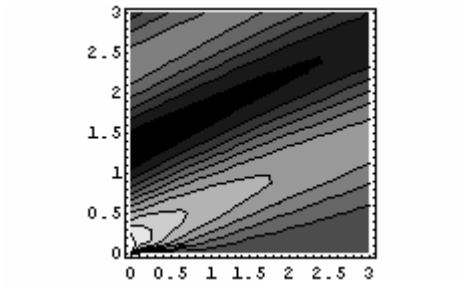
```
Plot3D[BesselJ[nu, 3x], {nu, 0, 3}, {x, 0, 3}]
```




```
- SurfaceGraphics -
```

Esto convierte el objeto producido por `Plot3D` en un gráfico de contorno.

```
Show[ ContourGraphics[ % ] ]
```



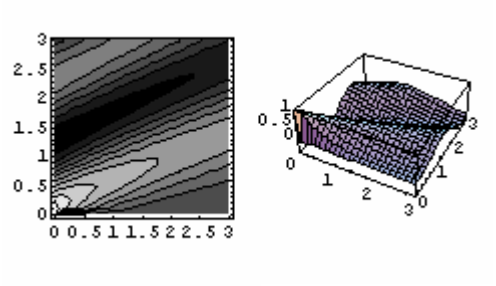

```
- ContourGraphics -
```

`Show[ContourGraphics[`*g*`]]`	convierte a un gráfico de contorno
`Show[DensityGraphics[`*g*`]]`	convierte a un gráfico de densidad
`Show[SurfaceGraphics[`*g*`]]`	convierte a una superficie
`Show[Graphics[`*g*`]]`	convierte a una imagen bidimensional

Conversiones entre tipos de gráficos.

Usted puede usar `GraphicsArray` para mostrar diferentes tipos de gráficos juntos.

```
Show[ GraphicsArray[ {%, %%} ] ]
```




```
- ContourGraphics -
```

10.8. Gráficas de listas de datos

Hasta ahora, hemos hablado como puede usar *Mathematica* para hacer los gráficos de funciones. Usted da una función a *Mathematica*, y *Mathematica* construye una curva o superficie evaluando la función en muchos puntos diferentes.

Esta sección describe como usted puede hacer gráficos de listas de datos, en vez de funciones. (La sección 12.3 menciona como leer datos de archivos externos y programas). Los comandos de *Mathematica* para trazar las listas de datos son análogos a los mencionados para trazar funciones.

`ListPlot[{`y_1`, `y_2`, ...}]`	grafica y_1, y_2, ... con los valores de x 1, 2, ...
`ListPlot[{{`x_1`, `y_1`}, {`x_2`, `y_2`}, ...}]`	grafica los puntos (x_1, y_1), ...

`ListPlot[`*list*`, PlotJoined -> True]`	una los puntos con líneas
`ListPlot3D[ {{`z_{11}`, `z_{12}`, ...}, {`z_{21}`, `z_{22}`, ...}, ...}]`	hace un gráfico tridimensional del array alturas z_{yx}
`ListContourPlot[`*array*`]`	hace un gráfico de contorno a partir de un array de alturas
`ListDensityPlot[`*array*`]`	hace un gráfico de densidad

Funciones para trazar listas de datos.

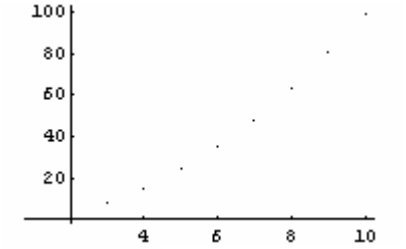
He aquí una lista de valores.

```
t = Table[i^2, {i, 10}]
```

```
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

Esto traza un gráfico de los valores.

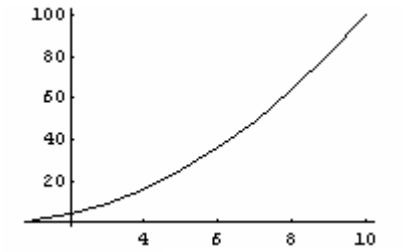
```
ListPlot[t]
```




```
- Graphics -
```

Esto une los puntos con líneas.

```
ListPlot[t, PlotJoined -> True]
```




```
- Graphics -
```

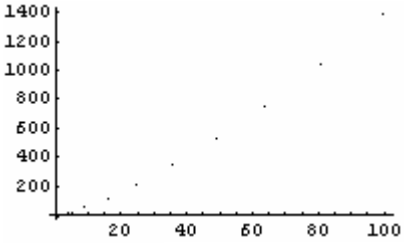
Esto da una lista de pares *x*, *y*.

```
Table[{i^2, 4 i^2 + i^3}, {i, 10}]
```

```
{{1, 5}, {4, 24}, {9, 63}, {16, 128}, {25, 225},
 {36, 360}, {49, 539}, {64, 768}, {81, 1053}, {100, 1400}}
```

Esto traza un gráfico los puntos.

```
ListPlot[%]
```



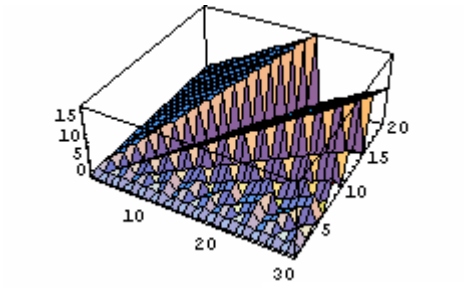
- Graphics -

Esto da un array rectangular de valores. El array es bastante grande, entonces terminamos la entrada con un punto y coma para evitar que se imprima.

```
t3 = Table[Mod[x, y], {y, 20}, {x, 30}] ;
```

Esto traza un gráfico tridimensional del array de valores.

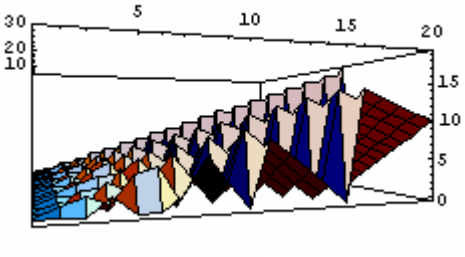
```
ListPlot3D[t3]
```



- SurfaceGraphics -

Usted puede volver a dibujar el gráfico usando Show, como de costumbre.

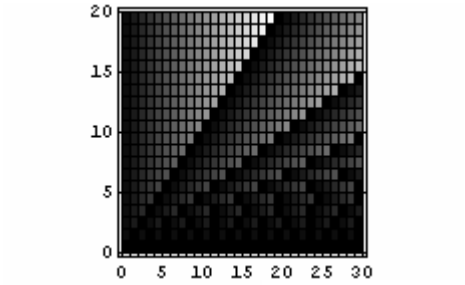
```
Show[%, ViewPoint -> {1.5, -0.5, 0}]
```



- SurfaceGraphics -

Esto traza un gráfico de densidad.

```
ListDensityPlot[t3]
```



- DensityGraphics -

10.9. Gráficos paramétricos

La sección 10.1 describió como trazar curvas en *Mathematica* en la cual usted da la coordenada *y* de cada punto como una función de la coordenada *x*. También puede usar *Mathematica* para hacer gráficos *paramétricos*. En un gráfico paramétrico, usted da ambas coordenadas *x* e *y* de cada punto como una función de un tercer parámetro, llamado *t*.

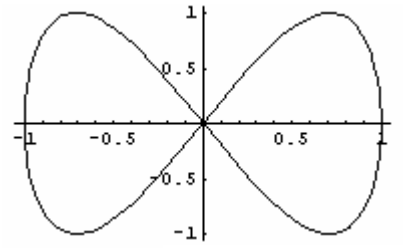
`ParametricPlot[{`f_x`,` f_y`},` `{`t`,` t_{min}`,` t_{max}`}]`	traza un gráfico paramétrico
`ParametricPlot[{` `{`f_x`,` f_y`}, {`g_x`,` g_y`}, ...},` `{`t`,` t_{min}`,` t_{max}`}]`	grafica varias curvas paramétricas juntas

`ParametricPlot[{`f_x`,` f_y`},` `{`t`,` t_{min}`,` t_{max}`},` `AspectRatio ->` `Automatic]`	trata de conservar las formas de las curvas

Funciones para generar gráficos paramétricos.

He aquí la curva hecha tomando la coordenada *x* de cada punto como `Sin[t]` y la coordenada *y* como `Sin[2t]`.

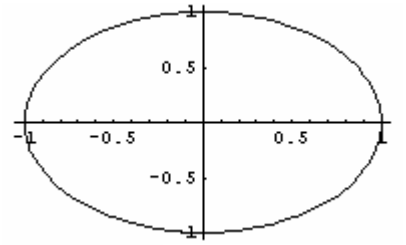
```
ParametricPlot[{Sin[t], Sin[2t]}, {t, 0, 2Pi}]
```



- Graphics -

La "forma" de la curva producida depende de la proporción de la altura y el ancho para el gráfico entero.

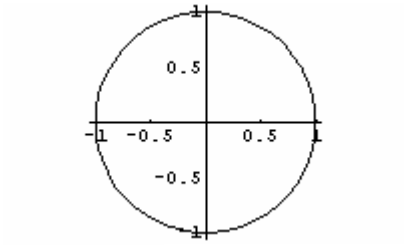
```
ParametricPlot[{Sin[t], Cos[t]}, {t, 0, 2Pi}]
```



- Graphics -

El ajuste de la opción `AspectRatio -> Automatic` hace que *Mathematica* conserve la "verdadera forma" de la curva, según lo definido por los actuales valores coordinados involucrados.

```
Show[%, AspectRatio -> Automatic]
```



- Graphics -

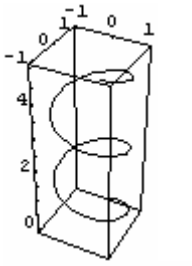
`ParametricPlot3D[` $\{f_x, f_y, f_z\}, \{t, t_{min}, t_{max}\}]$	traza un gráfico paramétrico de una curva tridimensional
`ParametricPlot3D[` $\{f_x, f_y, f_z\}, \{t, t_{min}, t_{max}\}, \{u, u_{min}, u_{max}\}]$	traza un gráfico paramétrico de una superficie tridimensional
`ParametricPlot3D[` $\{f_x, f_y, f_z, s\}, \ldots]$	sombrea las partes del gráfico paramétrico según la función *s*
`ParametricPlot[{` $\{f_x, f_y\}, \{g_x, g_y\}, \ldots\}, \ldots]$	grafica varios objetos juntos

Gráficos paramétricos tridimensionales.

`ParametricPlot3D[{`f_x, f_y, f_z`}, {`t, t_{min}, t_{max}`}]` es el análogo directo en tres dimensiones de `ParametricPlot[{`f_x, f_y`}, {`t, t_{min}, t_{max}`}]` en dos dimensiones. En ambos casos, *Mathematica* eficientemente genera una secuencia de puntos variando el parámetro *t*, luego forma una curva uniendo estos puntos. Con `ParametricPlot`, la curva está en dos dimensiones; con `ParametricPlot3D`, está en tres dimensiones.

Esto hace un gráfico paramétrico de una curva helicoidal. La variación de *t* produce movimiento circular en el plano *x*, *y*, y movimiento lineal en la dirección *z*.

```
ParametricPlot3D[{Sin[t], Cos[t], t/3}, {t, 0, 15}]
```

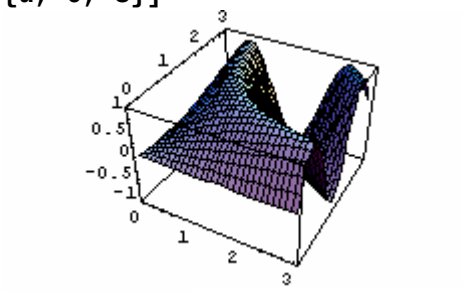


- Graphics3D -

`ParametricPlot3D[{`f_x`,` f_y`,` f_z`}, {`t`,` t_{min}`,` t_{max}`}, {`u`,` u_{min}`,` u_{max}`}]` crea una superficie, y no una curva. La superficie es formada de una colección de cuadriláteros. Las esquinas de los cuadriláteros tienen coordenadas correspondiente a los valores de los f_i cuando t y u toman valores en una cuadrícula regular.

Aquí las coordenadas x e y de los cuadriláteros están dadas simplemente por `t` y `u`. El resultado es un gráfico de la clase que puede producirse con `Plot3D`.

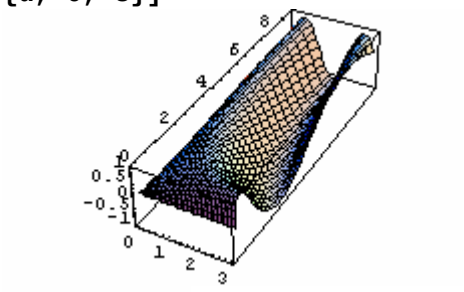
```
ParametricPlot3D[{t, u, Sin[t u]}, {t, 0, 3},
{u, 0, 3}]
```



```
- Graphics3D -
```

Esto muestra la misma superficie que antes, pero con las coordenadas deformadas por una transformación cuadrática.

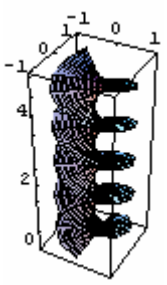
```
ParametricPlot3D[{t, u^2, Sin[t u]}, {t, 0, 3},
{u, 0, 3}]
```



```
- Graphics3D -
```

Esto produce una superficie helicoidal tomando la curva helicoidal antes mostrada (p. 157), y en cada sección de la curva dibuja un cuadrilátero.

```
ParametricPlot3D[{u Sin[t], u Cos[t], t/3},
{t, 0, 15}, {u, -1, 1}]
```

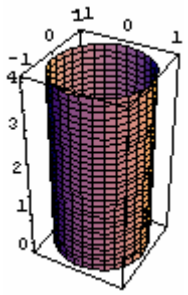


- Graphics3D -

En general, es posible construir muchas superficies complicadas usando `ParametricPlot3D`. En cada caso, puede imaginar que las superficies son formadas de alguna menera al “deformar” o “enrollar” una cuadrícula coordenada *t*, *u*.

Esto produce un cilindro. La variación del parámetro `t` origina un círculo en el plano *x*, *y*, mientras que la variación de `u` hace que los círculos se mueven en la dirección *z*.

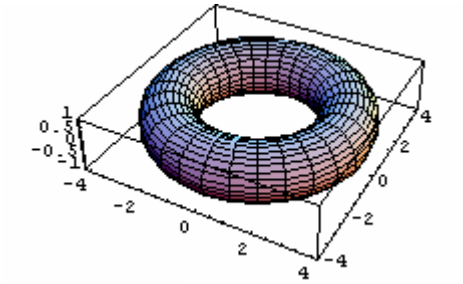
```
ParametricPlot3D[{Sin[t], Cos[t], u}, {t, 0, 2Pi},
{u, 0, 4}]
```



- Graphics3D -

Esto produce un toro La variación `u` genera un círculo, mientras que la variación de `t` hace girar el círculo alrededor del eje *z* para formar el toro.

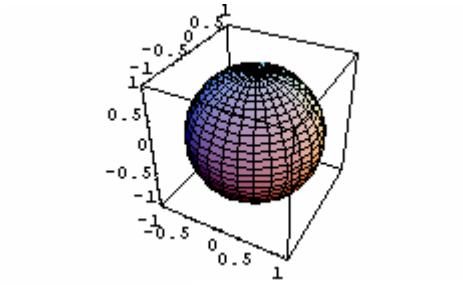
```
ParametricPlot3D[{Cos[t] (3 + Cos[u]),
Sin[t] (3 + Cos[u]), Sin[u]}, {t, 0, 2Pi},
{u, 0, 2Pi}]
```



- Graphics3D -

Esto produce una esfera.

```
ParametricPlot3D[{Cos[t] Cos[u],
Sin[t] Cos[u], Sin[u]}, {t, 0, 2Pi},
{u, -Pi/2, Pi/2}]
```



- Graphics3D -

Usted debe comprender que cuando dibuja superficies con `ParametricPlot3D`, la parametrización correcta es a menudo crucial. Debe ser cuidadoso, por ejemplo, evitando parametrizaciones en las cuales todo o parte de su superficie son cubiertos más de una vez. Tales cubrimientos múltiples a menudo conducen a discontinuidades en el dibujo de la red utilizada en la superficie, y pueden hacer que `ParametricPlot3D` tome mucho más tiempo para generar la superficie.

10.10. Algunos gráficos especiales

Mathematica incluye un lenguaje de programación lleno de gráficos. En este lenguaje, usted puede establecer muchas clases diferentes de gráficos. Algunos de los comunes son incluidos en los paquetes estándar de *Mathematica*.

`<<Graphics``	carga un paquete para establecer funciones gráficas adicionales
`LogPlot[`*f*`, {`*x*`, `x_{min}`, `x_{max}`}]`	genera un gráfico logarítmico lineal
`LogLogPlot[`*f*`, {`*x*`, `x_{min}`, `x_{max}`}]`	genera un gráfico logarítmico logarítmico
`LogListPlot[`*list*`]`	genera un gráfico logarítmico lineal de una lista de datos
`LogLogListPlot[`*list*`]`	genera un gráfico logarítmico logarítmico de una lista de datos
`PolarPlot[`*r*`, {`*t*`, `t_{min}`, `t_{max}`}]`	genera un gráfico polar del radio *r* como una función del ángulo *t*
`ErrorListPlot[{`$\{x_1, y_1, dy_1\}$`, ...}]`	genera un gráfico de datos con barras de error
`TextListPlot[{`$\{x_1, y_1,$ `"`s_1`"}, ...}]`	grafica una lista de datos con cada punto representado por la cadena de texto s_i
`BarChart[`*list*`]`	grafica una lista de datos como un diagrama de barras
`PieChart[`*list*`]`	grafica una lista de datos como un diagrama de pastel
`PlotVectorField[{`f_x, f_y`}, {`*x*`, `x_{min}`, `x_{max}`}, {`*y*`, `y_{min}`, `y_{max}`}]`	grafica el campo vectorial correspondiente a la función vectorial *f*
`ListPlotVectorField[`*list*`]`	grafica el campo vectorial correspondiente al array bidimensional de vectores en *list*
`SphericalPlot3D[`*r*`, {`*theta*`, `*min*`, `*max*`}, {`*phi*`, `*min*`, `*max*`}]`	genera un gráfico esférico tridimensional

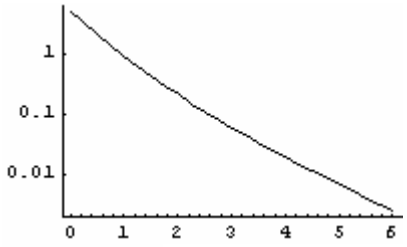
Algunas funciones especiales para graficar definidas en los paquetes estándar de *Mathematica*.

Esto carga un paquete estándar de *Mathematica* para establecer funciones gráficas adicionales.

```
<<Graphics`
```

Esto traza un gráfico logarítmico lineal.

```
LogPlot[ Exp[-x] + 4 Exp[-2x], {x, 0, 6} ]
```



- Graphics -

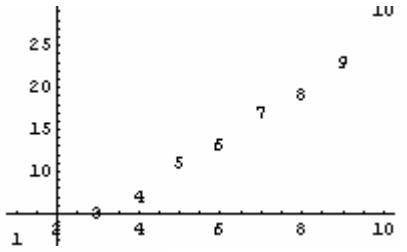
He aquí una lista de los 10 primeros primos.

```
p = Table[Prime[n], {n, 10}]
```

```
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29}
```

Esto traza un gráfico de los primos usando los enteros 1, 2, 3, ... símbolos parta graficar.

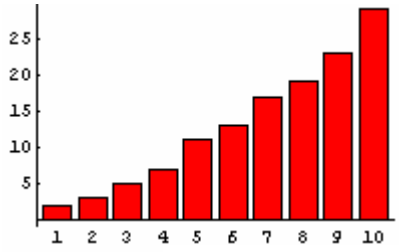
```
TextListPlot[p]
```



- Graphics -

He aquí un gráfico de barras de los primos.

```
BarChart[p]
```



- Graphics -

Esto traza un gráfico de pastel.

```
PieChart[p]
```


```
- Graphics -
```

10.11. Gráficos animados

En muchos sistemas informáticos, *Mathematica* puede producir imágenes no sólo estáticas, sino también gráficos animados o “películas”.

La idea básica en todos los casos es generar una secuencia de “cuadros” que pueden ser mostrados en forma rápida uno tras otro. Puede usar las funciones gráficas estándar de *Mathematica* descritas en la sección 10.10 para producir cada marco. El mecanismo para mostrar los cuadros como película depende del interfase de *Mathematica* que usted usa. Con una interfase de cuaderno, usted pone los cuadros en una secuencia de celdas, luego selecciona las celdas y escoge un comando para animarlos. Con una interfase basada en texto, hay a menudo un programa externo provisto para mostrar gráficos animados. Típicamente puede accederse al programa dentro de *Mathematica* usando la función `Animate`.

`<<Graphics`Animation``	carga el paquete de animación (si fuera necesario)
`Animate[`*plot*`,` `{`*t*`,` t_{min}`,` t_{max}`}]`	ejecuta el comando de gráficos con *plot* para una secuencia de valores de *t*, y anima la secuencia resultante de cuadros
`ShowAnimation[` `{`g_1`,` g_2`, ...}]`	produce una animación de una secuencia de objetos de gráficos

Formas típicas para producir gráficos animados.

Cuando usted produce una secuencia de cuadros para una película, es importante que los diferentes cuadros sean consistentes. Así, por ejemplo, usted debe dar un ajuste explícito para la opción `PlotRange`, antes que usar el ajuste por defecto

`Automatic`, para asegurarse que las escalas usadas en marcos diferentes sean las mismas. Si usted tiene gráficos tridimensionales con puntos de vista diferentes, de modo similar debe ajustar `SphericalRegion -> True` para asegurarse que la escala de gráficos diferentes es la misma.

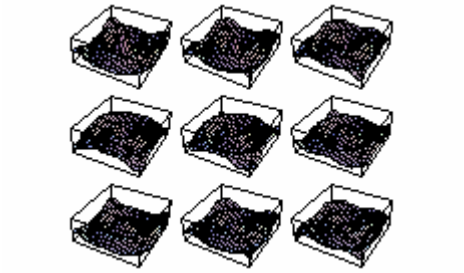
Esto genera una lista de objetos gráficos. Ajustando `DisplayFunction -> Identity` se detiene la salida que `Plot3D` produce. Explícitamente el ajuste de PlotRange asegura que la escala es la misma en cada uno de los gráficos.

```
Table[ Plot3D[ BesselJ[0, Sqrt[x^2 + y^2] + t],
{x, -10, 10}, {y, -10, 10}, Axes -> False,
PlotRange -> {-0.5, 1.0},
DisplayFunction -> Identity ], {t, 0, 8} ] // Short
```

```
{- SurfaceGraphics -, <<7>>, - SurfaceGraphics -}
```

Sobre un sistema informático apropiado, `ShowAnimation[%]` animaría los gráficos. Esto divide la gráfica en tres filas, y muestra el array resultante de imágenes.

```
Show[ GraphicsArray[ Partition[%, 3] ] ]
```



```
- GraphicsArray -
```

10.12. Sonido

En la mayor parte de sistemas informáticos, *Mathematica* puede producir no sólo gráficos, sino también sonido. *Mathematica* trata los gráficos y el sonido de una forma bastante análoga.

Por ejemplo, tal como usted puede usar `Plot[`f`, {`x`, `x_{min}`, `x_{max}`}]` para trazar el gráfico de una función, tan también puede usar `Play[`f`, {`t`, 0, `t_{max}`}]` para "producir el sonido" de una función. `Play` toma la función para definir la forma

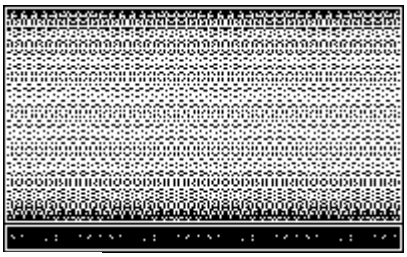
de la onda para un sonido: los valores de la función dan la amplitud del sonido como una función del tiempo.

`Play[`*f*`, {`*t*`, 0, `t_{max}`}]`	produce un sonido con la amplitud *f* como una función del tiempo *t* en segundos

Función para producir sonido.

En un sistema informático conveniente, esto produce un tono puro con una frecuencia de 440 hertz durante un segundo.

```
Play[Sin[2Pi 440 t], {t, 0, 1}]
```

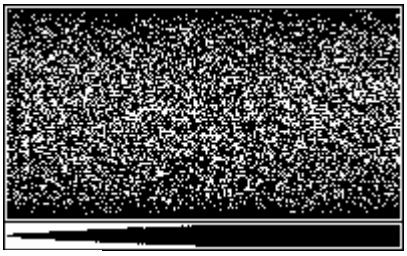


-Sound-

Los sonidos producidos por `Play` pueden tener cualquier forma de onda. Ellos, por ejemplo, no tienen que consistir en una colección de piezas armónicas. En general, la función amplitud que usted da a `Play` especifica la señal instantánea asociada con el sonido. Esta señal es convertida en un voltaje, y en última instancia en un desplazamiento. Note que la *amplitud* a veces es definida para ser la señal *pico* asociada con un sonido; en *Mathematica*, esto es siempre la señal *instantánea* como una función del tiempo.

Esto produce un sonido más complejo.

```
Play[ Sin[700 t + 25 t Sin[350 t]], {t, 0, 4} ]
```



-Sound-

`Play` se establece de modo que la variable de tiempo que aparece en él siempre sea medida en segundos absolutos. Cuando un sonido es producido, su amplitud es muestreada un cierto número de veces cada segundo. Usted puede especificar la tasa de muestreo ajustando la opción `SampleRate`.

`Play[`*f*`, {`*t*`, 0, `t_{max}`}, SampleRate ->` *r*`]`	produce un sonido, muestreándolo *r* veces por segundo

Especificación de la tasa de muestreo para un sonido.

Una tasa de muestreo *r* permite frecuencias de hasta *r*/2 hertz. El sistema humano auditivo puede percibir sonidos en una frecuencia que se extiende de 20 a 22000 hertz (dependiendo de la edad y el sexo). Las frecuencias fundamentales para las 88 notas de un piano se extienden de 27.5 a 4096 hertz. La tasa estándar de muestreo usada para reproducir discos compactos es 44100. La tasa eficaz de muestreo en un típico sistema telefónico es alrededor de 8000. En la mayor parte de sistemas informáticos, la tasa de muestreo por defecto usada por *Mathematica* es alrededor de 8000.

Usted puede usar `Play[{`f_1`, `f_2`}, ...]` para producir sonido estéreo. En general, *Mathematica* soporta cualquier número de canales de sonido.

`ListPlay[{`a_1`, `a_2`, ...}, SampleRate ->` *r*`]`	produce un sonido con una secuencia de niveles de amplitud

Funciones para manipular gráficos.

La función `ListPlay` le permite simplemente dar una lista de valores que son tomados como amplitudes de sonido muestreadas en una cierta tasa.

Cuando los sonidos son dados por *Mathematica*, sólo cierto rango de amplitudes es permitida. La opción `PlayRange` en `Play` y `ListPlay` especifica como deberían ser escaladas las amplitudes que usted da para ajustarlas al rango permitido. Los ajustes para esta opción son análogos a los de la opción de gráficos `PlotRange` mencionada en la sección 10.2.

`PlayRange -> Automatic` (default)	usa un procedimiento interno para escalar amplitudes
`PlayRange -> All`	escala de modo que todas las amplitudes quepan en el rango permitido
`PlayRange -> {`*amin*`, `*amax*`}`	ajusta amplitudes entre *amin* y *amax* en el rango permitido

Especificación del escalamiento de amplitudes de sonido.

Mientras a menudo es conveniente usar el ajuste por defecto `PlayRange -> Automatic`, debe comprender que `Play` puede correr considerablemente más rápido si usted especifica un `PlayRange`.

`Show[`*sound*`]`	usa algoritmos internos

Reproduciendo un objeto de sonido.

Tanto `Play` como `ListPlay` devuelven objetos `Sound` que contienen procedimientos para sintetizar sonidos. Puede volver a producir un objeto `Sound` en particular usando `Show` que también se usa para volver a mostrar los gráficos.

11. Entradas y salidas en cuadernos

11.1. Ingreso de letras griegas

clic en α	use un botón en una paleta
`\[Alpha]`	use un nombre completo
`[ESC]a[ESC]` or `[ESC]alpha[ESC]`	use un alias estándar (abajo se ve como `:a:`)
`[ESC]\alpha[ESC]`	use un alias TeX
`[ESC]&agr[ESC]`	use un alias SGML

Formas de ingresar letras griegas en un cuaderno.

He aquí una paleta para ingresar letras griegas comunes.

Usted puede usar letras griegas igual que las letras ordinarias que digita en su teclado.

```
Expand[(α + β)^3]
```

$$\alpha^3 + 3\,\alpha^2\,\beta + 3\,\alpha\,\beta^2 + \beta^3$$

Hay varias formas de ingresar letras griegas. Esta entrada usa nombres completos.

```
Expand[(\[Alpha] + \[Beta])^3]
```

$$\alpha^3 + 3\,\alpha^2\,\beta + 3\,\alpha\,\beta^2 + \beta^3$$

	nombre completo	*alias*		*nombre completo*	*alias*
α	\[Alpha]	:a:, :alpha:	Γ	\[CapitalGamma]	:G:, :Gamma:
β	\[Beta]	:b:, :beta:	Δ	\[CapitalDelta]	:D:, :Delta:
γ	\[Gamma]	:g:, :gamma:	Θ	\[CapitalTheta]	:Q:, :Th:, :Theta:
δ	\[Delta]	:d:, :delta:	Λ	\[CapitalLambda]	:L:, :Lambda:
ϵ	\[Epsilon]	:e:, :epsilon:	Π	\[CapitalPi]	:P:, :Pi:
ζ	\[Zeta]	:z:, :zeta:	Σ	\[CapitalSigma]	:S:, :Sigma:
η	\[Eta]	:h:, :et:, :eta:	Υ	\[CapitalUpsilon]	:U:, :Upsilon:
θ	\[Theta]	:q:, :th:, :theta:	Φ	\[CapitalPhi]	:F:, :Ph:, :Phi:
κ	\[Kappa]	:k:, :kappa:	$\Chi$	\[CapitalChi]	:C:, :Ch:, :Chi:
λ	\[Lambda]	:l:, :lambda:	Ψ	\[CapitalPsi]	:Y:, :Ps:, :Psi:
μ	\[Mu]	:m:, :mu:	Ω	\[CapitalOmega]	:O:, :W:, :Omega:
ν	\[Nu]	:n:, :nu:			
ξ	\[Xi]	:x:, :xi:			
π	\[Pi]	:p:, :pi:			
ρ	\[Rho]	:r:, :rho:			
σ	\[Sigma]	:s:, :sigma:			
τ	\[Tau]	:t:, :tau:			
ϕ	\[Phi]	:f:, :ph:, :phi:			
φ	\[CurlyPhi]	:j:, :cph:, :cphi:			
χ	\[Chi]	:c:, :ch:, :chi:			
ψ	\[Psi]	:y:, :ps:, :psi:			
ω	\[Omega]	:o:, :w:, :omega:			

Letras griegas comúnmente usadas. Los alias : se obtienen con la tecla ⎈. Los alias TeX no son listados explícitamente.

Note que en *Mathematica* la letra π se genera con `Pi`. Ninguna de las otras letras griegas tiene un significado especial.

π se genera con `Pi`.

```
N[π]
```

```
3.14159
```

Usted puede usar letras griegas solas o con otras letras.

```
Expand[(Rαβ + Ξ)^4]
```

$$R\alpha\beta^4 + 4\,R\alpha\beta^3\,\Xi + 6\,R\alpha\beta^2\,\Xi^2 + 4\,R\alpha\beta\,\Xi^3 + \Xi^4$$

El símbolo $\pi\alpha$ no es relacionado con el símbolo π.

```
Factor[πα^4 - 1]
```

$$(-1 + \pi\alpha)\,(1 + \pi\alpha)\,(1 + \pi\alpha^2)$$

11.2. Ingreso de entradas bidimensionales

Cuando *Mathematica* lee el texto `x^y`, lo interpreta como `x` elevado a la potencia `y`.

```
x^y
```

$$x^y$$

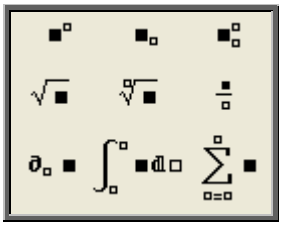
En un cuaderno, usted puede dar la entrada bidimensional x^y directamente.

```
x^y
```

$$x^y$$

Una manera de ingresar una forma bidimensional como x^y en un cuaderno de *Mathematica* es copiar esta forma de una paleta pulsando el botón apropiado en la paleta.

He aquí una paleta para ingresar algunas notaciones bidimensionales comunes.



Hay también varias maneras de ingresar formas bidimensionales directamente desde el teclado.

x Ctrl-^ *y* Ctrl-␣	use combinaciones de teclas que existen en la mayoría de teclados
x Ctrl-6 *y* Ctrl-␣	use combinaciones de teclas que deberían existir en todos los teclados
\! \ (*x*\^*y*\) seguido por Make 2D	use sólo caracteres imprimibles ordinarios

Formas de ingresar un superíndice directamente desde el teclado. Ctrl-␣ se usa para representar Control-Espacio.

Para efectuar la combinación Ctrl-^ presione la tecla Control y manteniéndola presionada, presione la tecla ^. En cuanto hace esto, su cursor saltará de la posición de exponente. Entonces puede escribir lo que quiera y aparecerá en aquella posición.

Cuando ha terminado, presiona Ctrl-␣ para regresar el cursor de la posición de exponente a la posición de escritura; Ctrl-␣ lo efectúa manteniendo presionada la tecla Control, para luego presionar la barra espaciadora.

Ésta secuencia de presionamientos de teclas ingresa x^y.

```
x Ctrl-^ y
```

x^y

Aquí la expresión completa `y+z` está en el exponente.

```
x Ctrl-^ y + z
```

x^{y+z}

Presionando Ctrl-␣ (Control-Espacio) baja de la posición de exponente.

```
x Ctrl-^ y Ctrl-␣ + z
```

$x^y + z$

Usted puede confundir el hecho que Ctrl-^ le da un exponente y pensar que Ctrl-^ es solamente una forma más inmediata de ^. Cuando digita usted `x^y`, *Mathematica* dejará esta forma unidimensional sin cambios hasta que la procese explícitamente. Pero si usted digita x Ctrl-^ `y` entonces *Mathematica* inmediatamente le dará un exponente.

En un teclado estándar de lengua inglesa, el carácter ^ aparece en la parte superior de la tecla 6. *Mathematica* por lo tanto acepta Ctrl-6 como una alternativa a Ctrl-^. Tenga presente que si usted usa un teclado diferente, *Mathematica* casi siempre aceptará Ctrl-6, pero puede que no acepte Ctrl-^.

Ésta es una forma alternativa de entrada que evita el uso de la tecla Control.

```
\!\( x \^ y \)
```

x^y

Con esta forma de la entrada, *Mathematica* entiende automáticamente que +z no debe ir en el exponente.

```
\!\( x \^ y + z \)
```

$x^y + z$

La utilización de la tecla Control reduce al mínimo el número de presionamientos de teclas que tiene que hacer para incorporar un exponente. Pero en particular si quiere guardar su entrada en un archivo, o enviarlo a otro programa, a menudo es más conveniente usar una forma que no involucre la tecla Control. Usted puede hacer esto utilizando secuencias de \!.

Si usted copia una secuencia de \! en *Mathematica*, ésta automáticamente aparecerá en forma bidimensional. Pero si ingresa la secuencia directamente del teclado, tiene que escoger explícitamente el ítem Make 2D del menú Edit para conseguir la forma bidimensional.

Cuando ingresa desde el teclado secuencias de \(… \), éstas son mostradas en forma literal.

```
\!\(x\^y + z\)
```

Escogiendo el ítem Make 2D en el menú Edit convierte estas secuencias en formas bidimensionales.

$x^y + z$

x CTRL[_] y CTRL[ ]	use combinaciones de teclas que existen en la mayoría de teclados
x CTRL[-] y CTRL[ ]	use combinaciones de teclas que deberían existir en todos los teclados
\! \(x_y\) seguido por Make 2D	use sólo caracteres imprimibles ordinarios

Formas de ingresar un subíndice directamente desde el teclado.

Los subíndices en *Mathematica* trabajan de manera muy similar a los superíndices. Sin embargo, mientras que *Mathematica* interpreta automáticamente x^y como x

elevada a la potencia y, no tiene ninguna interpretación similar para x_y. Y, sólo trata a x_y como un objeto puramente simbólico.

Esto ingresa y como un subíndice.

```
x CTRL[_] y
```

x_y

He aquí otra forma de ingresar y como un subíndice.

```
\!\( x \_ y \)
```

x_y

x CTRL[/] *y* CTRL[␣]	use la tecla Control
\!\(*x*\/*y*\) seguido por Make 2D	use sólo caracteres imprimibles ordinarios

Formas de ingresar una fracción directamente desde el teclado.

Esto ingresa la fracción $\frac{x}{y}$.

```
x CTRL[/] y + z
```

$\frac{x}{y} + z$

Aquí la expresión completa y+z va al denominador.

```
x CTRL[/] y + z
```

$\frac{x}{y + z}$

Pero presionando Control-Espacio se sale denominador, así que +z no aparece en el denominador.

```
x CTRL[/] y CTRL[␣] + z
```

$\frac{x}{y} + z$

Mathematica automáticamente interpreta una fracción como una división.

```
8888
----
2222
```

$\frac{8888}{2222}$

4

He aquí otra forma de ingresar una fracción.

```
\!\( 8888 \/ 2222 \)
4
```

Ctrl-[@] *x* Ctrl-[␣]	use combinaciones de teclas que existen en la mayoría de teclados
Ctrl-[2] *x* Ctrl-[␣]	use combinaciones de teclas que deberían existir en todos los teclados
\!\(\@*x*\) seguido por Make 2D	use sólo caracteres imprimibles ordinarios

Formas de ingresar una raíz cuadrada directamente desde el teclado.

Esto ingresa una raíz cuadrada.

Ctrl-[@] **x + y**

$\sqrt{x+y}$

Control-Espacio le saca de la raíz cuadrada.

Ctrl-[@] **x** Ctrl-[␣] **+ y**

$\sqrt{x}+y$

He aquí una forma sin usar la tecla Control.

```
\!\( \@ x + y \)
```

$\sqrt{x}+y$

Y he aquí una usual entrada unidimensional en *Mathematica* que da la misma expresión como salida.

```
Sqrt[x] + y
```

$\sqrt{x}+y$

Ctrl-[^] ó Ctrl-[6]	va a la posición de superíndice
Ctrl-[_] ó Ctrl-[-]	va a la posición de subíndice
Ctrl-[@] ó Ctrl-[2]	va al subradical de una raíz cuadrada
Ctrl-[%] ó Ctrl-[5]	va de subíndice a exponente o viceversa, o a la posición de índice radical
Ctrl-[/]	va al denominador de una fracción
Ctrl-[␣]	retorna de una posición especial (Control-Espacio)

Formas especiales de entrada usando la tecla Control. La segunda forma dada debería trabajar en cualquier teclado.

Esto pone tanto subíndice como superíndice en x.

```
x [CTRL-^] y [CTRL-%] z
```


He aquí otra forma de ingresar la misma expresión.

```
x [CTRL-_] z [CTRL-%] y
```


\!\(… \)	agrupación de toda entrada bidimensional
x \^ y	superíndice x^y con \!\(… \)
x _ y	subíndice x_y con \!\(… \)
x \^ y \% z	subíndice y superíndice x_z^y con \!\(… \)
\@ x	raíz cuadrada $\sqrt{x}$ con \!\(… \)
x \/ y	fracción $\frac{x}{y}$ con \!\(… \)

Formas especiales de entrada que generan entradas bidimensionales con el ítem Make 2D.

Usted debe inicializar los \(exteriores con \!.

```
\!\(a \/ b + \@ c \) + d
```


Usted puede usar \(y \) para indicar la agrupación de elementos en una expresión sin introducir paréntesis explícitos.

```
\!\(a \/ \( b + \@ c \) \) + d
```


Además de subíndices y superíndices, *Mathematica* también soporta la noción de y sobreescrituras–elementos de que van directamente debajo o encima. Entre otras cosas, usted puede usar subescrituras y sobreescrituras para ingresar los límites de sumas y productos.

x [CTRL-+] y [CTRL-.] ó x [CTRL-=] y [CTRL-.]	crea un subescrito x_y
\!\(x\+y\) seguido por Make 2D	crea un subescrito x_y
x [CTRL-&] y [CTRL-.] ó x [CTRL-7] y [CTRL-.]	crea un sobreescrito $\overset{y}{x}$
\!\(x\&y\) seguido por Make 2D	crea un sobreescrito $\overset{y}{x}$

Creación de subescrituras y sobreescrituras.

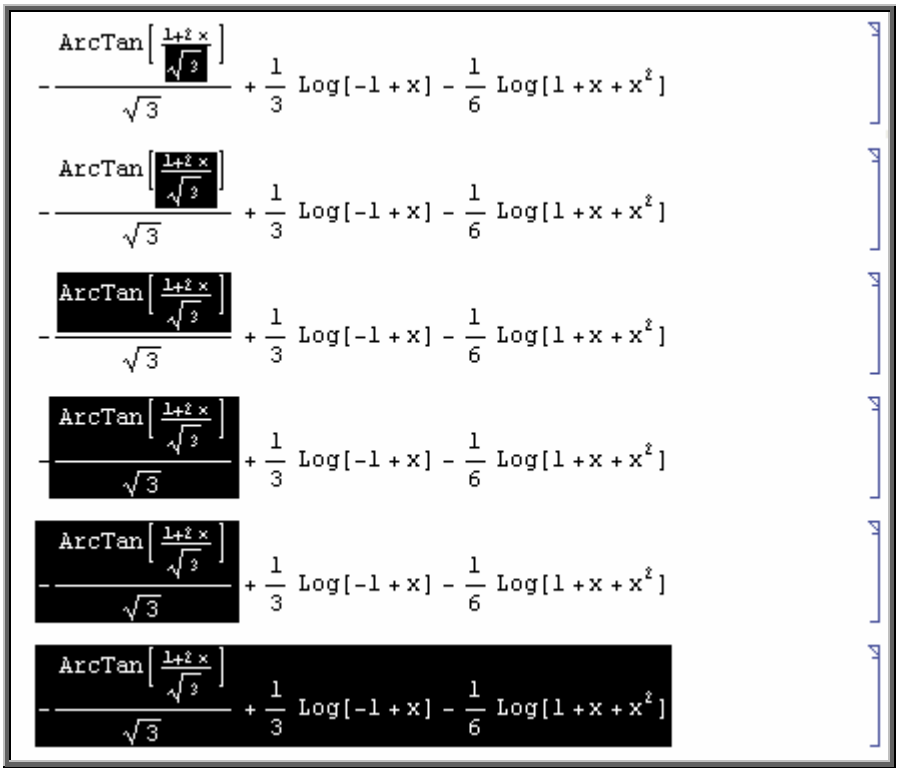
11.3. Edición y evaluación de expresiones bidimensionales

Cuando usted ve una expresión bidimensional en la pantalla, puede editarla como lo hace con el texto. Por ejemplo puede colocar su cursor en algún sitio y empezar a digitar. O puede seleccionar una parte de la expresión, luego quitarla usando la tecla Surpimir, o insertar una nueva versión digitándola.

CTRL[.]	selecciona la subexpresión siguiente
CTRL[␣]	mueve a la derecha de la estructura actual
→	mueve al siguiente caracter
←	mueve al anterior caracter

Formas de moverse alrededor de expresiones bidimensionales.

Esto muestra una secuencia de subexpreiones seleccionadas al presionar repetidamente CTRL[.].



Shift-Enter	evalúa la celda actual completa
Shift-Control-Enter	evalúa sólo la subexpresión seleccionada

Formas de evaluar expresiones bidimensionales.

En la mayor parte de cálculos, usted querrá ir de un paso al siguiente tomando la expresión completa que ha generado, y luego evaluándola. Pero si por ejemplo trata de manipular solamente una fórmula para expresarla en una forma particular, puede en cambio hallar más conveniente el hecho de realizar una secuencia de operaciones independientes sobre las diferentes partes de la expresión.

Usted hace esto seleccionando cada parte que quiere operar, para luego usar la combinación Shift-Control-Enter.

He aquí una expresión con una parte seleccionada.

```
{Factor[x^4 - 1], Factor[x^5 - 1], Factor[x^6 - 1],
 Factor[x^7 - 1]}
```

Presionando Shift-Control-Enter evalúa la parte seleccionada.

```
{Factor[x^4 - 1], (-1 + x) (1 + x + x^2 + x^3 + x^4), Factor[x^6 - 1],
 Factor[x^7 - 1]}
```

11.4. Ingreso de fórmulas

caracter	*forma corta*	*forma larga*	*símbolo*
π	:p:	\[Pi]	Pi
∞	:inf:	\[Infinity]	Infinity
°	:deg:	\[Degree]	Degree

Formas especiales de algunos símbolos comunes. : se obtienen con la tecla Esc.

Esto equivale a `Sin[60 Degree]`.

```
Sin[60°]
```

$$\frac{\sqrt{3}}{2}$$

He aquí la forma larga de la entrada anterior.

```
Sin[60 \[Degree]]
```

$$\frac{\sqrt{3}}{2}$$

Esto encuentra los mejores parámetros para un ajuste general.

```
Sin[60 :deg:]
```

$$\frac{\sqrt{3}}{2}$$

Aquí el ángulo está en radianes.

```
Sin[Pi/3]
```

$$\frac{\sqrt{3}}{2}$$

caracteres especiales	*forma corta*	*forma larga*	*caracteres ordinarios*
$x \le y$	x`:<=:`y	x`\[LessEqual]`y	x`<=`y
$x \ge y$	x`:>=:`y	x`\[GreaterEqual]`y	x`>=`y
$x \neq y$	x`:!=:`y	x`\[NotEqual]`y	x`!=`y
$x \in y$	x`:el:`y	x`\[Element]`y	`Element[`x`,`y`]`
$x \to y$	`:deg:`	x`\[Rule]`y	x`->`y

Formas especiales para algunos operadores.

Aquí la regla de transformación es ingresada usando dos caracteres ordinarios, como `->`.

```
x/(x+1) /. x -> 3 + y
```

$$\frac{3+y}{4+y}$$

Esto significa exactamente lo mismo.

```
x/(x+1) /. x \[Rule] 3 + y
```

$$\frac{3+y}{4+y}$$

También esto.

```
x/(x+1) /. x → 3 + y
```

$$\frac{3+y}{4+y}$$

O esto.

```
x/(x+1) /. x :->: 3 + y
```

$$\frac{3+y}{4+y}$$

La forma especial de la flecha → es también usada por defecto para salidas.

```
Solve[x^2 == 1, x]
```

```
{{x → -1}, {x → 1}}
```

caracteres especiales	*forma corta*	*forma larga*	*caracteres ordinarios*
$x \div y$	*x*:div:*y*	*x*\[Divide]*y*	*x* / *y*
$x \times y$	*x*:*:*y*	*x*\[Times]*y*	*x* * *y*
$x \times y$	*x*:cross:*y*	*x*\[Cross]*y*	Cross[*x*, *y*]
$x == y$	*x*:==:*y*	*x*\[Equal]*y*	*x* == *y*
$x = y$	*x*:l=:*y*	*x*\[LongEqual]*y*	*x* == *y*
$x \wedge y$	*x*:&&:*y*	*x*\[And]*y*	*x* && *y*
$x \vee y$	*x*:||:*y*	*x*\[Or]*y*	*x* || *y*
$\neg x$	:!:*x*	\[Not]*x*	!*x*
$x \Rightarrow y$	*x*:=>:*y*	*x*\[Implies]*y*	Implies[*x*, *y*]
$x \cup y$	*x*:un:*y*	*x*\[Union]*y*	Union[*x*, *y*]
$x \cap y$	*x*:inter:*y*	*x*\[Intersection]*y*	Intersection[*x*, *y*]
xy	*x*:,:*y*	*x*\[Invisible Comma]*y*	*x*, *y*
fx	*f*:@:*x*	*x*\[Invisible Application]*y*	*f*@*x* ó *f*[*x*]

Algunos operadores con formas especiales usados para entradas pero no para salidas.

Mathematica entiende ÷, pero no lo usa por defecto para salidas.

```
x ÷ y
```

$$\frac{x}{y}$$

Las formas de entrada discutidas hasta ahora en esta sección usan caracteres especiales, pero de en otros casos sólo consisten en líneas ordinarias unidimensionales de texto. Los cuadernos de *Mathematica*, sin embargo, también hacen posible el uso de formas bidimensionales de entrada.

bidimensional	*unidimensional*	
x^y	`x ^ y`	potencia
$\frac{x}{y}$	`x / y`	división
$\sqrt{x}$	`Sqrt[x]`	raíz cuadrada
$\sqrt[n]{x}$	`x ^ (1/n)`	raíz *n*-ésima
$\sum_{i=i_{min}}^{i_{max}} f$	`Sum[f, {i, imin, imax}]`	suma
$\prod_{i=i_{min}}^{i_{max}} f$	`Product[f, {i, imin, imax}]`	producto
$\int f \, dx$	`Integrate[f, x]`	integral indefinida
$\int_{x_{min}}^{x_{max}} f \, dx$	`Integrate[f, {x, xmin, xmax}]`	integral definida
$\partial_x f$	`D[f, x]`	derivada parcial
$\partial_{x,y} f$	`D[f, x, y]`	derivada parcial multivariable
z^*	`Conjugate[z]`	conjugado complejo
m^T	`Transpose[m]`	transpuesta
$m^\dagger$	`Conjugate Transpose[m]`	conjugada transpuesta
$expr_{[[i,j,\ldots]]}$	`Part[expr, i, j, ...]`	extracción de partes

Algunas formas bidimensionales que pueden usarse en cuadernos de *Mathematica*.

Usted puede ingresar formas bidimensionales usando cualquiera de los mecanismos mencionados en la sección 11.2. Note que límites superiores e inferiores para sumas y productos deben ser ingresados como sobreescrituras y subescrituras–no sólo como superíndices y subíndices.

Esto ingresa una integral indefinida. Note el uso de `:dd:` para ingresar la "diferencial d".

```
:int: f[x] :dd: x
```

$$\int f[x]\, dx$$

He aquí una integral indefinida que puede ser evaluada explícitamente.

$$\int \mathrm{Exp}[-x^2]\,dx$$

$$\frac{1}{2}\sqrt{\pi}\,\mathrm{Erf}[x]$$

He aquí la entrada usual de *Mathematica* para esta integral.

```
Integrate[Exp[-x^2], x]
```

$$\frac{1}{2}\sqrt{\pi}\,\mathrm{Erf}[x]$$

Esto ingresa exactamente la misma integral.

```
\!\( \[Integral] Exp[-x\^2] \[DifferentialD]x \)
```

$$\frac{1}{2}\sqrt{\pi}\,\mathrm{Erf}[x]$$

forma corta	*forma larga*	
:sum:	\[Sum]	signo de sumatoria Σ
:prod:	\[Product]	signo de productoria $\prod$
:int:	\[Integral]	signo de integral $\int$
:dd:	\[DifferentialD]	d especial para uso en integrales
:pd:	\[PartialD]	operador de derivada parcial ∂
:co:	\[Conjugate]	símbolo para conjugado *
:tr:	\[Transpose]	símbolo para transpuesta ^{T}
:ct:	\[Conjugate Transpose]	símbolo para conjugada transpuesta †
:[[:, :]]:	\[LeftDouble Bracket], \[RightDouble Bracket]	pares de corchetes

Algunos caracteres especiales usados para ingresar fórmulas.

Usted debe comprender que aun cuando un signo de sumatoria pueda parecer casi idéntico a una sigma mayúscula son tratados de un modo muy diferente por *Mathematica*. El punto es que sigma es sólo una letra; pero un signo de sumatoria es un operador que dice a *Mathematica* que realice una operación `Sum`.

La sigma mayúscula es sólo una letra.

```
a + \[CapitalSigma]^2
```

$$a + \Sigma^2$$

Un signo de sumatoria, por otra parte, es un operador.

```
Esc sum Esc  Ctrl [+] n=0  Ctrl [^] m  Ctrl [_] 1/f[n]
```

$$\sum_{n=0}^{m} \frac{1}{f[n]}$$

He aquí otra forma de ingresar la misma entrada.

```
\!\( \[Sum] \+ \( n = 0 \) \% m 1 \/ f[n] \)
```

$$\sum_{n=0}^{m} \frac{1}{f[n]}$$

Así como *Mathematica* distingue entre un signo de sumatoria y una sigma mayúscula, también distingue entre una d ordinaria y la "∂ diferencial" especial que es usada en la notación estándar para integrales. Es crucial que use esta ∂ diferencial–ingresada como Esc dd Esc–cuando digita una integral. Si trata de usar una d ordinaria, *Mathematica* la interpretará como un símbolo llamado d–no entenderá que usted esta ingresando la segunda parte de un operador de integración.

Esto calcula la derivada de x^n.

```
∂x x^n
```

$$\partial_x x^n$$

$$n\,x^{-1+n}$$

He aquí la misma derivada especificada en una forma unidimensional ordinaria.

```
D[x^n, x]
```

$$n\,x^{-1+n}$$

Esto calcula la tercera derivada.

```
∂x,x,x x^n
```

$$\partial_{x,x,x} x^n$$

$$(-2+n)\,(-1+n)\,n\,x^{-3+n}$$

He aquí la forma unidimensional equivalente.

```
D[x^n, x, x, x]
```

$$(-2+n)\,(-1+n)\,n\,x^{-3+n}$$

11.5. Ingreso de tablas y matrices

El front end de *Mathematica* proporciona un ítem Create Table/Matrix/Palette que le permite crear un array en blanco con cualquier número especificado de filas y columnas. Una vez que tiene este array, puede editarlo llenándolo con cualesquiera elementos que usted desee.

Mathematica trata un array como éste como una matriz representada por una lista de listas.

```
a  b  c
1  2  3
{{a, b, c}, {1, 2, 3}}
```

Poniendo paréntesis alrededor del array hacemos que se parezca más a una matriz, pero esto no afecta su interpretación.

$$\begin{pmatrix} a & b & c \\ 1 & 2 & 3 \end{pmatrix}$$

```
{{a, b, c}, {1, 2, 3}}
```

Usando `MatrixForm` le dice a *Mathematica* que muestre el resultado de `Transpose` como una matriz.

```
MatrixForm[Transpose[{{a, b, c}, {1, 2, 3}}]]
```

$$\begin{pmatrix} a & 1 \\ b & 2 \\ c & 3 \end{pmatrix}$$

CTRL[,]	agrega una columna
CTRL[↵] (Control-Enter)	agrega una fila
Tab	va al siguiente elemento □ ó ■
CTRL[␣] (Control-Espacio)	lo saca de la tabla o matriz

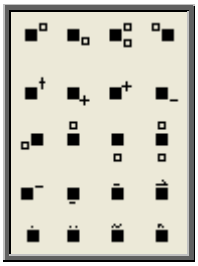
Ingresando tablas y matrices.

Note que puede usar Ctrl[,] y Ctrl[↵] para empezar a aumentar un array, en particular para arrays pequeños hacer esto es a menudo más conveniente que usar el ítem Create Table/Matrix/Palette.

El ítem Create Table/Matrix/Palette le permite hacer ajustes básicos, como el dibujo de líneas entre filas o columnas.

11.6. Subíndices, barras y otros adornos

He aquí una típica paleta de adornos.



Mathematica le permite usar cualquier expresión como subíndice.

```
Expand[(1 + x1+n)^4]
```

$$1 + 4\,x_{1+n} + 6\,x_{1+n}^2 + 4\,x_{1+n}^3 + x_{1+n}^4$$

A menos que usted lo exprese de otro modo, *Mathematica* interpretará un superíndice como una potencia.

```
Factor[x_n^4 - 1]
```

$$(-1 + x_n)\,(1 + x_n)\,(1 + x_n^2)$$

Ctrl[_] ó Ctrl[-]	va a la posición de subíndice
Ctrl[+] ó Ctrl[=]	va a la posición de subescritura
Ctrl[^] ó Ctrl[6]	va a la posición de superíndice
Ctrl[&] ó Ctrl[7]	va a la posición de sobreescritura
Ctrl[␣]	retorna de una posición especial (Control-Espacio)

Formas especiales de entrada usando la tecla Control. La segunda forma dada debería trabajar en cualquier teclado.

Esto ingresa un subíndice usando la tecla Control.

```
Expand[(1 + x^_1+n^_)^4]
```

$$1 + 4\,x_{1+n} + 6\,x_{1+n}^{2} + 4\,x_{1+n}^{3} + x_{1+n}^{4}$$

Tal como ^ y _ van a las posiciones de superíndice y subíndice, también & y = pueden usarse para ir directamente a las posiciones de superescritura y subescritura. Con la disposición de un teclado estándar de lengua inglesa & está directamente a la derecha de ^ mientras que = está directamente a la derecha de _.

combinación de teclas	*forma en que es mostrada*	*forma de expresión*
x^&_	$\bar{x}$	`OverBar[x]`
x^&:vec:	$\vec{x}$	`OverVector[x]`
x^&~	$\tilde{x}$	`OverTilde[x]`
x^&^	$\hat{x}$	`OverHat[x]`
x^&.	$\dot{x}$	`OverDot[x]`
x^&=	$\underline{x}$	`UnderBar[x]`

Algunas formas de ingresar ciertos adornos comunes usando la tecla Control.

He aquí $\bar{x}$.

```
x ^&_ ^_
```

$$\bar{x}$$

Usted puede usar $\bar{x}$ como variable.

```
Solve[a^2 == %, a]
```

$$\left\{\left\{a \to -\sqrt{\bar{x}}\right\}, \left\{a \to \sqrt{\bar{x}}\right\}\right\}$$

combinación de teclas	*forma en que es mostrada*	*forma de expresión*
x _y	x_y	`Subscript[x, y]`
x \+y	$\underset{y}{x}$	`Underscript [x, y]`
x \^y	x^y	`Superscript[x, y]` (interpretado como `Power[x, y]`)
x \&y	$\overset{y}{x}$	`Overscript [x, y]`
x \&_	$\bar{x}$	`OverBar[x, y]`

x \&\[RightVector]	$\vec{x}$	OverVector[x, y]
x \&~	$\tilde{x}$	OverTilde[x, y]
x \&^	$\hat{x}$	OverHat[x, y]
x \&.	$\dot{x}$	OverDot[x, y]
x \+_	$\underline{x}$	UnderBar[x, y]

Formas de ingresar adornos sin la tecla Control. Todas estas formas sólo pueden usarse dentro de \!\(... \).

11.7. Caracteres no ingleses

Si usted ingresa texto en lenguas que no sean el inglés, tendrá que usar varios caracteres acentuados adicionales y otros. Si su sistema informático esta establecido de forma apropiada, entonces usted podrá ingresar tales caracteres directamente usando teclas estándares de su teclado. Pero sino lo está es, *Mathematica* siempre proporciona un forma uniforme de manejar tales caracteres.

nombre completo	*alias*	*nombre completo*	*alias*
à \[AGrave]	:a`:	ø \[OSlash]	:o/:
å \[ARing]	:ao:	ö \[ODoubleDot]	:o":
ä \[AdoubleDot]	:a":	ù \[UGrave]	:u`:
ç \[CCedilla]	:c,:	ü \[UDoubleDot]	:u":
ć \[CHacek]	:cv:	ß \[SZ]	:sz:, :ss:
é \[EAcute]	:e':	Å \[CapitalARing]	:Ao:
è \[EGrave]	:e`:	Ä \[CapitalADoubleDot]	:A":
í \[IAcute]	:i':	Ö \[CapitalODoubleDot]	:O":
ñ \[NTilde]	:n~:	Ü \[CapitalUDoubleDot]	:U":
ò \[OGrave]	:o`:		

Algunos caracteres europeos comunes.

He aquí una función cuyo nombre involucra un caracter acentuado.

```
Lam\[EAcute][x, y]
```

```
Lamé[x, y]
```

Esta es otra forma de ingresar la misma entrada.

```
Lam:e':[x, y]
```

```
Lamé[x, y]
```

Usted debe comprender que no hay ningún estándar uniforme para teclados de computador en el mundo entero, y por consiguiente es inevitable que algunos detalles que han sido mencionados en este capítulo no puedan aplicarse a su teclado.

En particular, la identificación por ejemplo de CTRL[6] con CTRL[^] es válida sólo para teclados en los cuales ^ aparece como Shift-6. En otros teclados, *Mathematica* usa CTRL[6] para ir a una posición de superíndice, pero no necesariamente CTRL[^].

Independientemente de como esté establecido su teclado usted siempre puede usar paletas o ítems de menú para establecer superíndices y otros tipos de notación. Y asumiendo que tiene algún modo de ingresar caracteres como \, usted siempre puede ingresar entradas que usan nombres completos como `\[Infinity]` y formas textuales como `\(x\/y\)`.

11.8. Otras notaciones matemáticas

Mathematica soporta una amplia gama de notación matemática, aunque a menudo no le asigne un significado predefinido. Así, por ejemplo, usted puede ingresar una expresión como `x ⊕ y`, pero *Mathematica* inicialmente no hará ninguna asunción sobre lo que usted quiere decir con ⊕.

Mathematica sabe que ⊕ es un operador, pero no le asigna inicialmente ningún significado específico.

```
{17 ⊕ 5, 8 ⊕ 3}
```

```
{17⊕5, 8⊕3}
```

Esto da a *Mathematica* una definición de lo que hace el operador ⊕.

```
x_ ⊕ y_ := Mod[x + y, 2]
```

Ahora *Mathematica* puede evaluar operaciones con ⊕. Aquí tenemos una evaluación simbólica

```
a ⊕ b
```

```
Mod[a+b, 2]
```

He aquí evaluaciones numéricas.

```
{17 ⊕ 5, 8 ⊕ 3}
```

```
{0, 1}
```

	nombre completo	*alias*		*nombre completo*	*alias*
⊕	\[CirclePlus]	:c+:	⟶	\[LongRightArrow]	:-->:
⊗	\[CircleTimes]	:c*:	↔	\[LeftRightArrow]	:<->:
±	\[PlusMinus]	:+-:	↑	\[UpArrow]	
∧	\[Wedge]	:^:	⇌	\[Equilibrium]	:equi:
∨	\[Vee]	:v:	⊢	\[RightTee]	
≃	\[TildeEqual]	:~=:	⊃	\[Superset]	:sup:
≈	\[TildeTilde]	:~~:	⋂	\[SquareIntersection]	
~	\[Tilde]	:~:	∈	\[Element]	:elem:
∝	\[Proportional]	:prop:	∉	\[NotElement]	:!elem:
≡	\[Congruent]	:===:	∘	\[SmallCircle]	:sc:
≳	\[GreaterTilde]	:>~:	∴	\[Therefore]	
≫	\[GreaterGreater]		❘	\[VerticalSeparator]	:|:
≻	\[Succeeds]		❘	\[VerticalBar]	:|:
▷	\[RightTriangle]		\	\[Backslash]	:\:

Algunos de los operadores cuya entrada es soportada por *Mathematica*..

Mathematica asigna significado predefinido para ≥ y ⩾, pero no para ≳ ó ≫.

```
{3 ≥ 4, 3 ⩾ 4, 3 ≳ 4, 3 ≫ 4}
{False, False, 3 ≳ 4, 3 ≫ 4}
```

Hay algunas formas que se parecen a caracteres sobre un teclado estándar, pero que son interpretadas de un modo diferente por *Mathematica*. Así, por ejemplo, \[Backslash] ó :\: se muestran como \ pero no es interpretado de la misma manera que \ digitado directamente en el teclado.

Los caracteres \ y ∧ usados aquí son diferentes a \ y ^ que usted digita en el teclado.

```
{a :\: b, a :^: b}
{a \b, a∧b}
```

La mayor parte de operadores trabajan como ⊕ y van en medio de sus operandos. Pero algunos operadores pueden ir en otros sitios. Así, por ejemplo, :<: y :>: ó \[LeftAngleBracket] y \[RightAngleBracket] son efectivamente operadores que envuelven a su operando.

Los elementos del operador ángulo van alrededor de su operando.

```
\[LeftAngleBracket] 1 + x \[RightAngleBracket]
```

```
⟨1 + x⟩
```

nombre completo	*alias*	*nombre completo*	*alias*
ℓ \[ScriptL]	:scl:	Å \[Angstrom]	:Ang:
ℰ \[ScriptCapitalE]	:scE:	ℏ \[HBar]	:hb:
ℝ \[GothicCapitalR]	:goR:	£ \[Sterling]	
ℤ \[DoubleStruckCapitalZ]	:dsZ:	∠ \[Angle]	
ℵ \[Aleph]	:al:	• \[Bullet]	:bu:
ϕ \[EmptySet]	:es:	† \[Dagger]	:dg:
μ \[Micro]	:mi:	♮ \[Natural]	

Algunas letras adicionales y formas parecidas a letras.

Usted puede usar cualesquiera letras y formas parecidas a letras como nombres de símbolos.

```
{RØ, \[Angle]ABC}
```

```
{RØ, ∠ABC}
```

ϕ se asume como un símbolo, y tan solo es multiplicado por a y b.

```
a Ø b
```

```
a b Ø
```

También es posible generar todas las letras del alfabeto, incluyendo mayúsculas y minúsculas, en los tres primeros tipos que aparecen en la última tabla, esto es Script, Gothic y DoubleStruck.

```
{:scA:, :sca:, :goN:, :gon:, :dsR:, :dsr:}
```

```
{𝒜, 𝒶, 𝔑, 𝔫, ℝ, 𝕣}
```

11.9. Formas de entrada y salida

Los cuadernos en *Mathematica* le permiten dar entradas y obtener salidas en una variedad de formas diferentes. El front end proporciona comandos de menú para convertir celdas de una a otra forma.

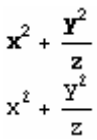
`InputForm`	una forma que puede ser digitada directamente usando caracteres en un teclado estándar
`OutputForm`	una forma para salida que sólo usa caracteres en un teclado estándar
`StandardForm`	una forma para entrada y salida que hace uso de caracteres especiales y posicionamiento
`TraditionalForm`	una forma principalmente para salida que imita todos los aspectos de notación tradicional matemática

Formas de entrada y salida.

La entrada de aquí trabaja en las ambas formas `InputForm` y `StandardForm`.

```
x^2 + y^2/z
```

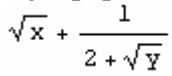
He aquí una versión del ingreso apropiado para `StandardForm`.



`InputForm` es la forma más general de entradas para *Mathematica*: esto trabaja si usted usa una interfase de cuaderno o una interfase a basada en texto.

Con una interfase de cuaderno, la salida es producida por defecto en `StandardForm`.

```
Sqrt[x] + 1/(2 + Sqrt[y])
```



Con una interfase basada en texto, en cambio es usada `OutputForm`.

```
Sqrt[x] + 1/(2 + Sqrt[y]) // OutputForm
```

```
                1
Sqrt[x] + ----------
          2 + Sqrt[y]
```

Con una interfase de cuaderno, la forma por defecto tanto para la entrada como para la salida es `StandardForm`.

La idea básica de `StandardForm` es proporcionar una representación exacta y elegante de expresiones en *Mathematica*, haciendo uso de caracteres especiales, posicionamientos bidimensionales, etcétera.

Entrada y salida son dadas aquí en `StandardForm`.

$$\int \frac{1}{(x^3+1)}\,dx$$

$$\frac{\text{ArcTan}\left[\frac{-1+2x}{\sqrt{3}}\right]}{\sqrt{3}} + \frac{1}{3}\text{Log}[1+x] - \frac{1}{6}\text{Log}[1-x+x^2]$$

Una característica importante de `StandardForm` es que cualquier salida usted obtiene en esta forma también puede usarla directamente como entrada.

$$\frac{\mathbf{ArcTan}\left[\frac{-1+2x}{\sqrt{3}}\right]}{\sqrt{3}} + \frac{\mathbf{Log[1+x]}}{3} - \frac{\mathbf{Log[1-x+x^2]}}{6}$$

$$\frac{\text{ArcTan}\left[\frac{-1+2x}{\sqrt{3}}\right]}{\sqrt{3}} + \frac{1}{3}\text{Log}[1+x] - \frac{1}{6}\text{Log}[1-x+x^2]$$

La naturaleza exacta de `StandardForm` le impide seguir todas las convenciones algo casuales de la notación matemática tradicional. *Mathematica* sin embargo también soporta `TraditionalForm`, que usa una amplia colección de reglas de interpretación bastante completa de notación matemática tradicional.

`TraditionalForm` nombres en minúsculas para funciones, y pone sus argumentos entre paréntesis y no entre corchetes.

$$\int \frac{1}{(x^3+1)}\,dx \text{ // TraditionalForm}$$

$$\frac{\tan^{-1}\left(\frac{2x-1}{\sqrt{3}}\right)}{\sqrt{3}} + \frac{1}{3}\log(x+1) - \frac{1}{6}\log(x^2-x+1)$$

He aquí algunas transformaciones hechas por `TraditionalForm`.

```
{Abs[x], ArcTan[x], BesselJ[0, x], Binomial[i, j]}
// TraditionalForm
```

$$\left\{|x|,\ \tan^{-1}(x),\ J_0(x),\ \binom{i}{j}\right\}$$

`TraditionalForm` es a menudo útil para generar salidas que pueden ser insertadas directamente en documentos que usen notación matemática tradicional. Pero usted debe entender que `TraditionalForm` se requiere principalmente para salidas: no tiene la clase de precisión que es necesaria para proporcionar una entrada confiable a *Mathematica*.

Así, por ejemplo, en `TraditionalForm`, `Ci(x)` es la representación tanto para `Ci[x]` como para `CosIntegral[x]`, así que si esta forma aparece sólo como entrada, *Mathematica* no tendrá ni idea de cuál de las dos interpretaciones es correcta.

En `StandardForm` estas tres expresiones son mostradas de un modo único e inequívoco.

```
{ Ci[1+x], CosIntegral[1+x], Ci(1+x) } //
StandardForm
```

```
{Ci[1 + x], CosIntegral[1 + x], Ci (1 + x)}
```

En `TraditionalForm`, sin embargo, las primeras dos son imposibles de distinguirse, y la tercera se diferencia sólo por la presencia de un espacio adicional.

```
{ Ci[1+x], CosIntegral[1+x], Ci(1+x) } //
TraditionalForm
```

$$\{\mathrm{Ci}(x+1),\ \mathrm{Ci}(x+1),\ \mathrm{Ci}\,(x+1)\}$$

Las ambigüedades de `TraditionalForm` lo hacen en general inadecuado para especificar entradas al núcleo de *Mathematica*. Pero al menos para casos suficientemente simples, *Mathematica* realmente incluye varias reglas heurísticas para tratar de interpretar expresiones `TraditionalForm` como entradas de *Mathematica*.

Se asume que las celdas previstas para dar entradas al núcleo por defecto contendrán expresiones `StandardForm`.

$$c\left(\sqrt{x}+\frac{1}{1}\right)+\Gamma(x)$$

```
c (1 + Sqrt[x]) + x Γ
```

Aquí al front end le fue dicho que la entrada sería dada específicamente en `TraditionalForm`. El corchete de la celda tiene una línea dentada para indicar las dificultades implicadas.

$$c\left(\sqrt{x}+1\right)+\Gamma(x)$$

```
c (1 + Sqrt[x]) + Gamma[x]
```

- La entrada es una copia o una edición simple de la salida anterior.
- La entrada ha sido convertida de `StandardForm`, quizás con ediciones simples.
- La entrada contiene información explícita oculta que da su interpretación.
- La entrada contiene sólo las notaciones más simples y familiares.

Algunas situaciones en que puede esperar que la entrada `TraditionalForm` trabaje.

Siempre que *Mathematica* genera una expresión en `TraditionalForm`, automáticamente inserta varias etiquetas ocultas de modo que más adelante la expresión pueda ser interpretada inequívocamente si se da como entrada. E incluso si usted corrige la expresión, las etiquetas a menudo serán dejadas suficientemente aceptables de modo que la interpretación inequívoca sea aún posible.

Esto genera una salida en `TraditionalForm`.

```
Exp[I Pi x] // TraditionalForm
```

$$e^{i\pi x}$$

Aquí le fue dicho a *Mathematica* que esperara una entrada `TraditionalForm`. La entrada fue copiada de la línea de salida anterior, y así contiene las etiquetas ocultas que aseguran la interpretación correcta.

```
e^(i π x) // StandardForm
```

$$e^{i\pi x}$$

Una edición simple a menudo no afecta las etiquetas ocultas.

```
e^(2 i π x) // StandardForm
e^(2 i π x)
```

Si usted ingresa una expresión `TraditionalForm` desde el principio, o la importa desde fuera de *Mathematica*, entonces *Mathematica* hará todo lo posible para averiguar lo que la expresión significa. Cuando hay ambigüedades, lo que típicamente hace es asumir que usted esta usando alguna notación común en aplicaciones de matemáticas elementales.

En una entrada `TraditionalForm`, esto es interpretado como una derivada.

```
∂y(x)/∂x // StandardForm
y'[x]
```

Esto es interpretado como una integral.

```
∫f(x) dx // StandardForm
∫f[x] dx
```

Esto es interpretado como un arco tangente.

```
tan^-1(x) // StandardForm
ArcTan[x]
```

Esto es interpretado como el cuadrado de una tangente.

```
tan^2(x) // StandardForm
Tan[x]^2
```

No hay una interpretación particular estándar tradicional para esto, *Mathematica* asume que es `1/Tan[x]^2`.

```
tan^-2(x) // StandardForm
Cot[x]^2
```

Usted debe comprender que `TraditionalForm` no proporciona ningún tipo de forma exacta o completa de especificar expresiones en *Mathematica*. Sin embargo,

para algunos objetivos elementales puede ser suficiente, en particular si usted usa algunos trucos adicionales.

- Use `x(y)` para funciones; `x (y)` para multiplicación.
- Use `:ee:` para la constante exponencial `E`.
- Use `:ii:` ó `:jj:` para la unidad imaginaria `I`.
- Use `:dd:` para operadores diferenciales en integrales y derivadas.

Algunos trucos para entradas `TraditionalForm`.

Con un espacio `f (1+x)` es interpretada como multiplicación. Sin un espacio `g(1+x)` es interpretada como una función.

$f\,(1+x)+g(1+x)$ // **StandardForm**

```
f (1+x) + g[1+x]
```

La `e` ordinaria es interpretada como un símbolo `e`. La "e exponencial" que se ingresa como `:ee:`, es interpretada como la constante exponencial.

$\{e^{3.7},\ e^{3.7}\}$ // **StandardForm**

```
{e^3.7, 40.4473}
```

11.10. Mezcla de texto y fórmulas

El modo más simple de mezclar texto y fórmulas en un cuaderno de *Mathematica* es poner cada clase de material en una celda separada. A veces, sin embargo, puede que usted quiera encajar una fórmula dentro de una celda de texto, o viceversa.

CTRL-(ó CTRL-9	comience ingresando una fórmula dentro del texto, o el texto dentro de una fórmula
CTRL-) ó CTRL-0	termine entrando una fórmula dentro del texto, o el texto dentro de una fórmula

Entrada de una fórmula dentro de texto, o viceversa.

He aquí un cuaderno con fórmulas encajados en una celda de texto.

Esta es una celda de texto, pero contiene fórmulas como $\int \frac{1}{x^3-1}\,dx$ ó

$-\frac{\log(x^2+x+1)}{6} - \frac{\tan^{-1}\left(\frac{2x+1}{\sqrt{3}}\right)}{\sqrt{3}} + \frac{\log(x-1)}{3}$. Las fórmulas fluyen con el texto.

Los cuadernos de *Mathematica* a menudo contienen tanto fórmulas requeridas para la evaluación actual por *Mathematica*, como texto que es requerido sólo para su lectura en forma pasiva.

Usted debe comprender, sin embargo, que hacer la tipografía detallada de fórmulas típicas tan buena como sea posible, *Mathematica* automáticamente hace cosas como insertar espacios alrededor de ciertos operadores. Pero estas clases de ajustes pueden ser potencialmente inadecuados si usted usa una notación muy diferente a la que *Mathematica* espera.

11.11. Creación de sus propias paletas

El front end de un cuaderno de *Mathematica* viene con una colección de paletas estándar. Pero también le permite crear sus propias paletas.

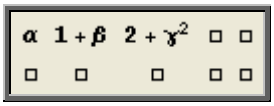
- Genere una paleta en blanco usando Create Table/Matrix/Palette del menú Input.
- Llene el contenido.
- Active la paleta usando Generate Palette from Selection del menú File.

Pasos básicos en la creación de una paleta.

Create Table/Matrix/Palette creará una paleta en blanco.

Usted entonces puede insertar lo que quiera en cada botón.

El ítem Generate Palette from Selection hace una paleta activa separada.



Haciendo clic sobre un botón de la paleta usted inserta el contenido de éste en su cuaderno.

Create Table/Matrix/Palette	genera una paleta en blanco
Generate Palette from Selection	hace una paleta separada activa
Generate Notebook from Palette	convierte una paleta en un cuaderno editable
Edit Button	corrige la escritura asociada a una paleta o botón

Ítems para generar paletas.

Cuando usted crea una paleta, puede usar los mismos mecanismos para añadir columnas y filas que usa cuando crea cualquier otra clase de tabla, matriz o rejilla. Así [Ctrl+,] añadirá una nueva columna de botones, y [Ctrl+↵] (Control-Enter) añadirá una nueva fila.

contenidos de botón	*acción*
X	substituye la selección actual por X
texto que contiene X■Y	substituye la selección actual S por XSY

Contenidos de botones.

En el caso más simple, cuando usted presiona un botón en una paleta lo que pasará es que el contenido del botón será insertado en su cuaderno, substituyendo lo que era su selección actual.

A veces sin embargo puede que usted simplemente no quiera sustituir su selección actual, sino más bien puede que quiera modificar la selección de algún modo. Como ejemplo, puede ser que desee aplicar una función como `Expand` a su selección actual. Puede hacer esto estableciendo un botón con el contenido `Expand[■]`. El ■ puede ingresarse como `:spl:` ó `\[SelectionPlaceholder]`. En general, ■ sirve como un guarda lugar para su selección actual. Cuando usted presiona un botón que contiene ■, el ■ es primero substituido por su selección actual, y sólo entonces el resultado es insertado en su cuaderno.

He aquí una celda en la cual la selección actual es parte de una expresión.

Presionando un botón que contiene `Expand[■]`, `Expand` envuelve la selección actual

$$1 + (1 + x)^4 + \text{Expand}\left[(2 + y)^3\right]$$

Mathematica le permite asociar cualquier acción que usted quiere con un botón. Puede establecer algunas acciones comunes usando el menú de Edit Button, seleccionando un solo botón o una paleta entera.

`Paste`	pega el contenido del botón (acción por defecto)
`Evaluate`	pega y luego evalúa en el lugar que ha sido pegado
`EvaluateCell`	pega y luego evalúa toda la celda
`CopyEvaluate`	copia la selección actual en una nueva celda, luego pega y evalúa en el lugar
`CopyEvaluateCell`	copia la selección actual en una nueva celda, luego pega y evalúa la celda completa

Acciones típicas para botones.

Con la acción de botón por defecto `Paste`, al presionar el botón se modifica el contenido de una celda, pero no se hace ninguna evaluación. Escogiendo otras acciones de botón, sin embargo, usted puede decir a *Mathematica* que realice una evaluación siempre que presiona el botón.

Con la acción de botón `Evaluate` el resultado de esta evaluación es sobrescrito sobre su selección actual. Esto es útil si usted quiere generar un botón que modifique las partes de una expresión en el lugar.

La acción de botón `Evaluate` realiza la evaluación sólo sobre lo que fue pegado en su celda actual. La acción de botón `EvaluateCell`, por otra parte, realiza la evaluación sobre la celda completa, generando una nueva celda para mostrar el resultado.

He aquí una expresión con una parte seleccionada.

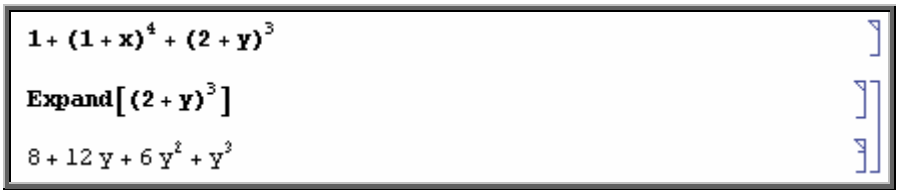
$$1 + (1 + x)^4 + (2 + y)^3$$

Esto muestra el resultado al presionar un botón que contiene `Expand[■]` con una acción de botón `EvaluateCell`.

A veces es útil extraer la selección actual de una célula, y luego operar esta selección en una nueva celda. Usted puede hacer esto utilizando las acciones de botón `CopyEvaluate` y `CopyEvaluateCell`.

He aquí una expresión con una parte seleccionada.

Un botón con una acción de botón `CopyEvaluateCell` copia la selección actual en una nueva celda, luego pega el contenido del botón, y luego realiza una evaluación, poniendo el resultado en una nueva celda.



Create Table/Matrix/Palette	genera una paleta en blanco
Create Button	genera un botón solo en una paleta
Generate Palette from Selection	crea una ventana aparte
Cell Active	activa botones dentro de una celda en un cuaderno

Formas de crear elementos activos en el front end.

Mathematica le permite establecer un amplio rango de elementos activos en el front end de cuaderno. En el caso más común, usted tiene una paleta que consiste en una serie de botones en una ventana aparte. Pero también puede tener arrays de botones, o hasta botones solos, dentro de una celda de un cuaderno común.

Además, usted puede hacer que un botón ejecute cualquier acción que usted quiera–realizar cálculos en el kernel de *Mathematica*, o cambiar la configuración de cuadernos en el front end.

11.12. Creación de Hipervínculos

Create Hyperlink	crea un hipervínculo en el objeto seleccionado

Ítem para generar hipervículos.

Un hipervínculo es una clase especial de botón que brinca a otra parte de un cuaderno cuando es presionado. Típicamente los hipervínculo se indican en *Mathematica* con texto azul o subrayado.

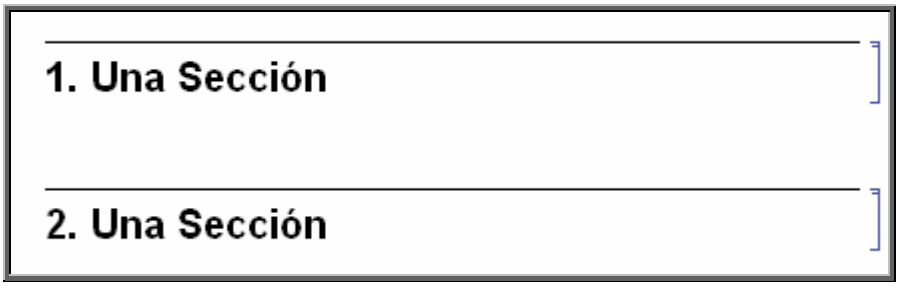
Para establecer un hipervínculo, solamente seleccione el texto u otro objeto al que le quiere hacer un hipervínculo. Entonces escoja el ítem Create Hyperlink y complete la especificación de donde usted quiere que esté el destino del hipervínculo.

11.13. Numeración automática

- Escoja un estilo de celda tal como `NumberedEquation`.
- Use el menú `Create Automatic Numbering Object`, con un nombre de conteo tal como `Section`

Dos formas de generar numeración automática en un cuaderno de *Mathematica*.

La entrada para cada celda aquí es exactamente la misma, pero las celdas contienen un elemento que muestra un número que va aumentando progresivamente cuando avanzamos a través del cuaderno.



Esta celda está en un estilo `NumberedEquation`.

$$\int \frac{x}{x+1}\,dx \tag{1}$$

$$\int \frac{\text{Sin}[x]}{x+1}\,dx \tag{2}$$

$$\int \frac{\text{Log}[x]+\text{Exp}[x]}{x+1}\,dx \tag{3}$$

12. Archivos y operaciones externas

12.1. Lectura y Escritura de archivos en *Mathematica*

Usted puede usar archivos en su sistema informático para almacenar definiciones y resultados de *Mathematica*. El enfoque más general es almacenar todo como texto simple que es apropiado como entrada para *Mathematica*. Con este enfoque, una versión de *Mathematica* corriendo en un sistema informático produce archivos que pueden ser leídos por una versión que corre en cualquier sistema informático. Además, tales archivos pueden ser manipulados por otros programas estándar, tales como editores de textos.

<< *name*	lee en un archivo de entrada de *Mathematica*
expr >> *name*	produce *expr* como un archivo texto simple
expr >>> *name*	añade *expr* un archivo
! ! *name*	muestra el contenido de un archivo de texto simple

Lectura y escritura de archivos.

Esto expande $(x+y)^3$ y produce el resultado en un arhivo llamado `tmp`.

```
Expand[ (x + y)^3 ] >> tmp
```

He aquí los contenidos de `tmp`. Ellos pueden ser usados directamente como una entrada para *Mathematica*.

```
!!tmp
x^3 + 3*x^2*y + 3*x*y^2 + y^3
```

Esto lee en `tmp`, evaluando la entrada de *Mathematica* que contiene.

```
<<tmp
```

$$x^3 + 3x^2y + 3xy^2 + y^3$$

Si usted esta familiarizado con los sistemas operativos, reconocerá la similitud de los operadores de redireccionamiento de *Mathematica* `>>`, `>>>` y `<<` con los operadores de command-line `>`, `>>` y `<`.

Los operadores de redireccionamiento `>>` y `>>>` son convenientes para almacenar resultados que usted obtiene de *Mathematica*. La función `Save["`*name*`", `*f*`, `*g*`, ...]` le permite guardar definiciones para variables y funciones.

`Save["`*dir*`", `*f*`, `*g*`, ...]`	guarde definiciones de variables o funciones en un archivo

Guardando definiciones en archivos de texto simples.

He aquí una definición para una función `f`.

```
f[x_] := x^2 + c
```

Esto le da a `c` el valor de `17`.

```
c = 17
```

```
17
```

Esto guarda la definición de `f` en el archivo `ftmp`.

```
Save["ftmp", f]
```

Mathematica automáticamente guarda la definición actual de `f`, y la definición de `c` de la cual depende.

```
!!ftmp
```

```
f[x_] := x^2 + c

c = 17
```

Esto borra las definiciones de `f` y `c`.

```
Clear[f, c]
```

Usted puede rehabilitar las definiciones que guardó simplemente leyendo en el archivo `ftmp`.

```
Exp[I Pi x] // TraditionalForm
17
```

file`.m`	archivo de expresión de *Mathematica* en formato de texto simple
file`.nb`	archivo de cuaderno de *Mathematica*
file`.mx`	definiciones de *Mathematica* en formato `DumpSave`

Nombres típicos de archivos de *Mathematica*.

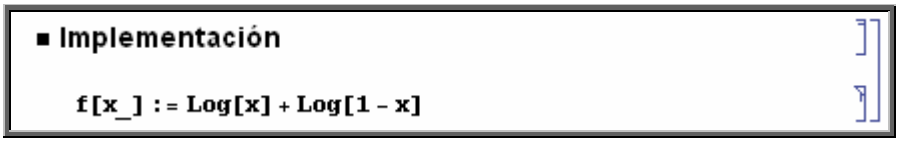
Si usted usa una interfase de cuaderno en *Mathematica*, entonces el front end de *Mathematica* le permite guardar cuadernos completos, incluyendo no sólo entradas y salidas de *Mathematica*, sino también texto, gráficos y otro material.

Es convencional dar nombres a los archivos de cuaderno de *Mathematica* que terminen en `.nb`, y la mayor parte de versiones de *Mathematica* hacen cumplir esta convención.

Cuando usted abre un cuaderno en el front end de *Mathematica*, *Mathematica* inmediatamente le mostrará el contenido del cuaderno, pero normalmente no enviará nada de este contenido al núcleo para su evaluación hasta que usted lo solicite explícitamente.

Dentro de un cuaderno de *Mathematica*, sin embargo, usted puede usar el menú Cell en el front end para identificar ciertas celdas como *celdas de inicialización*, y si usted hace esto, entonces el contenido de estas celdas automáticamente será evaluado cada vez que abra el cuaderno.

La I en el corchete de la celda indica que la segunda celda es una celda de inicialización que será evaluada cada vez cuaderno sea abierto.



A veces es conveniente mantener el material de *Mathematica* tanto en un cuaderno que contiene el texto explicativo, como en un paquete que contiene sólo la materia prima de las definiciones de *Mathematica*. Usted puede hacer esto poniendo las definiciones de *Mathematica* en celdas de inicialización en el cuaderno. Cuando

usted guarde el cuaderno, el front end le permitirá guardar un archivo asociado `.m` que sólo contiene la materia prima de las definiciones de *Mathematica*.

12.2. Localización y manipulación de archivos

Aunque los detalles como son llamados y organizados los archivos se diferencian de un sistema informático a otro, *Mathematica* proporciona algunos mecanismos bastante generales para localizar y manejar archivos.

Mathematica asume que los archivos en su sistema informático son organizados en una colección de *directorios*. En cualquier punto, usted tiene un *directorio de trabajo actual*. Usted siempre puede referirse a archivos en este directorio con tan solo dar sus nombres.

`Directory[ ]`	da su directorio de trabajo actual
`SetDirectory["`*dir*`"]`	fija su directorio de trabajo actual
`FileNames[ ]`	lista los archivos en su directorio de trabajo actual
`FileNames["`*form*`"]`	lista los archivos cuyos nombres se ajustan a una cierta forma
`<<`*name*	lee en un archivo con el nombre especificado
`<<`*context*`` ` ``	lee en un archivo correspondiente al contexto especificado
`CopyFile["`$file_1$`", "`$file_2$`"]`	copia $file_1$ a $file_2$
`DeleteFile["`*file*`"]`	borra un archivo

Funciones para localizar y manipular archivos.

Este es un directorio de trabajo actual. La forma que tiene se diferencia de un sistema informático a otro.

```
Directory[ ]
C:\Archivos de programa\Wolfram Research\
Mathematica\5.1
```

Esto resetea el directorio de trabajo actual.

```
SetDirectory["Ejemplos"]
C:\Archivos de programa\Wolfram Research\
Mathematica\5.1\Examples
```

Esto da una lista de todos los archivos en su directorio de trabajo actual cuyos nombres se ajustan a la forma `Test*.m`.

```
FileNames["Prueba*.m"]
{Prueba1.m,Prueba2.m,PruebaFinal.m}
```

Aunque usted por lo general quiera crear archivos sólo en su directorio de trabajo actual, a menudo tiene que leer en archivos de otros directorios. Por consiguiente, cuando pide a *Mathematica* leer en un archivo con un nombre particular, *Mathematica* automáticamente busca una lista de directorios (especificado por el valor de la variable de búsqueda de camino `$Path`) hasta que, en caso de que exista, encuentra un archivo con aquel nombre.

Una cuestión en archivos que se manejan en *Mathematica* es que la forma de los archivos y nombres de directorio varía entre sistemas informáticos. Esto quiere decir por ejemplo que los nombres de los archivos que contienen paquetes estándares de *Mathematica* pueden ser bastante diferentes en sistemas diferentes. Con una secuencia de convenciones, es sin embargo posible leer en un paquete estándar de *Mathematica* con el mismo comando en todos los sistemas. La forma en que esto trabaja es que cada paquete de programas define un llamado contexto de *Mathematica*, de la forma *name`name`* . En cada sistema, todos los archivos se nombran en correspondencia con los contextos que definen. Entonces cuando usted usa el comando `<<`*name`name`* *Mathematica* automáticamente traduce el nombre de contexto al nombre de archivo apropiado para su sistema informático particular.

`FindList["`*file*`", "`*text*`"]`	da una lista de todas las líneas en un archivo que contienen el texto especificado
`FindList[ FileNames[ ], "`*text*`"]`	busca en todos los archivos en su directorio corriente

Nombres típicos de archivos de *Mathematica*.

Esto busca todas las líneas en el archivo `Prueba1` que contienen `ChordLengths`.

```
FindList["Prueba1.m", "Rotation"]
{(* :Keywords: Traslation, Scaling, Reflection,
   Rotation *),
 Rotation::usage = "Rotation[].",
 Rotation[p_, t_, q_:{0, 0}] :=,
 Rotation[ptos : {{_, _} ..}, t_, q_:{0, 0}] :=,
 Table[Rotation[ptos[[i]], t, q], {i,
   Length[ptos]}]}
```

12.3. Importación y exportación de datos

`Import["`*file*`", "Table"]`	importa una tabla de datos desde un archivo
`Export["`*file*`",` *list*`, "Table"]`	exporta *list* a un archivo como una tabla de datos

Importación y exportación de datos tabulares.

Esto exporta un array de números al archivo `out.dat`.

```
Export["out.dat", {{5.7, 4.3}, {-1.2, 7.8}}]
out.dat
```

He aquí el contenido del archivo `out.dat`.

```
!!out.dat
5.7   4.299999999999999
-1.2   7.799999999999999
```

Esto importa el contenido de `out.dat` como una tabla de datos.

```
Import["out.dat", "Table"]
{5.7,4.3},{-1.2,7.8}}
```

`Import["`*file*`", "Table"]` manejará muchas clases de datos tabulares, automáticamente deduciendo los detalles del formato siempre que es posible. `Export["`*file*`",` *list*`, "Table"]` escribe datos separados por espacios, con números dados en forma parecida a C o Fortran, como en `2.3E5` etcétera.

`Import["`*name.ext*`"]`	importa datos asumiendo un formato deducido del nombre del archivo
`Export["`*name.ext*`",` *expr*`]`	importa datos en un formato deducido del nombre del archivo

Importación y exportación de datos generales.

formatos de tabla	`"CSV"`, `"TSV"`, `"XLS"`
formatos de matriz	`"HarwellBoeing"`, `"MAT"`, `"MTX"`
formatos de datos especializados	`"DIF"`, `"FITS"`, `"HDF5"`, `"MPS"`, `"SDTS"`, etc.

Algunos formatos comunes para datos tabulares.

`Import` y `Export` pueden manejar datos no sólo tabulares, sino también datos correspondientes a gráficos, sonidos, expresiones y aún documentos completos `Import` y `Export` a menudo pueden deducir el formato apropiado para datos con simplemente ver la extensión del nombre del archivo para el archivo en el cual los datos están siendo almacenados.

Note que usted también puede usar `Import` y `Export` para manipular archivos de datos binarios.

Esto importa un gráfico en formato GIF.

```
Import["girafa.gif"]
```

```
- Graphics -
```

Esto muestra el gráfico.

```
Show[%]
```

```
- Graphics -
```

`$ImportFormats`	importa formatos soportados por su sistema
`$ExportFormats`	importa formatos soportados por su sistema

Encuentra la lista completa de formatos de importación y exportación soportados.

12.4. Exportación de gráficos y sonidos

Mathematica le permite exportar gráficos y sonidos en una amplia variedad de formatos. Si usted usa el front end de un cuaderno de *Mathematica*, entonces usted solamente puede copiar y pegar gráficos y sonidos directamente en otros programas usando el mecanismo estándar disponible en su sistema informático.

`Export["`*name.ext*`",` *graphics*`]`	exporta gráficos a un archivo en un formato deducido del nombre del archivo
`Export["`*file*`",` *graphics*`,` `"`*format*`"]`	exporta gráficos en el formato especificado
`Export["!`*command*`",` *graphics*`,` `"`*format*`"]`	exporta gráficos a un comando externo

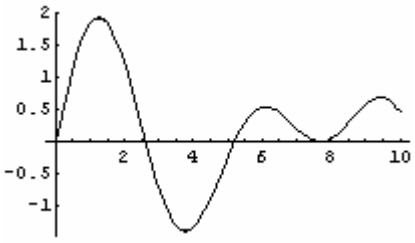
Exportación de gráficos y sonidos en *Mathematica*.

formatos de gráficos	`"EPS"`, `"TIFF"`, `"GIF"`, `"JPEG"`, `"PNG"`, `"PCX"`, `"PDF"`, `"SVG"`, etc.
formatos de sonido	`"SND"`, `"WAV"`, `"AIFF"`, `"AU"`, etc.

Algunos formatos comunes para gráficos y sonidos.

Esto traza un gráfico.

```
Plot[Sin[x] + Sin[Sqrt[2] x], {x, 0, 10}]
```




```
- Graphics -
```

Esto exporta el gráfico a un archivo en formato PostScript.

```
Export["sinplot.eps", %]
sinplot.eps
```

12.5. Exportación de fórmulas de cuadernos

He aquí una celda conteniendo una fórmula.

$$-\frac{\text{ArcTan}\left[\frac{1+2x}{\sqrt{3}}\right]}{\sqrt{3}} + \frac{1}{3}\text{Log}[-1+x] - \frac{1}{6}\text{Log}[1+x+x^2]$$

Esto es lo que usted obtiene si copia la fórmula y la pega en un programa externo basado en texto.

```
\!\(-\(ArcTan[\(1 + 2 x\)\/\@3]\/\@3\) + Log[-1 + x]\/3
  - Log[1 + x + x\^2]\/6\)
```

Pegando el texto nuevamente en un cuaderno inmediatamente reproduce la fórmula original.

$$-\frac{\text{ArcTan}\left[\frac{1+2x}{\sqrt{3}}\right]}{\sqrt{3}} + \frac{1}{3}\text{Log}[-1+x] - \frac{1}{6}\text{Log}[1+x+x^2]$$

Mathematica le permite exportar fórmulas tanto textual como visualmente. Usted puede usar la `Export` para decir a *Mathematica* que escriba una representación visual de una fórmula en un archivo.

`Export["`*file.eps*`", ToBoxes[`*expr*`]]`	guarda la forma visual de *expr* en un archivo en formato EPS
`Export["file", ToBoxes[`*expr*`], "`*format*`"]`	guarda la forma visual de *expr* el formato especificado

Exportación de expresiones en forma visual.

12.6. Generación e Importación de TeX

Los cuadernos de *Mathematica* proporcionan un ambiente sofisticado para crear documentos técnicos. Pero en particular si usted quiere combinar su trabajo con material existente en TeX, puede encontrar conveniente usar `TeXForm` para convertir expresiones de *Mathematica* en forma conveniente de entrada para TeX.

`TeXForm[`*expr*`]`	imprime *expr* en forma de entrada para TeX

Salidas de *Mathematica* para TeX.

He aquí una expresión, impresa en forma estándar de *Mathematica*.

```
(x + y)^2 / Sqrt[x y]
```

$$\frac{(x+y)^2}{\sqrt{x\,y}}$$

He aquí la expresión en forma de entrada para TeX.

```
TeXForm[%]
\frac{(x+y)^2}{\sqrt{x y}}
```

`ToExpression["`*input*`", TeXForm]`	convierte una entrada para TeX a *Mathematica*

Conversión de cadenas TeX a *Mathematica*.

Esto convierte una cadena TeX a *Mathematica*. Observe los dobles \ necesarios en la cadena.

```
ToExpression["\\sqrt{x y}", TeXForm]
```

$$\sqrt{x\,y}$$

Además de la capacidad de convertir expresiones individuales a TeX, *Mathematica* también proporciona capacidades para traducir cuadernos completos. Estas capacidades por lo general pueden accederse desde el menú Save As Special en el front end de un cuaderno, donde pueden fijarse varias opciones.

12.7. Cambio de material con la Web

`HTMLSave["`*file*`.html"]`	guarda su cuaderno actual completo en forma HTML
`HTMLSave["`*file*`.html", "`*source*`.nb"]`	guarda una versión HTML del cuaderno *source*`.nb`

Conversión de cuadernos a HTML.

`HTMLSave` tiene muchas opciones que le permiten especificar como deberían ser convertidos los cuadernos para navegadores de Web con capacidades diferentes. Usted puede encontrar detalles escribiendo HTMLSave en el Help Browser de *Mathematica*.

`MathMLForm[`*expr*`]`	imprime *expr* en forma MathML
`MathMLForm[ StandardForm[`*expr*`]]`	use `StandardForm` antes que la notación matemática tradicional
`ToExpression["`*string*`", MathMLForm]`	interpreta una cadena de MathML como entrada de *Mathematica*

Conversión a y desde MathML.

He aquí una expresión impresa en forma MathML.

```
MathMLForm[x^2/z]
<math>
<mfrac>
  <msup>
    <mi>x</mi>
    <mn>2</mn>
  </msup>
  <mi>z</mi>
</mfrac>
</math>
```

Si usted pega MathML en un cuaderno de *Mathematica*, *Mathematica* automáticamente tratará de convertirlo en entrada de *Mathematica*. Usted puede copiar una expresión de un cuaderno como MathML utilizando el menú Copy As en el front end de un cuaderno.

`Export["`*file*`.xml",` *expr*`]`	exporta en formato XML
`Import["`*file*`.xml"]`	importa desde XML
`ImportString[ "`*string*`", "XML"]`	importa datos de una cadena de XML

Importación y exportación XML.

Similar a las expresiones de *Mathematica*, XML es un formato general para representar datos. *Mathematica* automáticamente convierte ciertos tipos de expresiones a y desde tipos específicos de XML. MathML es un ejemplo. Otros ejemplos incluyen NotebookML para expresiones de cuaderno, y SVG para gráficos.

Si usted pide a *Mathematica* que importe una parte genérica de XML, esto producirá una expresión *SymbolicXML*. Cada elemento XML de la forma <*elem attr*='*val*'>*data*</*elem*> es traducido a una expresión SymbolicXML de *Mathematica* de la forma `XMLElement["`*elem*`",{"`*attr*`"->"`*val*`"},{`*datos*`}]`. Una vez que usted ha importado un parte de XML como SymbolicXML, puede usar las potentes capacidades de programación simbólica de *Mathematica* para manipular la expresión que obtiene. Entonces puede usar la `Export` para exportar el resultado en la forma de XML.

Esto genera una expresión SymbolicXML, con un `XMLElement` representando el elemento `a` en la cadena XML.

```
ImportString["<a aa='va'>s</a>", "XML"]
XMLObject[Document][{},
  XMLElement[a,{aa→va},{s}],{}]
```

Ahora hay dos niveles anidados en SymbolicXML.

```
ImportString[ "<a><b bb='1'>ss</b><b
bb='2'>ss</b></a>", "XML"]
XMLObject[Document][{},
  XMLElement[
    a,{},{XMLElement[b,{bb→1},{ss}],
      XMLElement[b,{bb→2},{ss}]}],{}]
```

Esto hace una transformación simple en SymbolicXML.

```
%/."ss" -> XMLElement["c",{},{"xx"}]
XMLObject[Document][{},
  XMLElement[a,{},{XMLElement[b,{bb→1},
    {XMLElement[c,{},{xx}]}],XMLElement[b,
    {bb→2},{XMLElement[c,{},{xx}]}]}],{}]
```

Esto muestra el resultado como una cadena XML.

```
ExportString[%, "XML"]
<a>
 <b bb='1'>
  <c>xx</c>
 </b>
 <b bb='2'>
  <c>xx</c>
 </b>
</a>
```

`Import["http://`*url*`", ...]`	importación de datos de un website
`Import["ftp://`*url*`", ...]`	importación de datos de un servidor FTP

Importación de datos de fuentes Web.

12.8. Generación de expresiones C y Fortran

Si usted tiene programas con destino especial escritos en C o Fortran, puede ser que quiera tomar fórmulas que ha generado en *Mathematica* e insertarlas en el código original de sus programas. *Mathematica* le permite convertir expresiones matemáticas en expresiones C y Fortran.

`CForm[`*expr*`]`	escribe *expr* para que pueda ser usada en un programa en C
`FortranForm[`*expr*`]`	escribe *expr* para Fortran

Salidas de *Mathematica* para lenguajes de programación.

He aquí una expresión, escrita en forma estándar de *Mathematica*.

```
Expand[(1 + x + y)^2]
```

$$1 + 2\,x + x^2 + 2\,y + 2\,x\,y + y^2$$

He aquí la expresión en forma Fortran.

```
FortranForm[%]
```

```
1 + 2*x + x**2 + 2*y + 2*x*y + y**2
```

He aquí la misma expresión en forma C. Las macros para objetos como `Power` son definidas en el archivo `mdefs.h` que viene con la mayoría de las versiones de *Mathematica*.

```
CForm[%]
```

```
1 + 2*x + Power(x,2) + 2*y + 2*x*y + Power(y,2)
```

Usted debe comprender que hay muchas diferencias entre *Mathematica* y C o Fortran. Por consiguiente, las expresiones que usted traduce puede que no trabajen exactamente del mismo modo que en *Mathematica*. Además, hay tantas diferencias en construcciones de programación que no se hace ninguna tentativa de traducir éstos automáticamente.

`Compile[x,` *expr*`]`	compila una expresión en eficiente código interno

Una forma de compilar expresiones en *Mathematica*.

Una de las motivaciones comunes para convertir expresiones de *Mathematica* en C o Fortran es tratar de hacerlas más rápidas de evaluar numéricamente. Pero la razón más importante por la cual C y Fortran son potencialmente más eficientes que

Mathematica es que en estos lenguajes uno siempre especifica que tipo de cada variable usará–entera, real, array, etc.

La función `Compile` de *Mathematica* hace tales asunciones dentro de *Mathematica*, y genera un código interno sumamente eficiente. Por lo general estas corridas de código no son más lentas que las acostumbradas en C o Fortran.

12.9. Empalmar salidas de *Mathematica* con archivos externos

Si usted quiere hacer uso de una salida de *Mathematica* en un archivo externo como un programa o documento, a menudo encontrará útil "empalmar" la salida automáticamente en el archivo.

`Splice["`*file*`.m`*x*`"]`	empalma la salida de *Mathematica* en un archivo externo llamado *file*`.m`*x*, pone los resultados en el archivo *file*`.`*x*
`Splice["`*infile*`", "`*outfile*`"]`	empalma la salida de *Mathematica* en *infile*, enviando la salida a *outfile*

Empalmes de salida de *Mathematica* en archivos.

La idea básica es establecer las definiciones que usted necesita en una sesión particular de *Mathematica*, luego correr `Splice` para usar las definiciones que ha hecho para producir la salida apropiada a insertar en los archivos externos.

```
#include "mdefs.h"

double f(x)
double x;
{
double y;

y = <* Integrate[Sin[x]^5, x] *> ;

return(2*y - 1) ;
}
```

Un programa simple en C conteniendo una fórmula de *Mathematica*

```
#include "mdefs.h"

double f(x)
double x;
{
double y;

y = -5*Cos(x)/8 + 5*Cos(3*x)/48 - Cos(5*x)/80 ;

return(2*y - 1) ;
}
```

El programa anterior en C procesado con `Splice`

Bibliografía

Wolfram Research. *The Mathematica Book*.