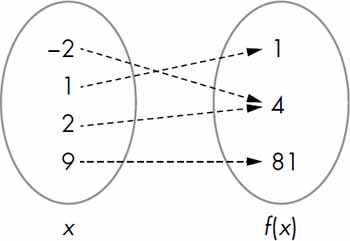
Capítulo 7: Resolución de problemas de cálculo

### **¿Qué es una función?**

Empecemos con algunas definiciones básicas. Una función es una *correspondencia* entre un conjunto de entrada y un conjunto de salida. La condición especial de una función es que un elemento del conjunto de entrada esté relacionado *exactamente* con *un* elemento del conjunto de salida. Por ejemplo, [la Figura 7-1](ch07.html#ch7fig1) muestra dos conjuntos tales que un elemento del conjunto de salida es el cuadrado de un elemento que pertenece al conjunto de entrada.



*Figura 7-1: Una función describe una correspondencia entre un conjunto de entrada y un conjunto de salida. Aquí, un elemento del conjunto de salida es el cuadrado de un elemento del conjunto de entrada.*

Utilizando la conocida notación de funciones, escribiríamos esta función como *f*(*x*) = x2, donde *x* es la cantidad variable independiente. Así, *f*(2) = 4, *f*(100) = 10000, etc. Nos referimos a *x* como la cantidad variable independiente porque somos libres de asumir un valor para ella siempre que ese valor esté dentro de su dominio (véase el siguiente apartado).

Las funciones también pueden definirse en términos de múltiples variables. Por ejemplo, *f(x*, *y*) = x2 + y2 define una función de dos variables, *x* e *y*.

#### ***Dominio y rango de una función***

El *dominio* de una función es el conjunto de valores de entrada que la variable independiente puede asumir válidamente. El conjunto de valores de salida de una función se llama *rango*.

Por ejemplo, el dominio de la función *f(x*) = *1/x* son todos los números reales y complejos distintos de cero, porque 1/0 no está definido. El rango está formado por el conjunto de valores obtenidos al sustituir cada número del dominio por *1/x*, por lo que en este caso también son todos los números reales y complejos distintos de cero.

**NOTA**

*El dominio y el rango de una función pueden ser ciertamente diferentes. Por ejemplo, para la funciónx2, el dominio son todos los números positivos y negativos, pero el rango son sólo los números positivos.*

#### ***Visión general de las funciones matemáticas más comunes***

Ya hemos utilizado varias funciones matemáticas comunes del módulo math de la biblioteca estándar de Python. Un par de ejemplos familiares son las funciones sin() y cos(), que corresponden a las funciones trigonométricas seno y coseno. También se definen otras funciones trigonométricas:tan() y los equivalentes inversos de estas funciones, asin(), acos() y atan().

El módulo math también incluye funciones que hallan el logaritmo de un número -la función logaritmo natural log(), el logaritmo de base-2 log2(), y el logaritmo de base-10 log10()-, así como la función exp(), que halla el valor de *ex*, donde *e* es el número de Euler (aproximadamente 2,71828).

Un inconveniente de todas estas funciones es que no son adecuadas para trabajar con expresiones simbólicas. Si queremos manipular una expresión matemática que incluya símbolos, tenemos que empezar a utilizar las funciones equivalentes definidas por SymPy.

Veamos un ejemplo rápido:

>>> import math  
>>> math.sin(math.pi/2)  
1.0

Aquí, encontramos el seno del ángulo *π/2* utilizando la función sin() definida por el módulo math de la biblioteca estándar. A continuación, podemos hacer lo mismo utilizando SymPy.

>>> import sympy  
>>> sympy.sin(math.pi/2)  
1.00000000000000

Al igual que la función sin() de la biblioteca estándar, la función sin() de SymPy espera que el ángulo se exprese en radianes. Ambas funciones devuelven 1.

Ahora, intentemos llamar a cada función con un símbolo en su lugar y veamos qué ocurre:

>>> from sympy import Symbol  
>>> theta = Symbol('theta')  
➊ >>> math.sin(theta) + math.sin(theta)  
Traceback (most recent call last):  
File "<pyshell#53>", line 1, in  
math.sin(theta) + math.sin(theta)  
File "/usr/lib/python3.4/site-packages/sympy/core/expr.py", line 225, in  
\_\_float\_\_  
raise TypeError("can't convert expression to float")  
TypeError: can't convert expression to float  
  
➋ >>> sympy.sin(theta) + sympy.sin(theta)  
2\*sin(theta)

La función sin() de la biblioteca estándar no sabe qué hacer cuando la llamamos con theta en ➊, así que lanza una excepción para indicar que espera un valor numérico como argumento de la función sin(). En cambio, SymPy es capaz de realizar la misma operación en ➋, y devuelve como resultado la expresión  2\*sin(theta) . Esto apenas nos sorprende ahora, pero ilustra el tipo de tareas en las que las funciones matemáticas de la biblioteca estándar pueden quedarse cortas.

Consideremos otro ejemplo. Supongamos que queremos deducir la expresión del tiempo que tarda un cuerpo en movimiento de proyectil en alcanzar el punto más alto si se lanza con velocidad inicial u con un ángulo theta (ver "[Movimiento de proyectil](ch02.html#ch02lev2sec08)" en [la página 48](ch02.html#page_48)).

En el punto más alto, u\*sin(theta)-g\*t = 0, así que para hallar t, utilizaremos la función solve() que aprendimos en el [Capítulo 4](ch04.html#ch04):

>>> from sympy import sin, solve, Symbol  
>>> u = Symbol('u')  
>>> t = Symbol('t')  
>>> g = Symbol('g')  
>>> theta = Symbol('theta')  
>>> solve(u\*sin(theta)-g\*t, t)  
[u\*sin(theta)/g]

La expresión para t, como aprendimos antes, resulta ser u\*sin(theta)/g, e ilustra cómo la función solve() puede utilizarse también para encontrar soluciones a ecuaciones que contienen funciones matemáticas.

[anterior](ch07_1.html)[Subtema 2 de 11: (Ver todo)](ch07.html)[siguiente](ch07_3.html)