Variables simbólicas

Las variables `x` o `y` que vamos a definir no son números, ni tampoco pertenecen a los objetos definidos con **NumPy**. Todas las variables simbólicas son objetos de la clase `sp.Symbol` y sus atributos y métodos son completamente diferentes

```
x = sp.Symbol('x')          # define la variable simbólica x
y = sp.Symbol('y')

a, b, c = sp.symbols('a:c') # define como simbólicas las variables a, b, c.


p = sp.Symbol('p', positive = True)  # Natural
q = sp.Symbol('q', real = True)      # Real
```

```
x = sp.Symbol('x', nonnegative = True) # La raíz cuadrada de un número no negativo es real
y = sp.sqrt(x)
print(y.is_real) # La salida de una variable lógica es True o None

y = 4**sp.S(2)
print(y.is_integer)

b = sp.sqrt(2) # √2
print(f"Es b un número racional? {b.is_rational}")


b = 2**sp.S(-2) # 1/2²
print(f"Es b un número entero? {b.is_integer}")
```

Constantes simbólicas y numéricas

Por ejemplo, podemos definir la constante simbólica $1/3$. Si hacemos lo mismo con números representados por defecto en Python, obtendríamos resultados muy diferentes.

```
pi = sp.pi
E = sp.
raiz = sp.sqrt(p)
infinito = sp.oo

frac_simbolico = sp.Rational(1,3) # Número racional de sympy
frac_numerico = 1/3               # Float
```

Manipulación de expresiones

```
expr = (x-3)*(x-3)**2*(y-2)

expr_expandida = sp.expand(expr)    # Expandir

expr_factorizada = sp.factor(expr)  # Factorizar

expr_simplificada = sp.simplify((x**2 - 6*x + 9)/(x-3) - 3) # Simplificar
```

Solución de ecuaciones

El comando `solve` nos permite resolver una ecuación o un sistema de ecuaciones

```
# Ecuación simple / 1º: Expresión | 2º: Valor igualado
sol = sp.solve(sp.Eq(x**2 - 4, 0), x) # x² - 4 = 0

# Sistema de ecuaciones
x, y = sp.symbols('x y')
ec1 = sp.Eq(2*x + y, 1)
ec2 = sp.Eq(x - y, 3)

sol_sistema = sp.solve((ec1, ec2), (x, y))
```

Funciones

El comando `lambda` nos permite el paso de una expresión a una función

```
exprf = x**2+sp.exp(-3*x)+1

f = sp.Lambda((x),exprf)  # se define la función f

exprf.subs({x:3})         # evaluar la expresión
f(3) # evaluar la función
# Definir la función a trozos
```

```
f = sp.Piecewise(
    (x**2, x < -1),
    (x + 1, (x >= -1) & (x <= 2)),
    (sp.sin(x), x > 2)
)

display(f)
```

Dada una expresión en **SymPy** podemos sustituir unas variables simbólicas por otras o reemplazando las variables simbólicas por constantes. Empleamos la función `subs` y los valores a utilizar en la sustitución vienen definidos por un diccionario de Python:

```
expr = x*x + x*y + y*x + y*y

res = expr.subs({x:1, y:2})    # Sustitución de las variables simbólicas por valores
print(f"Valor de la expresión cuando x=1 e y=2: {res}\n")

expr_sub = expr.subs({x:1-y})  # Sustitución de variable simbólica por una expresión
print(f"Sustitución por otra expresión: {expr_sub}")
```

Funciones y lambdify

`lambdify` convierte la función en una función **NumPy** que puede ser evaluada en una matriz de valores x.

```
expr = x**2 + sp.exp(-x)          # Definir función simbólica

f = sp.lambdify(x, expr, 'numpy') # Convertir a función Python/NumPy

# Usar con NumPy
import numpy as np
import matplotlib.pyplot as plt

x_vals = np.linspace(-2, 2, 100)
y_vals = f(x_vals)

plt.plot(x_vals, y_vals)
plt.grid(True)
plt.show()
```

Representación de funciones

```
from sympy.plotting import plot, plot3d

plot(x**2, (x, -3, 3))  # 2D
plot3d(x**2 + y**2, (x, -3, 3), (y, -3, 3))  # 3D
```

Límites

```
x = sp.Symbol('x')          # Variable simbólica x

f = x**2                    # Expresión

a = -3                      # Punto en el que se calcula el límite

lim = sp.limit(f, x, a)     # Límite de la expresión cuando x se aproxima al punto a

display(lim)
```

Límites laterales

```
f1 = 1/x

limi = sp.limit(f1, x, 0,'+')   # Límite de f1 en a=0 por la DERECHA (+)
print('Limite por la derecha: ',limi)

limd = sp.limit(f1, x, 0,'-')   # Límite de f1 en a=0 por la IZQUIERDA (-)
print('Limite por la izquierda: ',limd)


lim= sp.limit(f1, x, 0)         # Si no especificamos el lado, lo calcula por la DERECHA
display(lim)
```

Asíntotas

- La asíntota horizontal indica que la función se acerca a un **valor constante** cuando $x \to \infty$ o $x \to -\infty$.

$y=L$ es asíntota horizontal si $\lim_{x \to \infty} f(x) = L$ o $\lim_{x \to -\infty} f(x) = L$

- Una asíntota vertical ocurre cuando la función se acerca a **infinito** (o menos infinito) al acercarse a cierto valor x=a.

$\lim_{x \to -a} f(x) = \pm \infty$ y $\lim_{x \to a} f(x) = \pm \infty$

- Asíntota oblicua es la recta con una **pendiente diferente de cero** a la que una función se aproxima indefinidamente cuando x tiende a infinito o menos infinito
  - Una función racional f(x) = P(x)/Q(x) tiene una asíntota oblicua si y solo si el grado del **numerador (P)** es **una unidad mayor** que el **denominador (Q)**
  - Si existe una **asíntota horizontal, no habrá asíntota oblicua**

Es de la forma: $y = mx + n$

1. $m = \lim_{x \to \pm\infty} [f(x)/x]$
2. $n = \lim_{x \to \pm\infty} [f(x) - mx]$

```
f2 = x*x/(x + 1)
```

Asíntota horizontal

```
ahd = sp.limit(f2, x, oo)
ahi = sp.limit(f2, x, -oo)
print('Límite en +oo =',ahd)
print('Límite en -oo =',ahi)

# Como los límites no son un valor constante ( 1 , 5 ...), no existe asíntota horizontal
```

Asíntota vertical

Comprobamos asíntotas verticales en los puntos para los que no está definida la función

```
avd = sp.limit(f2, x, -1,'+')
avi = sp.limit(f2, x, -1,'-')
print('Limite en a=-1 por la derecha = ',avd)
print('Limite en a=-1 por la izquierda = ',avi)

# Como ambos límites dan ±∞ , existe asíntota vertical en ese punto ( -1 )
```

Asíntota oblicua

```
aomd = sp.limit(f2/x, x, oo)          # Pendiente de la asíntota oblicua por la derecha
print('aomd = ',aomd)
aond = sp.limit(f2 - aomd*x, x, oo)   # n de la asíntota oblicua por la derecha
print('aond = ',aond)
```

```python
aomi = sp.limit(f2/x, x, -oo)         # Pendiente de la asíntota oblicua por la izquierda
print('aomi = ',aomi)

aoni = sp.limit(f2 - aomi*x, x, -oo)  # n de la asíntota oblicua por la derecha
print('aoni = ',aoni)

# Como ambas m y n son iguales, hay una sola asíntota oblicua ( y = x -1 )
```