Capítulo 4: Álgebra y matemáticas simbólicas con SymPy

### **Definición de símbolos y operaciones simbólicas**

*Los símbolos* forman los bloques de construcción de la matemática simbólica. El término *símbolo* no es más que un nombre general para las *x*, *y*, *as*y *b*que utilizas en ecuaciones y expresiones algebraicas. Crear y utilizar símbolos nos permitirá hacer las cosas de forma diferente a como las hacíamos antes. Considera las siguientes afirmaciones:

>>> x = 1  
>>> x + x + 1  
3

Aquí creamos una etiqueta, x, para referirnos al número 1. Luego, cuando escribimos la expresión x + x + 1, se evalúa por nosotros, y el resultado es 3. ¿Y si quisieras el resultado en términos del símbolo *x*? Es decir, ¿si en lugar de 3, quisieras que Python te dijera que el resultado es*2x* + 1? No podrías escribir x + x + 1 *sin* la sentencia x = 1, porque Python no sabría a qué se refiere x.

SymPy nos permite escribir programas en los que podemos expresar y evaluar expresiones matemáticas en términos de dichos símbolos. Para utilizar un símbolo en tu programa, tienes que crear un objeto de la clase Symbol, de la siguiente manera:

>>> from sympy import Symbol  
>>> x = Symbol('x')

Primero, importamos la clase Symbol de la biblioteca sympy. A continuación, creamos un objeto de esta clase pasando 'x' como parámetro. Observa que este 'x' se escribe como una cadena entre comillas. Ahora podemos definir expresiones y ecuaciones en términos de este símbolo. Por ejemplo, aquí tienes la expresión anterior:

>>> from sympy import Symbol  
>>> x = Symbol('x')  
>>> x + x + 1  
2\*x + 1

Ahora el resultado se da en términos del símbolo *x*. En la expresión x = Symbol('x'), el x de la izquierda es la etiqueta Python. Es el mismo tipo de etiqueta que hemos utilizado antes, sólo que esta vez se refiere al símbolo *x* en lugar de a un número -más concretamente, a un objeto Symbol que representa el símbolo 'x'. Esta etiqueta tampoco tiene que coincidir necesariamente con el símbolo; podríamos haber utilizado una etiqueta como a o var1. Por lo tanto, no hay ningún problema en escribir las declaraciones anteriores de la siguiente manera:

>>> a = Symbol('x')  
>>> a + a + 1  
2\*x + 1

Sin embargo, utilizar una etiqueta que no coincida puede resultar confuso, por lo que te recomiendo que elijas una etiqueta que tenga la misma letra que el símbolo al que se refiere.

**ENCONTRAR EL SÍMBOLO REPRESENTADO POR UN OBJETO SÍMBOLO**

Para cualquier objeto Symbol, su atributo name es una cadena que es el símbolo real que representa:

>>> x = Symbol('x')  
>>> x.name  
'x'  
>>> a = Symbol('x')  
>>> a.name  
'x'

Puedes utilizar .name en una etiqueta para recuperar el símbolo que almacena.

Para que quede claro, el símbolo que crees debe especificarse como una cadena. Por ejemplo, no puedes crear el símbolo *x* utilizando x = Symbol(x)-debes definirlo como x = Symbol('x').

Para definir varios símbolos, puedes crear objetos Symbol separados o utilizar la función symbols() para definirlos de forma más concisa. Supongamos que quieres utilizar tres símbolos *-x*, *y* y *z- en*tu programa. Podrías definirlos individualmente, como hemos hecho antes:

>>> x = Symbol('x')  
>>> y = Symbol('y')  
>>> z = Symbol('z')

Pero un método más breve sería utilizar la función symbols() para definir los tres a la vez:

>>> from sympy import symbols  
>>> x,y,z = symbols('x,y,z')

Primero, importamos la función symbols() de SymPy. Luego, la llamamos con los tres símbolos que queremos crear, escritos como una cadena con comas separándolas. Una vez ejecutada esta sentencia, x, y, y z harán referencia a los tres símbolos 'x', 'y', y 'z'.

Una vez que hayas definido los símbolos, puedes realizar operaciones matemáticas básicas con ellos, utilizando los mismos operadores que aprendiste en [el Capítulo 1](ch01.html#ch01) (+, -, /, \*, y \*\*). Por ejemplo, puedes hacer lo siguiente:

>>> from sympy import Symbol  
>>> x = Symbol('x')  
>>> y = Symbol('y')  
  
>>> s = x\*y + x\*y  
>>> s  
2\*x\*y

Veamos si podemos encontrar el producto de x(x + x):

>>> p = x\*(x + x)  
>>> p  
2\*x\*\*2

SymPy realizará automáticamente estos cálculos sencillos de suma y multiplicación, pero si introducimos una expresión más compleja, no cambiará. Veamos qué ocurre cuando introducimos la expresión (x + 2)\*(x + 3):

>>> p = (x + 2)\*(x + 3)  
>>> p  
(x + 2)\*(x + 3)

Es posible que esperaras que SymPy lo multiplicara todo e imprimiera x\*\*2 + 5\*x + 6. En lugar de eso, la expresión se imprimió exactamente como la introdujimos. SymPy sólo simplifica automáticamente las expresiones más básicas y deja que el programador exija explícitamente la simplificación en casos como el anterior. Si quieres multiplicar la expresión para obtener la versión expandida, tendrás que utilizar la función expand(), que veremos dentro de un momento.

[anterior](ch04_1.html)[Subtema 2 de 7: (Ver todo)](ch04.html)[siguiente](ch04_3.html)