Capítulo 4: Álgebra y matemáticas simbólicas con SymPy

### **Retos de programación**

Aquí tienes algunos retos de programación que te ayudarán a aplicar mejor lo que has aprendido. Puedes encontrar ejemplos de soluciones en [*http://www.nostarch.com/doingmathwithpython/*](http://www.nostarch.com/doingmathwithpython/).

#### ***#nº 1: Buscador de factores***

Has aprendido sobre la función factor(), que imprime los factores de una expresión. Ahora que sabes cómo puede manejar tu programa las expresiones introducidas por un usuario, escribe un programa que pida al usuario que introduzca una expresión, calcule sus factores y los imprima. Tu programa debe ser capaz de manejar entradas no válidas haciendo uso del manejo de excepciones.

#### ***#2: Solucionador gráfico de ecuaciones***

Antes has aprendido a escribir un programa que pida al usuario que introduzca una expresión como *3x* + *2y* - 6 y cree la gráfica correspondiente. Escribe un programa que pida al usuario dos expresiones y luego grafique ambas, como se indica a continuación:

>>> expr1 = input('Enter your first expression in terms of x and y: ')  
>>> expr2 = input('Enter your second expression in terms of x and y: ')

Ahora, expr1 y expr2 almacenarán las dos expresiones introducidas por el usuario. Deberás convertir ambas en objetos SymPy utilizando el paso sympify() en un bloque try...except.

Todo lo que tienes que hacer a partir de aquí es trazar estas dos expresiones en lugar de una.

Una vez hecho esto, mejora tu programa para que imprima la solución, es decir, el par de valores *x* e *y* que satisface ambas ecuaciones. Éste será también el punto de intersección de las dos rectas de la gráfica. (Sugerencia: consulta cómo hemos utilizado antes la función solve() para hallar la solución de un sistema de dos ecuaciones lineales).

#### ***#3: Suma de una serie***

Ya vimos cómo hallar la suma de una serie en "[Imprimir una serie](ch04.html#ch04lev3sec01)", en [la página 99](ch04.html#page_99). Allí sumamos manualmente los términos de la serie haciendo un bucle sobre todos los términos. Aquí tienes un fragmento de ese programa:

for i in range(2, n+1):  
series = series + (x\*\*i)/i

La función summation() de SymPy puede utilizarse directamente para hallar dichas sumas. El siguiente ejemplo imprime la suma de los cinco primeros términos de la serie que hemos considerado antes:

>>> from sympy import Symbol, summation, pprint  
>>> x = Symbol('x')  
>>> n = Symbol('n')  
➊ >>> s = summation(x\*\*n/n, (n, 1, 5))  
>>> pprint(s)  
x5 x4 x3 x2  
-- + -- + -- + -- + x  
5 4 3 2

Llamamos a la función summation() en ➊, siendo el primer argumento el término *n*de la serie y el segundo una tupla que indica el rango de *n*. Aquí queremos la suma de los cinco primeros términos, así que el segundo argumento es (n, 1, 5).

Una vez que tengas la suma, puedes utilizar el método subs() para sustituir *x* por un valor y hallar el valor numérico de la suma:

>>> s.subs({x:1.2})  
3.51206400000000

Tu reto es escribir un programa que sea capaz de hallar la suma de una serie arbitraria cuando le proporcionas el término *enésimo*de la serie y el número de términos que la componen. Aquí tienes un ejemplo de cómo funcionaría el programa:

Enter the nth term: a+(n-1)\*d  
Enter the number of terms: 3  
3·a + 3·d

En este ejemplo, el *enésimo*término suministrado es el de una *progresión aritmética*. Partiendo de a y d como *diferencia común*, el número de términos hasta el que se debe calcular la suma es 3. La suma resulta ser 3a + 3d, lo que concuerda con la fórmula conocida para la misma.

#### ***#nº 4: Resolución de inecuaciones de una sola variable***

Ya has visto cómo resolver una ecuación utilizando la función solve() de SymPy. Pero SymPy también es capaz de resolver desigualdades de una sola variable, como *x* + 5 > 3 y *sinx* - 0,6 > 0. Es decir, SymPy puede resolver relaciones además de la igualdad, como >, <, etc. Para este reto, crea una función, isolve(), que tome cualquier desigualdad, la resuelva y devuelva la solución.

En primer lugar, vamos a conocer las funciones SymPy que te ayudarán a ponerlo en práctica. Las funciones de resolución de desigualdades están disponibles como tres funciones distintas para desigualdades polinómicas, racionales y todas las demás. Tendremos que elegir la función adecuada para resolver las distintas desigualdades, o obtendremos un error.

Un *polinomio* es una expresión algebraica formada por una variable y coeficientes y en la que sólo intervienen las operaciones de suma, resta y multiplicación y sólo potencias positivas de la variable. Un ejemplo de desigualdad polinómica es x2 + 4 < 0.

Para resolver una desigualdad polinómica, utiliza la función solve\_poly\_inequality():

>>> from sympy import Poly, Symbol, solve\_poly\_inequality  
>>> x = Symbol('x')  
➊ >>> ineq\_obj = -x\*\*2 + 4 < 0  
➋ >>> lhs = ineq\_obj.lhs  
➌ >>> p = Poly(lhs, x)  
➍ >>> rel = ineq\_obj.rel\_op  
>>> solve\_poly\_inequality(p, rel)  
[(-oo, -2), (2, oo)]

Primero, crea la expresión que representa una desigualdad, -x2 + 4 < 0, en ➊ y refiérete a esta expresión con la etiqueta ineq\_obj. A continuación, extrae el lado izquierdo de la desigualdad -es decir, la expresión algebraica -x2 + 4- utilizando el atributo lhs en ➋. A continuación, crea un objeto Poly en ➌ para representar el polinomio que hemos extraído en ➋. El segundo argumento que se pasa al crear el objeto es el objeto símbolo que representa la variable, x. En ➍, extrae el operador relacional del objeto desigualdad utilizando el atributo rel. Por último, llama a la función solve\_poly\_inequality() con el objeto polinomio, p, y rel como los dos argumentos. El programa devuelve la solución como una lista de tuplas, en la que cada tupla representa una solución para la desigualdad como el límite inferior y el límite superior del rango de números. Para esta desigualdad, la solución son todos los números menores que -2 y todos los números mayores que 2.

Una *expresión* racional es una expresión algebraica en la que el numerador y el denominador son polinomios. Aquí tienes un ejemplo de desigualdad racional:

image

Para las inecuaciones racionales, utiliza la función solve\_rational\_inequalities():

>>> from sympy import Symbol, Poly, solve\_rational\_inequalities  
>>> x = Symbol('x')  
➊ >>> ineq\_obj = ((x-1)/(x+2)) > 0  
>>> lhs = ineq\_obj.lhs  
➋ >>> numer, denom = lhs.as\_numer\_denom()  
>>> p1 = Poly(numer)  
>>> p2 = Poly(denom)  
>>> rel = ineq\_obj.rel\_op  
➌ >>> solve\_rational\_inequalities([[((p1, p2), rel)]])  
(-oo, -2) U (1, oo)

Crea un objeto desigualdad que represente nuestra desigualdad racional de ejemplo en ➊ y, a continuación, extrae la expresión racional utilizando el atributo lhs. Separa el numerador y el denominador en las etiquetas numer y denom utilizando el método as\_numer\_denom() en ➋, que devuelve una tupla con el numerador y el denominador como los dos miembros. A continuación, crea dos objetos polinomio, p1 y p2, que representen el numerador y el denominador, respectivamente. Recupera el operador relacional y llama a la función solve\_rational\_inequalities(), pasándole los dos objetos polinomio -p1 y p2- y el operador relacional.

El programa devuelve la solución (-oo, -2) U (1, oo), donde U denota que la solución es la *unión* de los dos *conjuntos* de soluciones formados por todos los números menores que -2 y todos los números mayores que 1. (Aprenderemos sobre conjuntos en el [Capítulo 5](ch05.html#ch05).)

Por último, *sinx* - 0,6 > 0 es un ejemplo de desigualdad que no pertenece ni a la categoría de los polinomios ni a la de las expresiones racionales. Si tienes que resolver una desigualdad de este tipo, utiliza la función solve\_univariate\_inequality():

>>> from sympy import Symbol, solve, solve\_univariate\_inequality, sin  
>>> x = Symbol('x')  
>>> ineq\_obj = sin(x) - 0.6 > 0  
>>> solve\_univariate\_inequality(ineq\_obj, x, relational=False)  
(0.643501108793284, 2.49809154479651)

Crea un objeto desigualdad que represente la desigualdad sin(x) – 0.6 > 0 y luego llama a la función solve\_univariate\_inequality() con los dos primeros argumentos como el objeto desigualdad, ineq\_obj, y el objeto símbolo, x. El argumento de palabra clave relational=False especifica a la función que queremos que la solución se devuelva como un *conjunto*. La solución de esta desigualdad son todos los números que se encuentran entre el primer y el segundo miembro de la tupla que devuelve el programa.

##### **Consejos: Funciones útiles**

Ahora recuerda: tu reto es (1) crear una función, isolve(), que tome cualquier desigualdad y (2) elegir una de las funciones adecuadas que se han comentado en esta sección para resolverla y devolver la solución. Las siguientes pistas pueden ser útiles para implementar esta función.

El método is\_polynomial() puede utilizarse para comprobar si una expresión es un polinomio o no:

>>> x = Symbol('x')  
>>> expr = x\*\*2 - 4  
>>> expr.is\_polynomial()  
True  
>>> expr = 2\*sin(x) + 3  
>>> expr.is\_polynomial()  
False

El método is\_rational\_function() puede utilizarse para comprobar si una expresión es una expresión racional:

>>> expr = (2+x)/(3+x)  
>>> expr.is\_rational\_function()  
True  
>>> expr = 2+x  
>>> expr.is\_rational\_function()  
True  
>>> expr = 2+sin(x)  
>>> expr.is\_rational\_function()  
False

La función sympify() puede convertir una desigualdad expresada como cadena en un objeto desigualdad:

>>> from sympy import sympify  
>>> sympify('x+3>0')  
x + 3 > 0

Cuando ejecutes tu programa, debe pedir al usuario que introduzca una expresión de desigualdad e imprimir la solución.

[anterior](ch04_6.html)[Subtema 7 de 7: (Ver todo)](ch04.html)