VARIABLES Y FUNCIONES

Las variables comienzan con letras y también pueden contener números:
(Es mejor comenzar con letras en minúsculas, reservando las mayúsculas para objetos incorporados)

In[477]:=
```
a1 / 2
```
Out[477]=

$$\frac{a1}{2}$$

Un espacio entre dos variables o números indica multiplicación:
(En otras palabras, “a b” es igual a a por b, mientras que “ab” corresponde a la variable ab)

In[478]:=
```
a b + 5 x x
```
Out[478]=

$$a\ b + 5\ x^2$$

Use /. y → para realizar sustituciones en una expresión:
(La “regla” → puede ser escrita como ->)

In[479]:=
```
1 + 2 x /. x → 2
```
Out[479]=

5

In[480]:=
```
Clear[x]
```
borra

También podemos hacerlo:

In[481]:=
```
f[x_] := 1 + 2 x;
f[2]
```
Out[482]=

5

ÁLGEBRA

Puedes factorizar o expandir expresiones algebraicas:
(Utiliza CTRL+6 para exponentes de composición tipográfica.)

In[483]:=
```
Factor[x^2 + 2 x + 1]
```
factoriza

Out[483]=

$$(1 + x)^2$$

Wolfram Language utiliza == (dos signos de igual) para probar equidad:

In[484]:=
```
2 + 2 == 4
```
Out[484]=
```
True
```

Los comandos como Solve encuentran soluciones exactas a ecuaciones:

In[485]:=
```
Solve[x^2 + 5 x - 6 == 0, x]
```
resuelve

Out[485]=
```
{{x → -6}, {x → 1}}
```

Para resultados aproximados, usa NSolve:

In[486]:=
```
NSolve[7 x^2 + 3 x - 5 == 0, x]
```
resuelve numéricamente

Out[486]=
```
{{x → -1.08618}, {x → 0.657611}}
```

Pasa un sistema de ecuaciones como una lista:

In[487]:=
```
Solve[{x^2 + 5 == y, 7 x - 5 == y}, {x, y}]
```
resuelve

Out[487]=
```
{{x → 2, y → 9}, {x → 5, y → 30}}
```

Encuentra las raíces de una ecuación:
(|| es el símbolo para Or.)

In[488]:=
```
Roots[x^2 + 3 x - 4 == 0, x]
```
raíces

Out[488]=
```
x == -4 || x == 1
```

Si un polinomio no es fácilmente factorizable, los resultados aproximados pueden ser más útiles:

In[489]:=
```
NRoots[360 + 234 x - 1051 x^2 + 11 x^3 + 304 x^4 - 20 x^5 == 0, x]
```
raíces aproximadas

Out[489]=
```
x == -1.79025 || x == -0.498678 || x == 0.797819 || x == 1.68398 || x == 15.0071
```

El comando Reduce reduce un conjunto de desigualdades a una forma simple:
(Escribe <= para el símbolo ≤.)

In[490]:=
```
Reduce[{0 < x < 2, 1 ≤ x ≤ 4}, x]
```
reduce

Out[490]=
```
1 ≤ x < 2
```

La forma reducida puede incluir múltiples intervalos:

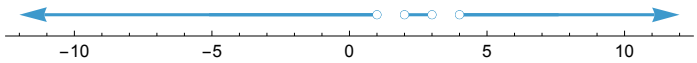
In[491]:=
```
Reduce[(x - 1) (x - 2) (x - 3) (x - 4) > 0, x]
```
reduce

Out[491]=
```
x < 1 || 2 < x < 3 || x > 4
```

NumberLinePlot es una forma útil de visualizar estos resultados:

In[492]:=
```
NumberLinePlot[x < 1 || 2 < x < 3 || x > 4, {x, -10, 10}]
```
representación de línea numérica

Out[492]=

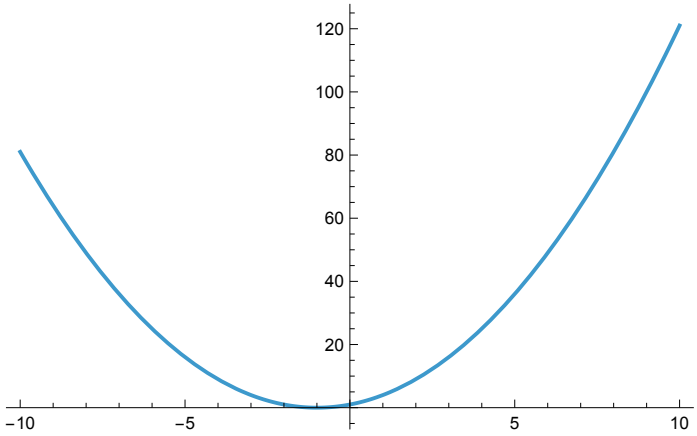


GRÁFICOS EN 2D

Genera un gráfico en 2D de una función de polinomio:
(La notación de intervalo de {x,min,max} define el dominio.)

In[493]:=
```
Plot[x^2 + 2 x + 1, {x, -10, 10}]
```
representación gráfica

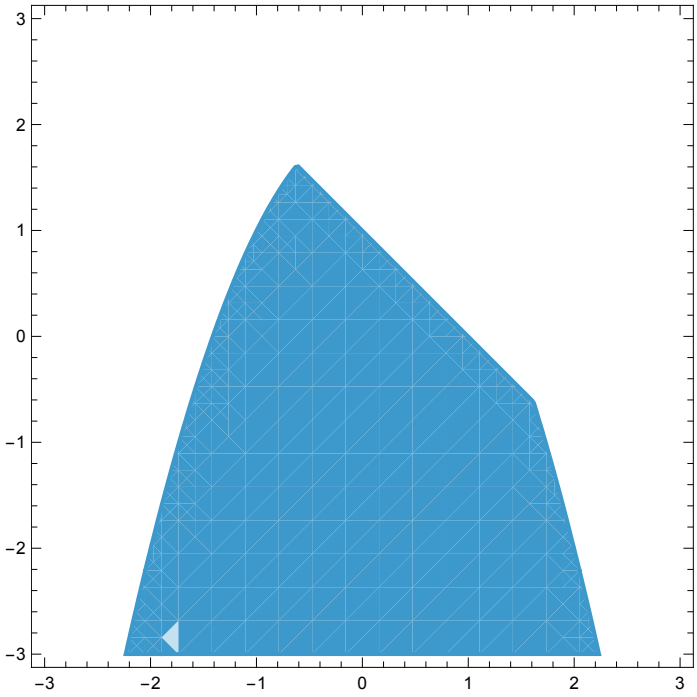
Out[493]=



O representa gráficamente una región en 2D para un conjunto de desigualdades:
(&& es el símbolo para And.)

In[494]:=
```
RegionPlot[Reduce[{x^2 + y < 2 && x + y < 1}], {x, -3, 3}, {y, -3, 3}]
```
representació··· reduce

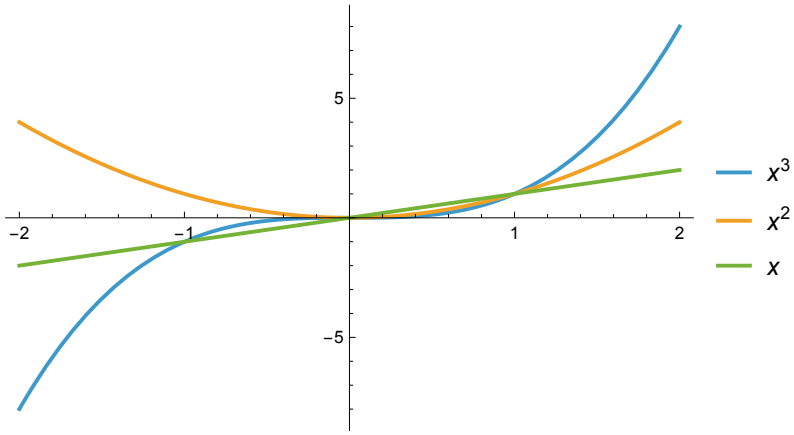
Out[494]=



Existen muchas opciones muy útiles para personalizar las visualizaciones, tales como agregar leyendas:

In[495]:=
```
Plot[{x^3, x^2, x}, {x, -2, 2}, PlotLegends → "Expressions"]
```
representación gráfica leyendas de representación

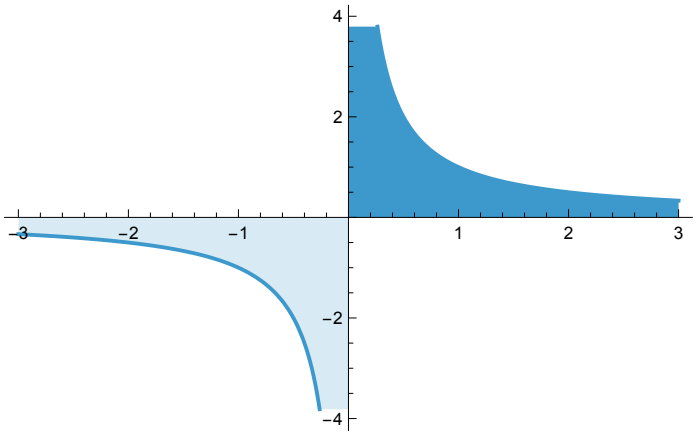
Out[495]=



O rellenar un gráfico para visualizar el área bajo una curva:

In[496]:=
```
Plot[1 / x, {x, -3, 3}, Filling → Axis]
```
representación gráfica relleno eje

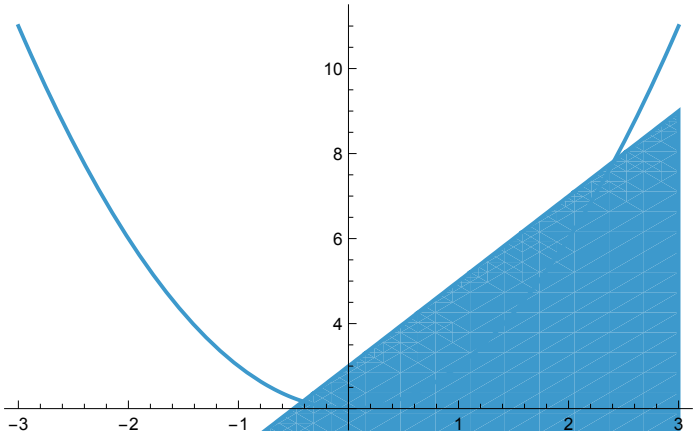
Out[496]=



Combina distintos tipos de representaciones gráficas con **Show**:

In[497]:=
```
Show[{Plot[x^2 + 2, {x, -3, 3}], RegionPlot[2 x > y - 3, {x, -3, 3}, {y, 0, 9}]}]
```
muestra representación gráfica representación gráfica de una región

Out[497]=



TRIGONOMETRÍA

Para funciones trigonométricas básicas, usa abreviaturas estándar (con la primera letra mayúscula):

In[498]:=
```
Sin[x] / Cos[x] == Tan[x]
```
seno coseno tangente

Out[498]=
```
True
```

Agrega “Arc” para la función inversa:

In[499]:=
```
ArcTan[1]
```
arco tangente

Out[499]=
$$\frac{\pi}{4}$$

Usa Pi cuando trabajes con expresiones de radianes:
(Escribe ESCpiESC para el carácter π.)

In[500]:=
```
Sin[π / 2]
```
seno

Out[500]=
```
1
```

O escribe ESCdegESC para el símbolo incorporado de grado:

In[501]:=
```
Sin[90 °]
```
seno

Out[501]=
```
1
```

Automáticamente expande (o reduce) expresiones trigonométricas usando identidades:

In[502]:=
```
TrigExpand[Sin[2 x]]
```
expande funci··· seno

Out[502]=
```
2 Cos[x] Sin[x]
```

Factoriza polinomios trigonométricos:

In[503]:=
```
TrigFactor[Cos[x]^2 - Sin[x]^2]
```
factoriza funci··· coseno seno

Out[503]=
$$2 \text{Sin}\left[\frac{\pi}{4} - x\right] \text{Sin}\left[\frac{\pi}{4} + x\right]$$

Resuelve ecuaciones trigonométricas:

In[504]:=
```
Solve[Cos[x]^2 + Sin[x]^2 == x]
```
resuelve coseno seno

Out[504]=
```
{{x → 1}}
```

Especifica un dominio para soluciones:

In[505]:=
```
Solve[{Tan[x] == 1, 0 < x < 2 Pi}]
```
resuelve tangente núme

Out[505]=
$$\left\{\left\{x \to \frac{\pi}{4}\right\}, \left\{x \to \frac{5\pi}{4}\right\}\right\}$$

COORDENADAS POLARES

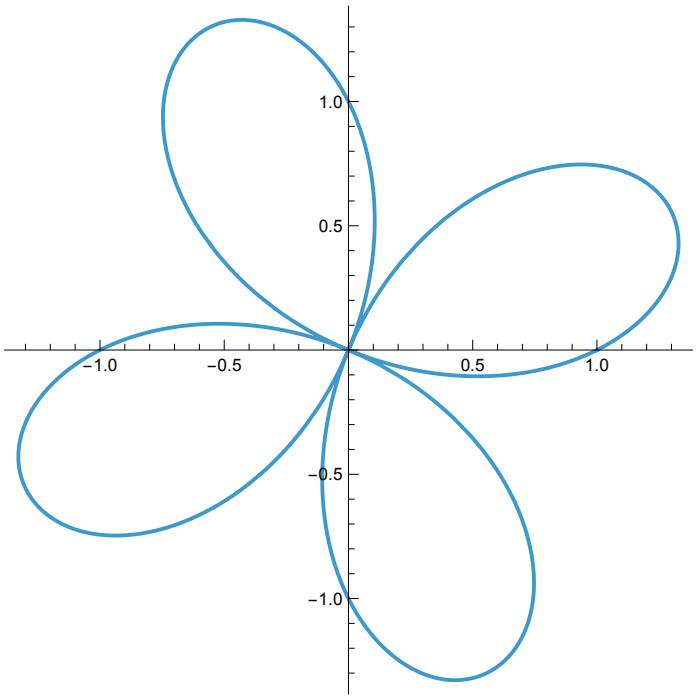
Crea un gráfico polar en 2D:
(Escribe ESCthESC para el símbolo θ.)

In[506]:=

```
PolarPlot[Sin[2 θ] + Cos[2 θ], {θ, 0, 2 Pi}]
```

representaci··· seno coseno núme

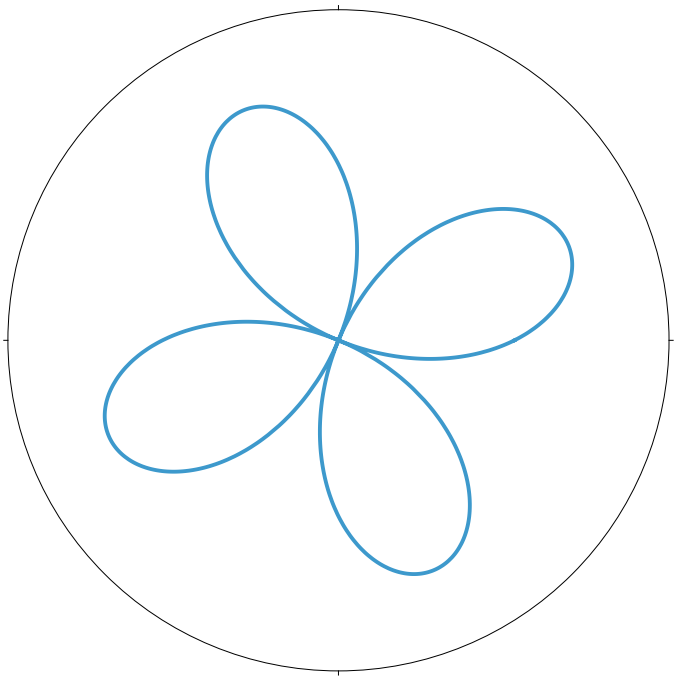
Out[506]=



Muestra los ejes polares:

In[507]:=
```
PolarPlot[Sin[2 θ] + Cos[2 θ], {θ, 0, 2 Pi},
 PolarAxes → Automatic, PolarTicks → {0 °, 90 °, 180 °, 270 °}]
```
representaci··· seno coseno número pi
ejes polares automático marcas polares

Out[507]=



Convierte coordenadas cartesianas a polares:

In[508]:=
```
ToPolarCoordinates[{1, 1}]
```
convierte a coordenadas polares

Out[508]=

$$\left\{\sqrt{2}\,,\ \frac{\pi}{4}\right\}$$

EXPONENCIALES Y LOGARITMOS

El logaritmo natural (neperiano) de una expresión:

In[509]:=
```
N[Log[2]]
```
··· logaritmo

Out[509]=
```
0.693147
```

Sería lo mismo si lo escribimos:

In[510]:=
```
N[Log[E, 2]]
```
··· log··· número e

Out[510]=
```
0.693147
```

In[511]:=
```
Log[E^2]
```
logaritmo

Out[511]=
```
2
```

El logaritmo decimal de un número:

In[512]:=
```
N[Log[10, 2]]
```
··· logaritmo

Out[512]=
```
0.30103
```

El logaritmo en base 2 de un número:

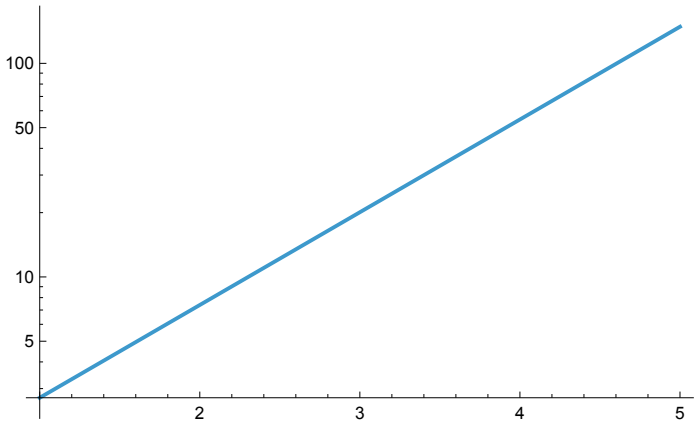
In[513]:=
```
N[Log[2, 64]]
```
··· logaritmo

Out[513]=
```
6.
```

Crea representaciones gráficas en una escala logarítmica:

In[514]:=
```
LogPlot[E^x, {x, 1, 5}]
```
represen··· número e

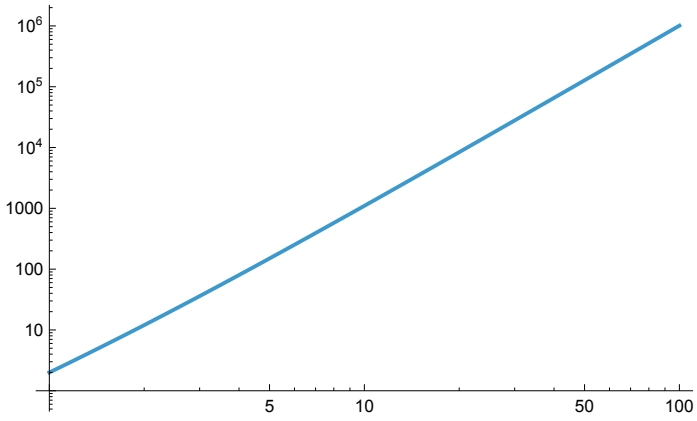
Out[514]=



En una escala doblemente logarítmica:

In[515]:=
```
LogLogPlot[x^2 + x^3, {x, 1, 100}]
```
representación log log

Out[515]=



LÍMITES

Calculamos el límite de una expresión, cuando la variable tiende a un número:

In[516]:=
```
Limit[(x^3 - 1) / (x - 1), x → 1]
```
límite

Out[516]=

3

Calculamos el límite de una expresión, cuando la variable tiende a infinito:

In[517]:=
```
Limit[(2 x^3 - 1) / (5 x^3 + x - 1), x → ∞]
```
límite

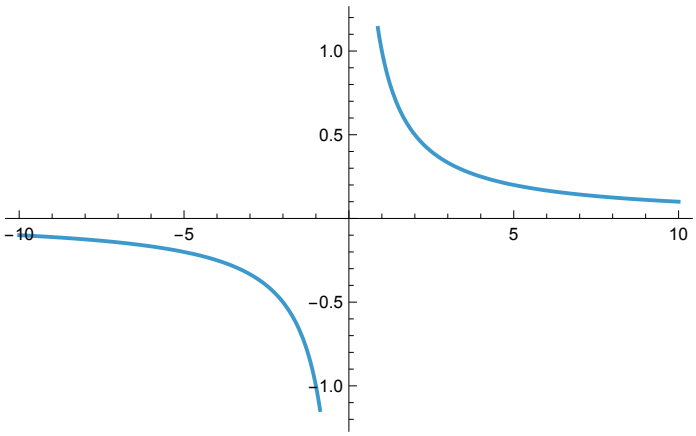
Out[517]=

$$\frac{2}{5}$$

Podemos calcular límites laterales de la función 1/x. En primer lugar la representamos:

In[518]:=
```
Plot[1 / x, {x, -10, 10}]
```
representación gráfica

Out[518]=



Los límites laterales son:

In[519]:=
```
Limit[1 / x, x → 0, Direction → 1]
```
límite dirección

Out[519]=
$$-\infty$$

In[520]:=
```
Limit[1 / x, x → 0, Direction → -1]
```
límite dirección

Out[520]=
$$\infty$$

Usamos la expresión TraditionalForm para escribir una expresión matemática sin calcular:

In[521]:=
```
TraditionalForm[HoldForm[Limit[1 / x, x → Infinity]]]
```
forma tradicional forma sin ⋯ límite infinito

Out[521]//TraditionalForm=
$$\lim_{x\to\infty} \frac{1}{x}$$

DERIVADAS

Calculamos las derivadas de la siguiente forma:

In[522]:=
```
D[x^6, x]
```
deriva

Out[522]=
$$6\,x^5$$

Podemos utilizar la notación de primas:

In[523]:=
```
Sin'[x]
```
seno

Out[523]=
```
Cos[x]
```

Podemos derivar una función creada por el usuario:

In[524]:=
```
f[x_] := x^2 + 2 x + 1;
```

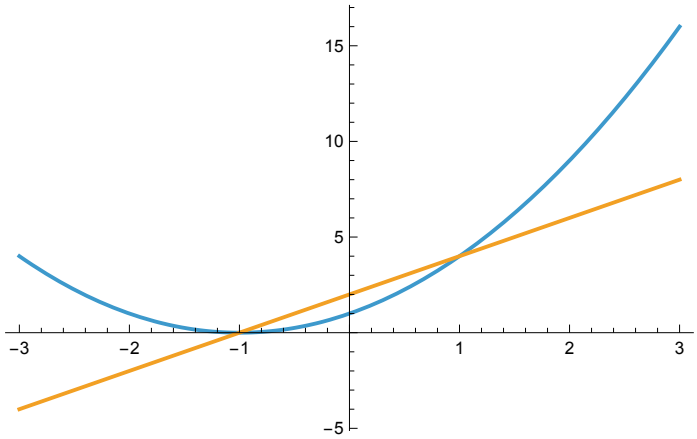
In[525]:=
```
f'[x]
```

Out[525]=
$$2 + 2\,x$$

Podemos representar gráficamente una función y su derivada:

In[526]:=
```
Plot[{f[x], f'[x]}, {x, -3, 3}]
```
representación gráfica

Out[526]=



Podemos calcular múltiples derivadas:

In[527]:=
```
D[x^6, {x, 3}]
```
deriva

Out[527]=
$$120\, x^3$$

Y también lo podemos hacer:

In[528]:=
```
Sin''[x]
```
seno

Out[528]=
```
-Sin[x]
```

INTEGRALES

Podemos calcular la integral de una función como:

In[529]:=
```
Integrate[8 x^4, x]
```
integra

Out[529]=
$$\frac{8\, x^5}{5}$$

O también, escriba **ESC intt ESC**:

In[530]:=
$$\int 8\, x^4 \, dx$$

Out[530]=
$$\frac{8\, x^5}{5}$$

Si queremos una aproximación numérica, debemos utilizar **NIntegrate**:

In[531]:=
```
NIntegrate[x^3 Sin[x] + 2 Log[3 x]^2, {x, 0, Pi}]
```
integra numéricame··· seno logaritmo núme

Out[531]=
```
28.1531
```

SECUENCIAS, SUMAS Y SERIES

Las secuencias de números se representan mediante listas:

In[532]:=
```
Table[x^2, {x, 1, 10}]
```
tabla

Out[532]=
```
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

Podemos utilizar secuencias ya incorporadas:

In[533]:=
```
Table[Fibonacci[x], {x, 1, 9}]
```
tabla sucesión de Fibonacci

Out[533]=
```
{1, 1, 2, 3, 5, 8, 13, 21, 34}
```

Podemos definir secuencias recurrentes usando **RecurrenceTable**. Por ejemplo, calcula los 8 primeros términos de la sucesión $a_n = 2\, a_{n-1}$, $a_1 = 1$

In[534]:=
```
RecurrenceTable[{a[n] == 2 a[n - 1], a[1] == 1}, a, {n, 1, 8}]
```
tabla de recurrencia

Out[534]=
```
{1, 2, 4, 8, 16, 32, 64, 128}
```

Si queremos calcular la suma total de los términos:

In[535]:=
```
Total[%]
```
total

Out[535]=
```
255
```

Si queremos calcular una secuencia a partir de una fórmula generadora:

In[536]:=
```
Sum[i (i + 1), {i, 1, 10}]
```
suma

Out[536]=
```
440
```

Esto mismo, lo podemos escribir, ESC sumt ESC:

In[537]:=
$$\sum_{i=1}^{10} i\,(i+1)$$

Out[537]=
```
440
```

Podemos realizar sumas indefinidas y múltiples:

In[538]:=
$$\sum_{i=1}^{n}\sum_{j=1}^{n} ij$$

Out[538]=
$$ij\, n^2$$

Podemos generar aproximaciones por series de potencias:

In[539]:=
```
Series[Exp[x^2], {x, 0, 8}]
```
serie exponencial

Out[539]=
$$1 + x^2 + \frac{x^4}{2} + \frac{x^6}{6} + \frac{x^8}{24} + O[x]^9$$

Donde $O[x]^9$ representa términos de orden superior que han sido omitidos. Podemos eliminar ese término:

In[540]:=
```
Normal[%]
```
normal

Out[540]=
$$1 + x^2 + \frac{x^4}{2} + \frac{x^6}{6} + \frac{x^8}{24}$$

Un ejemplo de una serie convergente:

In[541]:=
$$\sum_{n=0}^{\infty} 0.5^n$$

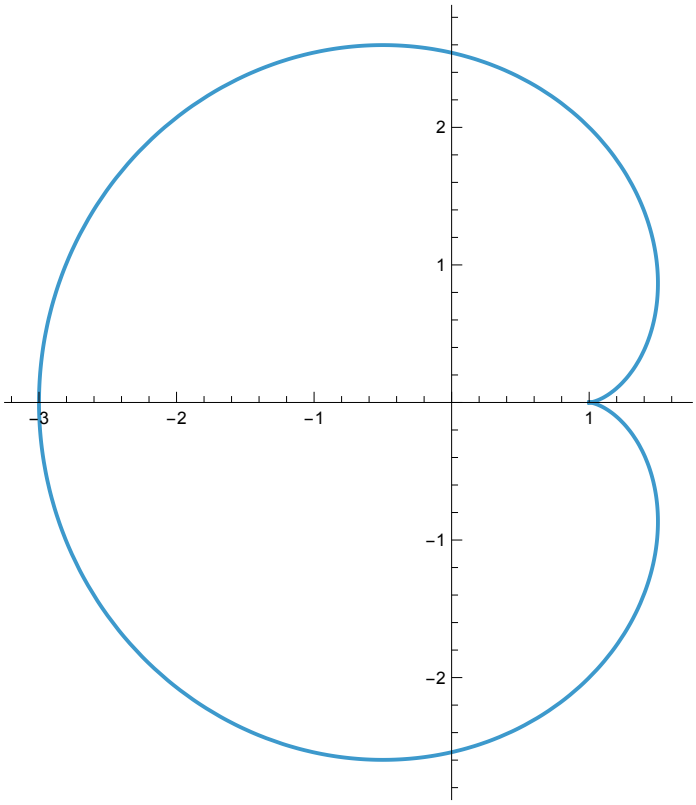
Out[541]=
$$2.$$

GRÁFICOS EN DOS DIMENSIONES

Usa **ParametricPlot** para representar gráficamente un conjunto de ecuaciones paramétricas:

In[542]:=
```
ParametricPlot[{2 Cos[t] - Cos[2 t], 2 Sin[t] - Sin[2 t]}, {t, 0, 2 Pi}]
```
gráfico paramétrico coseno coseno seno seno númer

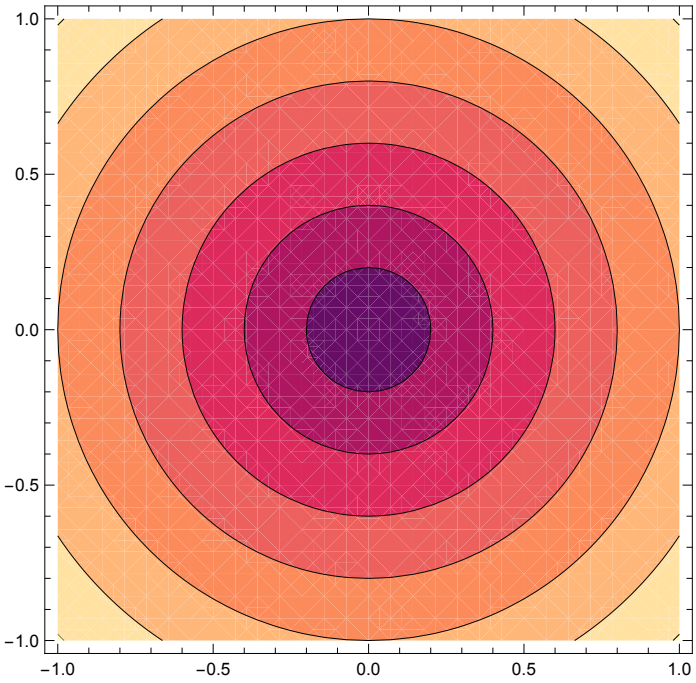
Out[542]=



Creamos un **ContourPlot** de una función en múltiples variables:

In[543]:=
```
ContourPlot[Sqrt[x^2 + y^2], {x, -1, 1}, {y, -1, 1}]
```
representación ⋯ raíz cuadrada

Out[543]=



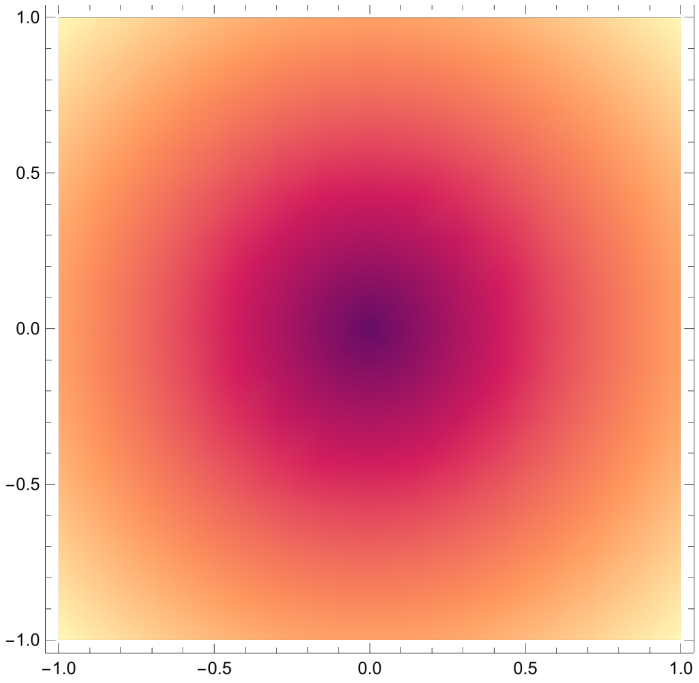
Para la versión contínua usar **DensityPlot**:

In[544]:=

```
DensityPlot[Sqrt[x^2 + y^2], {x, -1, 1}, {y, -1, 1}]
```

representación ⋯ raíz cuadrada

Out[544]=



GRÁFICOS EN TRES DIMENSIONES

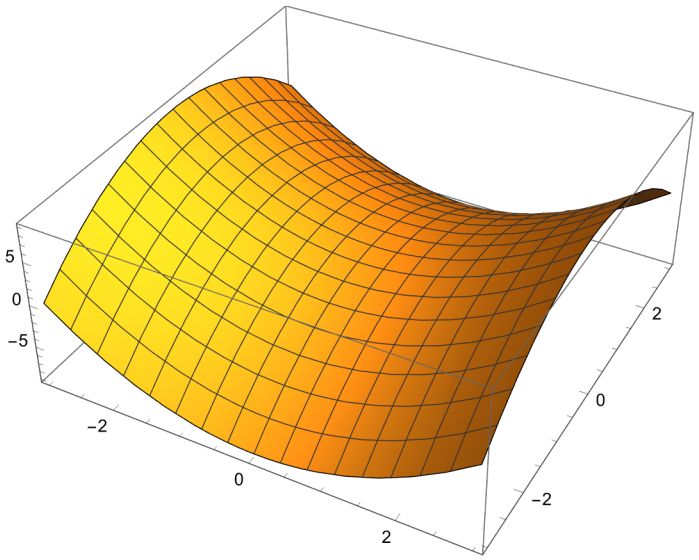
Plot3D representará una curva o superficie cartesiana en 3D:

In[545]:=

```
Plot3D[x^2 - y^2, {x, -3, 3}, {y, -3, 3}]
```

representación gráfica 3D

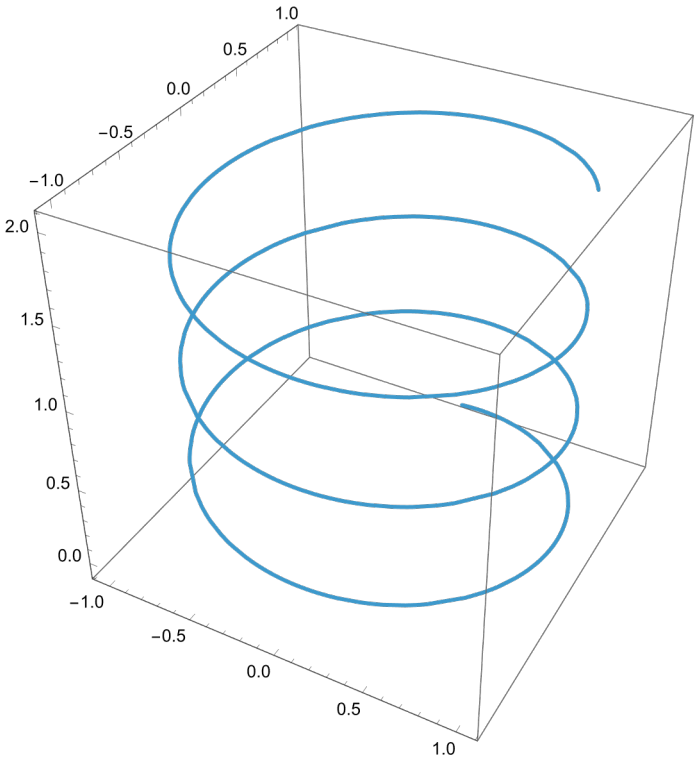
Out[545]=



Usa **ParametricPlot3D** para representar una curva en el espacio 3D:

In[546]:=
```
ParametricPlot3D[{Sin[u], Cos[u], u / 10}, {u, 0, 20}]
```
gráfico paramétrico 3D seno coseno

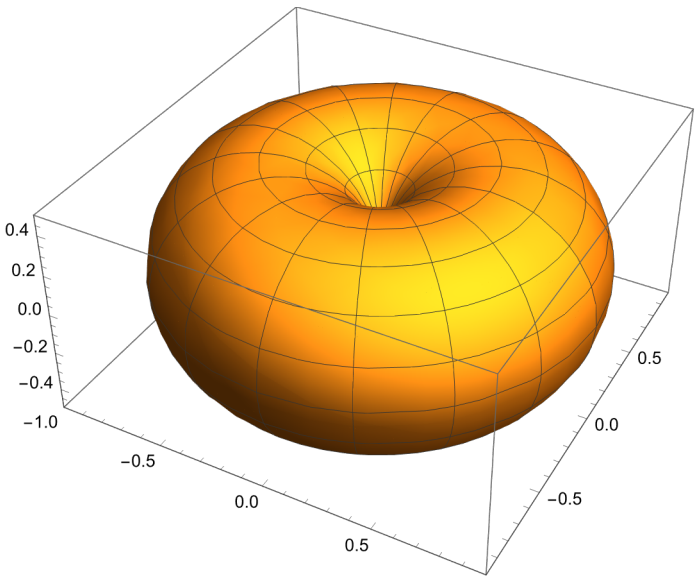
Out[546]=



Si queremos la representación en coordenadas esféricas usa **SphericalPlot3D**:

In[547]:=
```
SphericalPlot3D[Sin[θ], {θ, 0, Pi}, {ϕ, 0, 2 Pi}]
```
representación 3D esf⋯ seno número pi númer

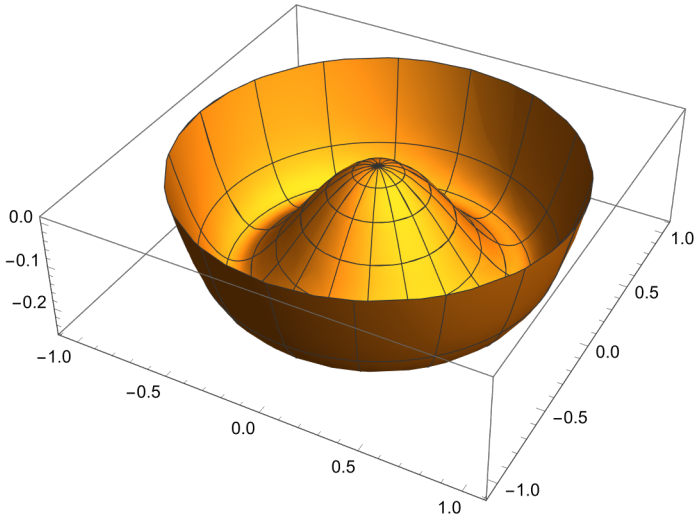
Out[547]=



RevolutionPlot3D construye la superficie formada girando una expresión sobre un eje:

In[548]:=
```
RevolutionPlot3D[x^4 - x^2, {x, -1, 1}]
```
gráfico de revolución 3D

Out[548]=



CÁLCULO MULTIVARIABLE

D funciona para derivadas parciales, simplemente especifica las variables a diferenciar:

In[549]:=
```
D[x^3 z + 2 y^2 x + y z^3, y, z]
```
deriva

Out[549]=
$$3\,z^2$$

Podemos usar el símbolo $\partial_{\square}\square$. (Escribe ESC pd ESC para ∂ y CTRL+- para el subíndice.):

In[550]:=
```
∂x,y (x^2 - 2 x y + x y z)
```

Out[550]=
$$-2 + z$$

Las integrales múltiples usan la misma notación que las integrales simples.
(Escribe ESC int ESC para $\int$ y ESC dd ESC para d.)

In[551]:=
```
∫∫∫ (x^2 + y^2 + z^2) dx dy dz
```

Out[551]=
$$\frac{1}{3}\,x^3\,y\,z + \frac{1}{3}\,x\,y^3\,z + \frac{1}{3}\,x\,y\,z^3$$

Los resultados simbólicos, a menudo, suelen ser complicados:

In[552]:=
$$\int_{-1}^{1}\int_{-2}^{x}\left(x\,\text{Sin}\left[y^2\right]+y\,\text{Cos}\left[x^2\right]\right)\,dy\,dx$$
seno coseno

Out[552]=
$$\frac{1}{4}\left(2\,\text{Cos}[1]+\sqrt{2\pi}\left(-9\,\text{FresnelC}\left[\sqrt{\frac{2}{\pi}}\right]+\text{FresnelS}\left[\sqrt{\frac{2}{\pi}}\right]\right)+2\,\text{Sin}[1]\right)$$

Cuando sucede esto, siempre se puede obtener una salida aproximada, utilizando el comando **N**:

In[553]:=
```
N[%, 5]
```
valor numérico

Out[553]=
```
-3.2243
```

ANÁLISIS Y VISUALIZACIÓN DE VECTORES

En Wolfram los vectores se representan por medio de listas de longitud n.

Por ejemplo, calcula el producto escalar de dos vectores:

In[554]:=
```
{1, 2, 3}.{a, b, c}
```

Out[554]=
```
a + 2 b + 3 c
```

Otra forma de escribir el producto escalar, sería:

In[555]:=
```
Dot[{1, 2, 3}, {a, b, c}]
```
producto escalar

Out[555]=
```
a + 2 b + 3 c
```

Si queremos calcular el producto vectorial de los vectores:
(Escribe ESC cross ESC para el símbolo del producto vectorial)

In[556]:=
```
{1, 2, c} × {a, b, c}
```

Out[556]=
```
{2 c - b c, -c + a c, -2 a + b}
```

Otra forma de escribir el producto vectorial, sería:

In[557]:=
```
Cross[{1, 2, 3}, {a, b, c}]
```
producto vectorial

Out[557]=
```
{-3 b + 2 c, 3 a - c, -2 a + b}
```

Si queremos calcular el módulo (norma) de un vector:

In[558]:=
```
Norm[{1, 1, 1}]
```
norma

Out[558]=
$$\sqrt{3}$$

Si queremos calcular la proyección de un vector en el eje x:

In[559]:=
```
Projection[{8, 6, 7}, {1, 0, 0}]
```
proyección

Out[559]=
```
{8, 0, 0}
```

Podemos encontrar el ángulo formado por dos vectores:

In[560]:=
```
VectorAngle[{1, 0}, {0, 1}]
```
ángulo de vector

Out[560]=
$$\frac{\pi}{2}$$

Si queremos calcular el gradiente de un vector:
(Para el símbolo ∇, usa ESC grad ESC.)

In[561]:=
```
∇{x,y} {x^2 + y, x + y^2}
```

Out[561]=
```
{{2 x, 1}, {1, 2 y}}
```

Si lo que queremos es computar la divergencia o rotacional de un campo vectorial:

In[562]:=
```
Div[{f[x, y, z], g[x, y, z], h[x, y, z]}, {x, y, z}]
```
divergencia

Out[562]=
```
h^(0,0,1)[x, y, z] + g^(0,1,0)[x, y, z] + f^(1,0,0)[x, y, z]
```

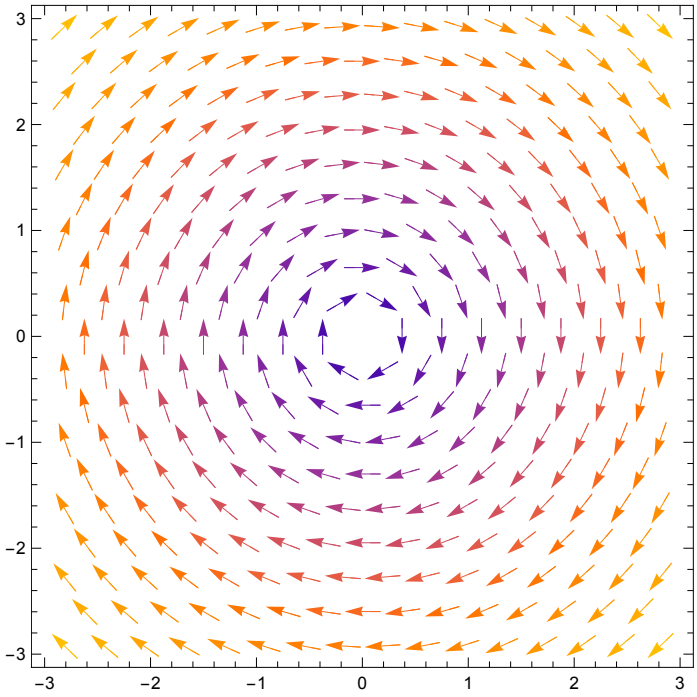
Wolfram Language posee funciones para 2D y 3D adecuadas para la visualización de campos vectoriales:

In[563]:=

```
VectorPlot[{y, -x}, {x, -3, 3}, {y, -3, 3}]
```

representación vectorial

Out[563]=

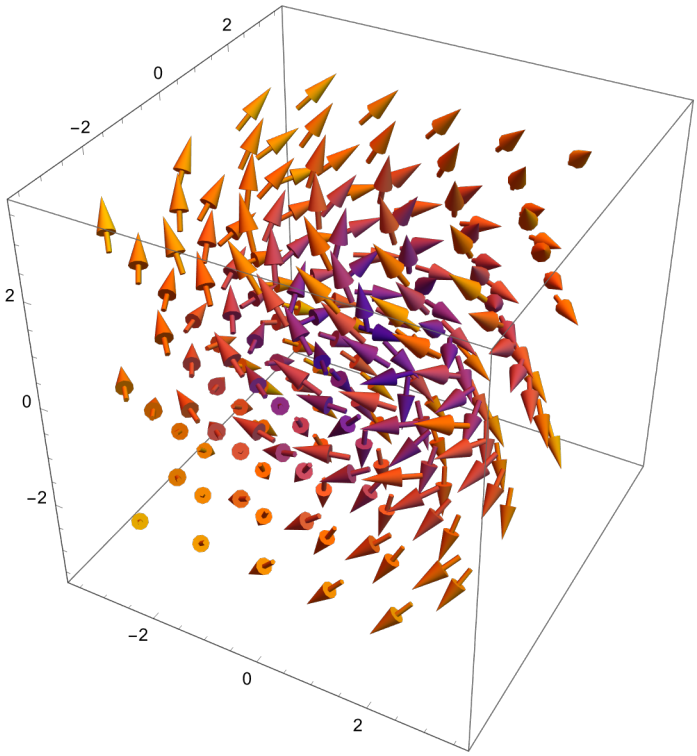


In[564]:=

```
VectorPlot3D[{y, -x, z}, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}]
```

representación 3D vectorial

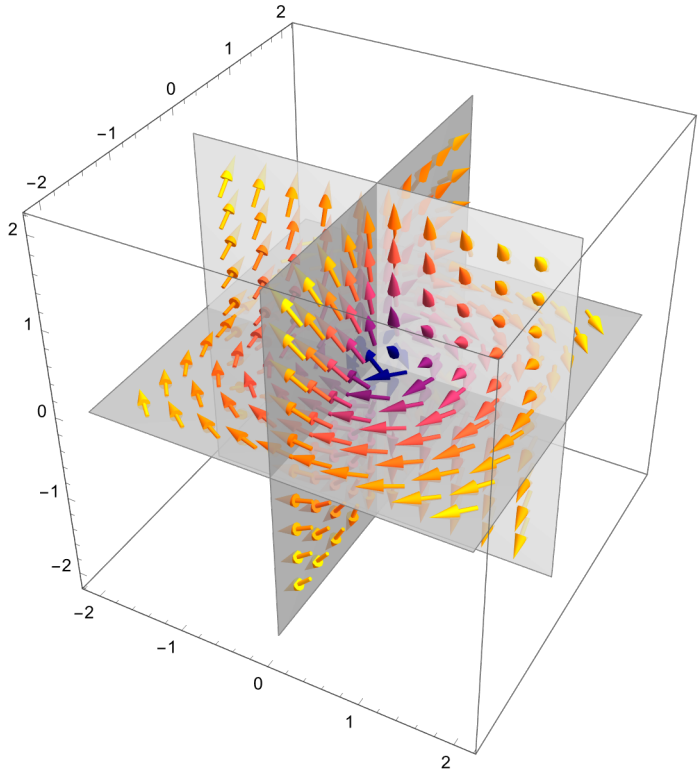
Out[564]=



Representa gráficamente un campo vectorial sobre una superficie de corte:

In[565]:=
```
SliceVectorPlot3D[{y, -x, z}, "CenterPlanes", {x, -2, 2}, {y, -2, 2}, {z, -2, 2}]
```
sección de representación 3D vectorial

Out[565]=



ECUACIONES DIFERENCIALES

DSolveValue toma una ecuación diferenciaL y regresa una solución general:
(C[1] representa una constante de integración.)

In[566]:=
```
sol = DSolveValue[y'[x] + y[x] == x, y[x], x]
```
valor de resolución diferencial

Out[566]=
$$-1 + x + e^{-x} C_1$$

Usa /. para remplazar la constante:

In[567]:=
```
sol /. C[1] → 1
```
constante

Out[567]=
$$-1 + e^{-x} + x$$

O agrega condiciones para una solución específica:

In[568]:=
```
DSolveValue[{y'[x] + y[x] == x, y[0] == -1}, y[x], x]
```
valor de resolución diferencial

Out[568]=
$$-1 + x$$

NDSolveValue encuentra soluciones numéricas:

In[569]:=
```
NDSolveValue[{y'[x] == Cos[x^2], y[0] == 0}, y[x], {x, -5, 5}]
```
valor de resolución diferencial ⋯ coseno

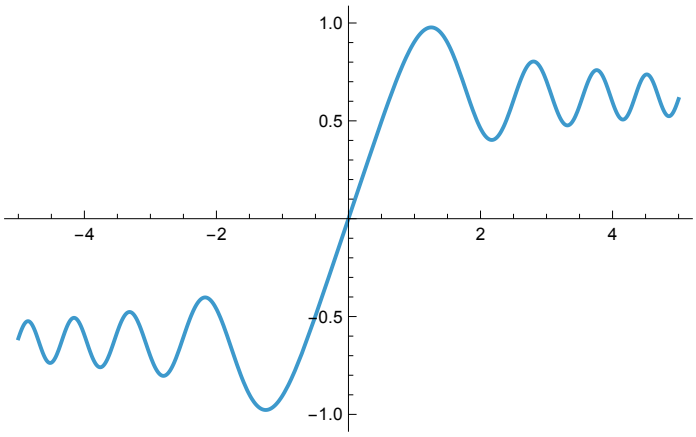
Out[569]=

InterpolatingFunction [Domain: {{−5., 5. }} Output: scalar] [x]

Puedes representar gráficamente esta función **InterpolatingFunction** directamente:

In[570]:=
```
Plot[%, {x, -5, 5}]
```
representación gráfica

Out[570]=



Para resolver sistemas de ecuaciones diferenciales, incluye todas las ecuaciones y sus condiciones en una lista:
(Nota que los saltos de línea no tienen efecto.)

In[571]:=
```
{xsol, ysol} = NDSolveValue[{x'[t] == -y[t] - x[t]^2,
                  valor de resolución diferencial numérica
                                y'[t] == 2 x[t] - y[t]^3,
                                x[0] == y[0] == 1},
                                {x, y}, {t, 20}]
```

Out[571]=

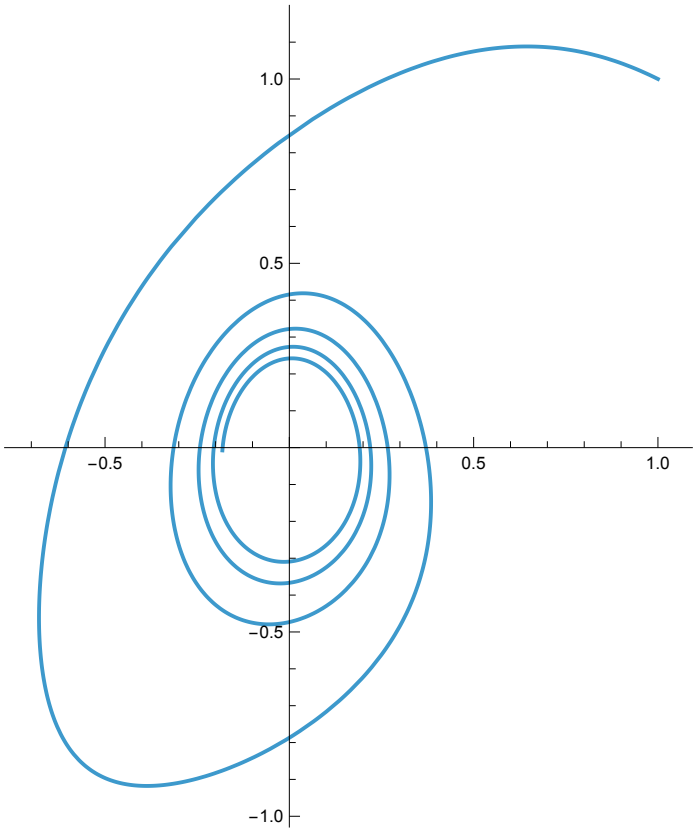
{InterpolatingFunction [Domain: {{0., 20. }} Output: scalar],

InterpolatingFunction [Domain: {{0., 20. }} Output: scalar]}

Visualiza la solución como una representación gráfica paramétrica:

In[572]:=
```
ParametricPlot[{xsol[t], ysol[t]}, {t, 0, 20}]
```
gráfico paramétrico

Out[572]=



ANÁLISIS COMPLEJO

La unidad imaginaria $\sqrt{-1}$ se representa como I:

In[573]:=
```
I^2
```

Out[573]=
```
-1
```

La mayoría de las operaciones manejan números complejos de forma automática:

In[574]:=
```
(1 + I) (2 - 3 I)
```
número i núm

Out[574]=
```
5 - i
```

Expande expresiones complejas:

In[575]:=
```
ComplexExpand[Sin[x + I y]]
```
expande funciones⋯ seno número

Out[575]=
```
Cosh[y] Sin[x] + i Cos[x] Sinh[y]
```

Convierte las expresiones entre formas exponenciales y trigonométricas:

In[576]:=
```
ExpToTrig[E^(I x)]
```
exponencial a trigonométrico

Out[576]=
```
Cos[x] + i Sin[x]
```

In[577]:=
```
TrigToExp[%]
```
trigonométrico a exponencial

Out[577]=
$$e^{i x}$$

Escribe *ESCcoESC* para el símbolo del conjugado:

In[578]:=
```
(3 + 2 I)*
```
núm

Out[578]=
```
3 - 2 i
```

Extrae las partes reales e imaginarias de una expresión:

In[579]:=
```
ReIm[3 + 2 I]
```
partes real e· núm

Out[579]=
```
{3, 2}
```

O encuentra el valor absoluto y el argumento:

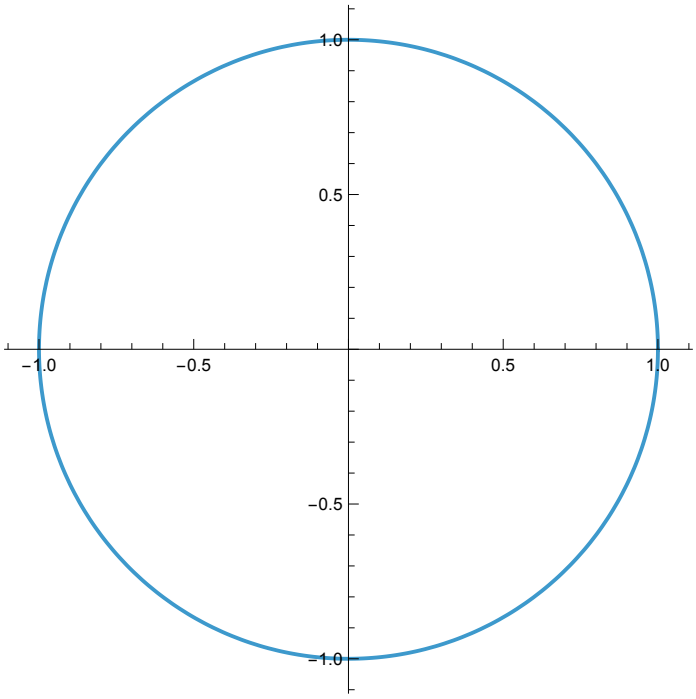
In[580]:=
```
AbsArg[(1 + I)]
```
valor absoluto··· núme

Out[580]=
$$\left\{\sqrt{2}\,,\ \frac{\pi}{4}\right\}$$

Representamos una función compleja de forma paramétrica:

In[581]:=
```
ParametricPlot[ReIm[E^(I ω)], {ω, 0, 2 π}]
```
gráfico paramétrico parte·· n··· número i

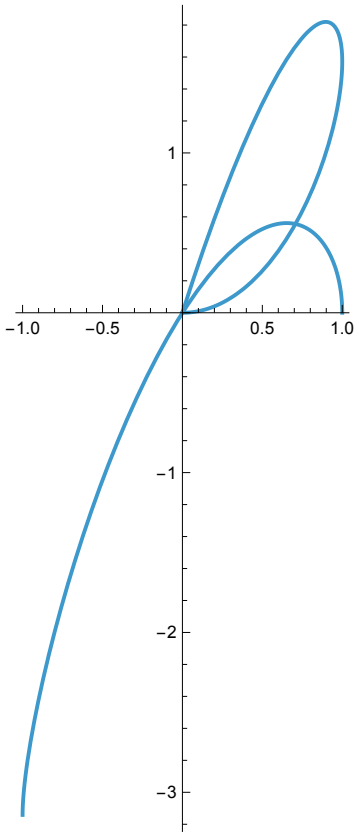
Out[581]=



Usamos **AbsArg** en un **PolarPlot**:

In[582]:=
```
PolarPlot[AbsArg[E^(I ω)], {ω, 0, π}]
```
representaci··· valor a··· n··· número i

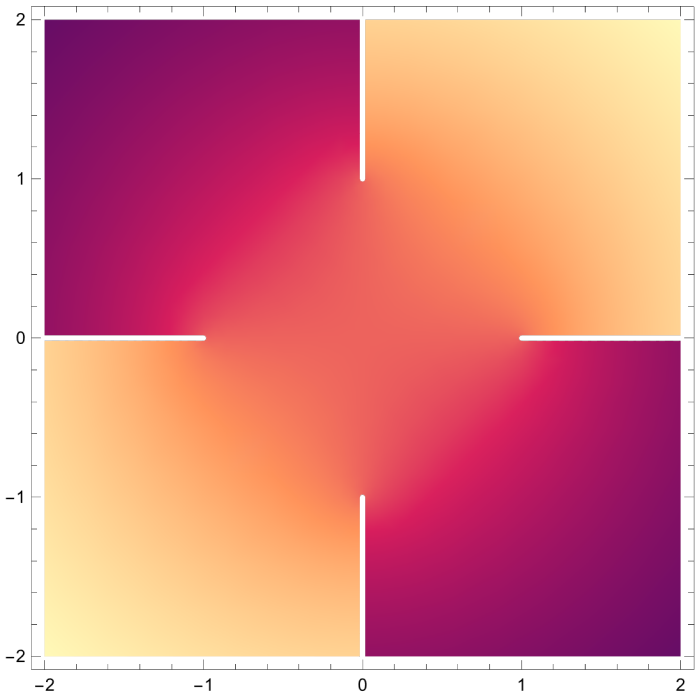
Out[582]=



Visualizamos componentes complejos con un **DensityPlot**:

In[583]:=
```
DensityPlot[Im[ArcSin[(x + I y)^2]], {x, -2, 2}, {y, -2, 2}]
```
representación ··· p··· arco seno número i

Out[583]=



MATRICES Y ALGEBRA LINEAL

Wolfram Language representa matrices como listas de listas:

In[584]:=
```
MatrixForm[{{1, 2}, {3, 4}}]
```
forma de matriz

Out[584]//MatrixForm=

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Puedes construir una matriz con funciones iterativas:

In[585]:=
```
MatrixForm[Table[x + y, {x, 1, 3}, {y, 0, 2}]]
```
forma de matriz tabla

Out[585]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{pmatrix}$$

Puedes importar datos que representan una matriz:

In[586]:=
```
MatrixForm[Import[
forma de matriz  importa
  "/Users/msm/Documents/Ciencia/Mathematica/Aprendizaje/Matemáticas/Aprendizaje
    /Recordatorios/data.csv"]]
```
Out[586]//MatrixForm=
$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Las operaciones de matriz estándar funcionan con base en elementos:

In[587]:=
```
{1, 2, 3} {a, b, c}
```
Out[587]=
```
{a, 2 b, 3 c}
```

Computa el producto escalar de dos matrices:

In[588]:=
```
{{1, 2}, {3, 4}}.{{a, b}, {c, d}}
```
Out[588]=
```
{{a + 2 c, b + 2 d}, {3 a + 4 c, 3 b + 4 d}}
```

Otra forma de escribirlo:

In[589]:=
```
Dot[{{1, 2}, {3, 4}}.{{a, b}, {c, d}}]
producto escalar
```
Out[589]=
```
{{a + 2 c, b + 2 d}, {3 a + 4 c, 3 b + 4 d}}
```

Si queremos calcular el determinante:

In[590]:=
```
Det[{{a, b}, {c, d}}]
determinante
```
Out[590]=
```
-b c + a d
```

Si queremos calcular la matriz inversa:

In[591]:=
```
Inverse[{{1, 1}, {0, 1}}]
matriz inversa
```
Out[591]=
```
{{1, -1}, {0, 1}}
```

Use **LinearSolve** para resolver un sistema lineal:

In[592]:=
```
LinearSolve[{{1, 1}, {0, 1}}, {x, y}]
resuelve ecuación lineal
```
Out[592]=
```
{x - y, y}
```

MATEMÁTICAS DISCRETAS

Realiza operaciones básicas de teoría de números tales como la factorización:

In[593]:=
```
FactorInteger[30]
```
factoriza entero

Out[593]=
```
{{2, 1}, {3, 1}, {5, 1}}
```

Encuentra el GCD (máximo común divisor) y el LCM (mínimo común múltiplo) de dos números enteros cualquiera:

In[594]:=
```
GCD[24, 60]
```
máximo común divisor

Out[594]=
```
12
```

In[595]:=
```
LCM[24, 60]
```
mímino común múltiplo

Out[595]=
```
120
```

Muestra el cuarto número primo:

In[596]:=
```
Prime[4]
```
número primo

Out[596]=
```
7
```

Prueba si el número es primo:

In[597]:=
```
PrimeQ[%]
```
¿primo?

Out[597]=
```
True
```

In[598]:=
```
Table[Prime[n], {n, 1, 100}]
```
tabla número primo

Out[598]=
```
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79,
 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163,
 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251,
 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349,
 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439,
 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541}
```

Si queremos calcular el resto de una división:

In[599]:=
```
Mod[17, 5]
```
operación módulo

Out[599]=
```
2
```

Obtener las permutaciones posibles de una lista:

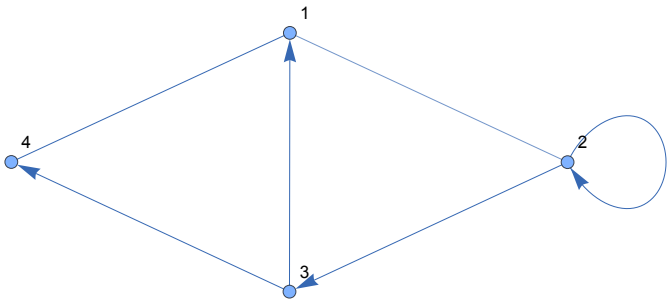
In[600]:=
```
Permutations[{a, b, c}]
```
permutaciones

Out[600]=
```
{{a, b, c}, {a, c, b}, {b, a, c}, {b, c, a}, {c, a, b}, {c, b, a}}
```

Genera un **Graph** a partir de una lista de bordes:
(Usa ESCueESC para un UndirectedEdge o ESCdeESC para un DirectedEdge.)

In[601]:=
```
Graph[{1 ↔ 2, 2 ↦ 3, 3 ↦ 4, 4 ↔ 1, 3 ↦ 1, 2 ↦ 2}, VertexLabels → All]
```
grafo etiquetas de vértices todo

Out[601]=



Encuentra la ruta más corta entre dos vértices:

In[602]:=
```
FindShortestPath[%, 3, 2]
```
encuentra camino más corto

Out[602]=
```
{3, 1, 2}
```

PROBABILIDAD

Calcula factoriales usando la notación:

In[603]:=
```
5!
```

Out[603]=
```
120
```

Para combinaciones, usa **Binomial**:

In[604]:=
```
Binomial[4, 3]
```
número binomial

Out[604]=
```
4
```

Computa probabilidades para una distribución binomial:

(Escribe ESCdistESC para el símbolo ≈.)

In[605]:=
```
Probability[x == 1, x ≈ BinomialDistribution[1, 1 / 2]]
```
probabilidad distribución binomial

Out[605]=
$$\frac{1}{2}$$

Calcula la expectación de una expresión polinómica:

In[606]:=
```
Expectation[2 x + 3, x ≈ NormalDistribution[]]
```
expectación distribución normal

Out[606]=
3

Obtén el PDF de una distribución normal en forma simbólica:

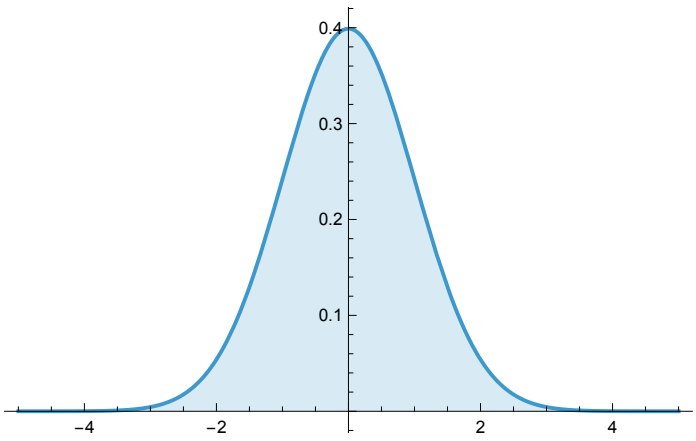
In[607]:=
```
PDF[NormalDistribution[0, 1], x]
```
fun··· distribución normal

Out[607]=
$$\frac{e^{-\frac{x^2}{2}}}{\sqrt{2\,\pi}}$$

Crea una representación del resultado:

In[608]:=
```
Plot[%, {x, -5, 5}, Filling → Axis]
```
representación gráfica relleno eje

Out[608]=



ESTADÍSTICA

Computa la media de una lista de números:

In[609]:=
```
Mean[{1, 2, 4, 5}]
```
media

Out[609]=
3

Encuentra la correlación de múltiples listas:

In[610]:=
```
Correlation[{1, 3, 5}, {6, 4, 2}]
```
correlación

Out[610]=
$$-1$$

Encuentra la desviación estándar de una distribución de Poisson:

In[611]:=
```
StandardDeviation[PoissonDistribution[2]]
```
desviación estándar distribución Poisson

Out[611]=
$$\sqrt{2}$$

Calcula el momento de una lista de símbolos:

In[612]:=
```
Moment[{x, y, z}, 2]
```
momento

Out[612]=
$$\frac{1}{3}\left(x^2+y^2+z^2\right)$$

Obtén la función generadora del momento de una distribución:
(Escribe ESCmESC para μ y ESCsESC para σ.)

In[613]:=
```
MomentGeneratingFunction[NormalDistribution[μ, σ], t]
```
función generatriz de momentos distribución normal

Out[613]=
$$e^{t\,\mu+\frac{t^2\,\sigma^2}{2}}$$

Genera datos estadísticos con RandomVariate:
(Usa //Short para obtener un resumen breve de la salida.)

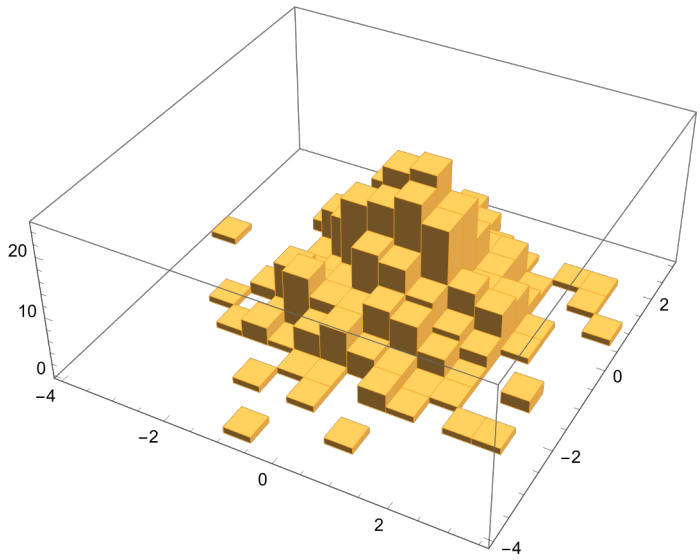
In[614]:=
```
RandomVariate[NormalDistribution[0, 1], {500, 2}] // Short
```
variable aleatoria distribución normal corto

Out[614]//Short=
```
{{0.0962335, 0.785926}, <<498>>, {1.34816, 1.38803}}
```

Visualiza los datos resultantes:

In[615]:=
```
Histogram3D[%]
```
histograma 3D

Out[615]=

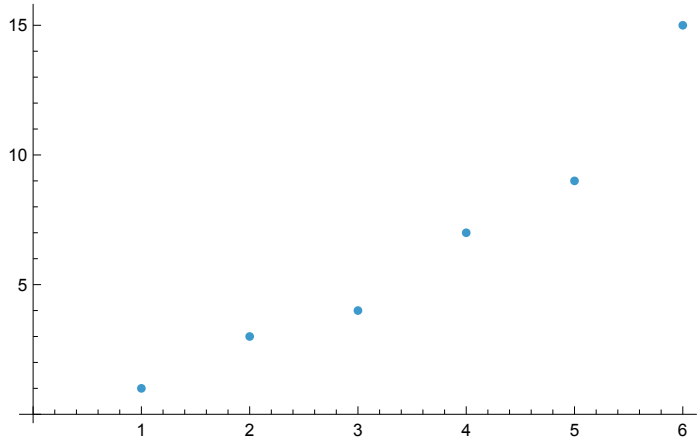


GRÁFICOS DE DATOS Y CURVAS DE MEJOR AJUSTE

Usa **ListPlot** para visualizar los datos en un gráfico de dispersión:

In[616]:=
```
ListPlot[{1, 3, 4, 7, 9, 15}]
```
representación de lista

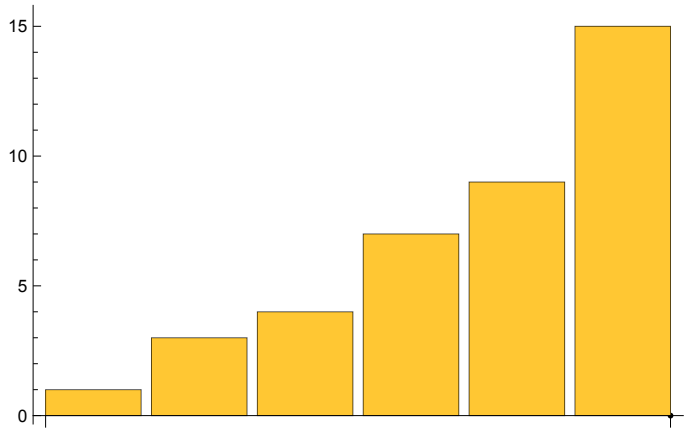
Out[616]=



O también podemos representarlos como un diagrama de barras:

In[617]:=
```
BarChart[{1, 3, 4, 7, 9, 15}]
```
diagrama de barras

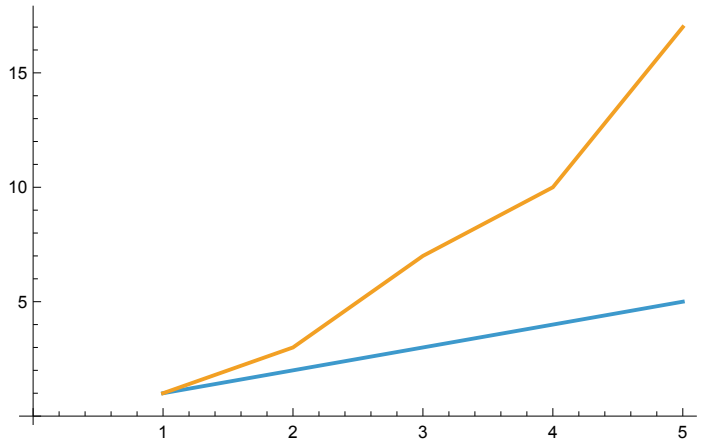
Out[617]=



Múltiples conjuntos de datos son coloreados de forma distinta automáticamente:

In[618]:=
```
ListLinePlot[{{1, 2, 3, 4, 5}, {1, 3, 7, 10, 17}}]
```
gráfico de línea de una lista

Out[618]=



Encuentra una curva de mejor ajuste con el comando **Fit**:
({1,x,x2} significa ajuste cuadrático sobre x.)

In[619]:=
```
Fit[{2, 3, 5, 7, 11, 13}, {1, x, x^2}, x]
```
ajusta

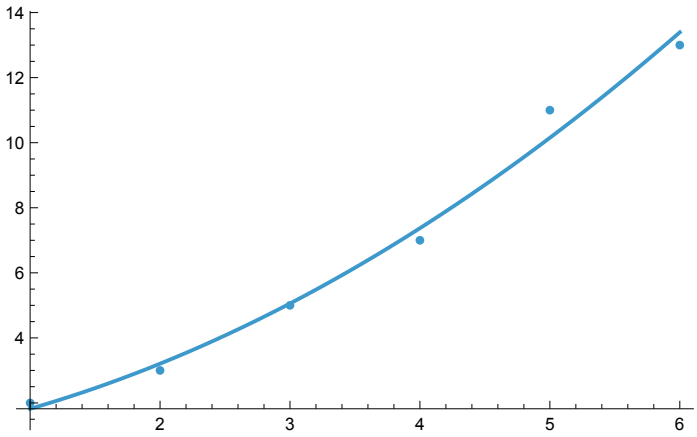
Out[619]=

$0.9 + 0.689286\, x + 0.232143\, x^2$

Usa **Show** para comparar la curva con sus puntos de datos:

In[620]:=
```
Show[{Plot[%, {x, 1, 6}], ListPlot[{2, 3, 5, 7, 11, 13}]}]
```
muestra | representación gráfica | representación de lista

Out[620]=



TEORIA DE GRUPOS

Obtén una lista de elementos de grupo:

In[621]:=
```
GroupElements[SymmetricGroup[2]]
```
elementos de grupo | grupo simétrico

Out[621]=
```
{Cycles[{}], Cycles[{{1, 2}}]}
```

Determina los generadores de un grupo:

In[622]:=
```
GroupGenerators[SymmetricGroup[3]]
```
generadores de grupo | grupo simétrico

Out[622]=
```
{Cycles[{{1, 2}}], Cycles[{{1, 2, 3}}]}
```

Crea un grupo de permutación a partir de los generadores:

In[623]:=
```
PermutationGroup[{Cycles[{{3, 1, 2}}], Cycles[{{2, 5, 6}}]}]
```
grupo de permutaciones | ciclos | ciclos

Out[623]=
```
PermutationGroup[{Cycles[{{1, 2, 3}}], Cycles[{{2, 5, 6}}]}]
```

Computa su orden:

In[624]:=
```
GroupOrder[%]
```
orden de grupo

Out[624]=
```
60
```

Muestra la tabla de multiplicación para un grupo:

In[625]:=
```
TableForm[GroupMultiplicationTable[DihedralGroup[2]], TableHeadings → Automatic]
```
forma de tabla | tabla de multiplicación de grupo | grupo dihedral | cabeceras de tabla | automático

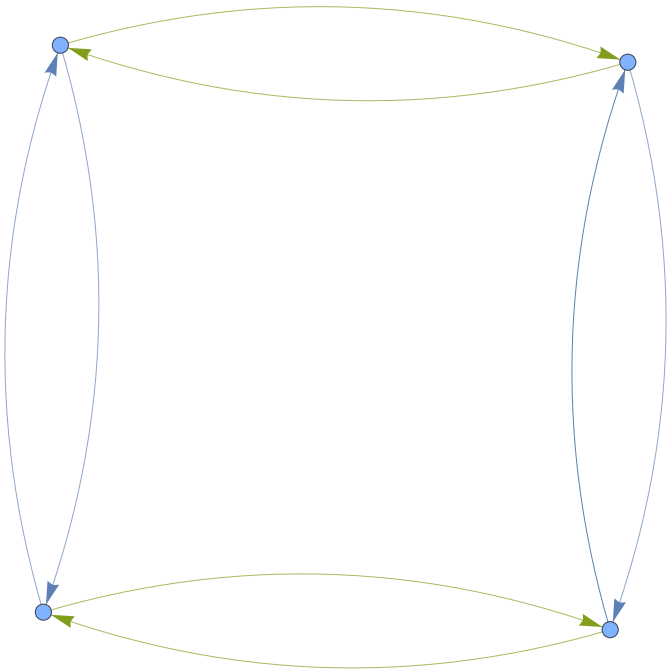
Out[625]//TableForm=

	1	2	3	4
1	1	2	3	4
2	2	1	4	3
3	3	4	1	2
4	4	3	2	1

Visualízalo como un grafo de Cayley:

In[626]:=
```
CayleyGraph[DihedralGroup[2]]
```
grafo Cayley | grupo dihedral

Out[626]=

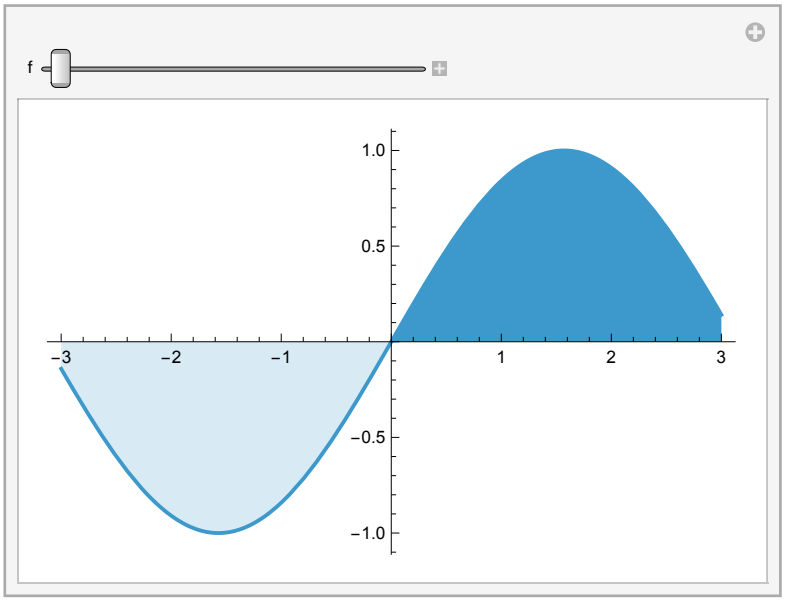


MODELOS INTERACTIVOS

El comando **Manipulate** te permite explorar de forma interactiva lo que sucede cuando varias los parámetros en tiempo real:

In[627]:=
```
Manipulate[Plot[Sin[f x], {x, -3, 3}, Filling → Axis], {f, 1, 5}]
```
manipula repr··· seno relleno eje

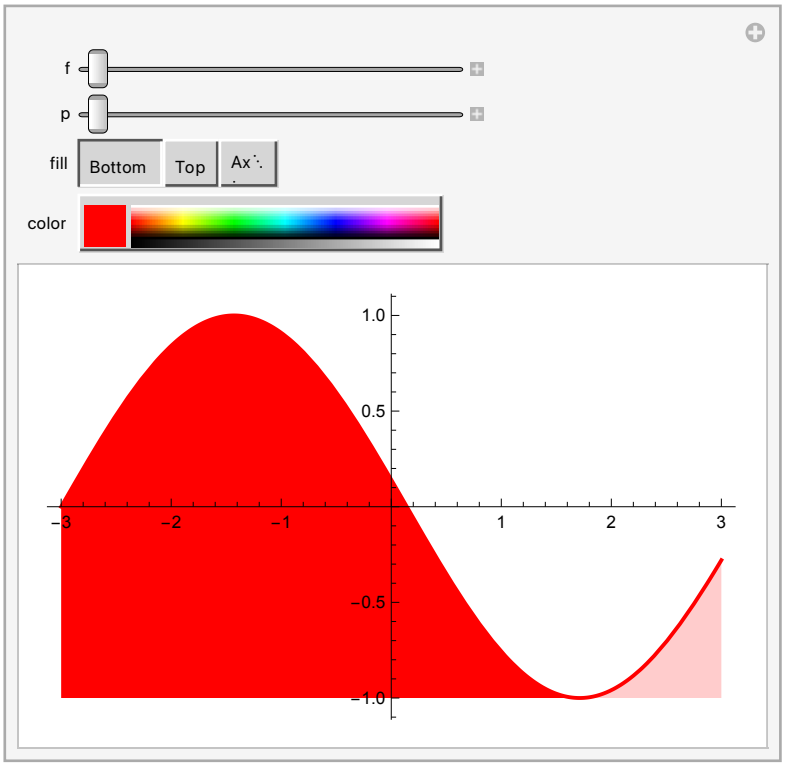
Out[627]=



Un sólo comando Manipulate puede tener múltiples controladores, y Wolfram Language, selecciona automáticamente la mejor presentación de esos controladores:

In[628]:=
```
Manipulate[Plot[Sin[f * x + p], {x, -3, 3}, Filling → fill, PlotStyle → color],
 {f, 1, 5}, {p, 3, 9}, {fill, {Bottom, Top, Axis}}, {color, Red}]
```
manipula repr··· seno relleno estilo de representación
abajo arriba eje rojo

Out[628]=

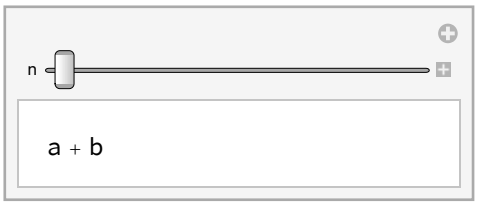


Cualquier expresión en Wolfram Language puede ser manipulada, incluyendo las expresiones no

gráficas:

In[629]:=
```
Manipulate[Expand[(a + b)^n], {n, 1, 20, 1}]
```
manipula expande factores

Out[629]=



IMPLEMENTACIÓN EN LA NUBE

CloudDeploy implementa objetos en Wolfram Cloud.

Crea una página web que tenga una imagen de un gráfico de onda sinusoidal.

In[630]:=
```
CloudDeploy[Plot[Sin[x], {x, 0, 50}]]
```
despliega en la··· repr··· seno

Out[630]=
```
CloudObject[
 https://www.wolframcloud.com/obj/af969100-bb88-4c46-aaa0-3da0f15316d7]
```

Podemos desplegar una interfaz dinámica:

In[631]:=
```
CloudDeploy[Manipulate[Plot[Sin[f x], {x, 0, 2 Pi}], {f, 1, 10}]]
```
despliega en la··· manipula repr··· seno número pi

Out[631]=
```
CloudObject[
 https://www.wolframcloud.com/obj/da4d187e-3df2-4cff-9cfe-e2ef1b80c3bb]
```