Resumen Python 3

Víctor Mardones Bravo

Índice general

Capítulo 1

El módulo math

1.1. El módulo math

El módulo `math` contiene funciones y constantes comúnmente usadas en matemáticas generales.

```
import math
```

En este capítulo también se verán otras funciones matemáticas que están disponibles en Python por sí solo, sin necesidad de importar este módulo.

1.2. Constantes matemáticas

El módulo `math` tiene 5 constantes matemáticas muy importantes. Primero, se verán las 3 constantes numéricas que almacenan los valores de números irracionales importantes.

La constante `pi` representa el número π, el cual es la relación entre el perímetro y diámetro de una circunferencia.

$\pi = 3{,}14159...$

```
import math

print(math.pi)
```

Salida:

```
3.141592653589793
```

La constante `tau`, escrita en matemáticas como τ, es el doble de `pi`, y también se usa bastante.

$\tau = 2\pi$

```
import math

print(math.tau)

print(math.tau == 2 * math.pi)
```

Salida:

```
6.283185307179586
True
```

La constante `e` representa el número de Euler, otro de los números irracionales más importantes. $e = 2{,}71828...$

```
import math

print(math.e)
```

Salida:

```
2.718281828459045
```

1.3. Infinito

Para representar un número infinito, se puede usar el método `float()` y entregarle el string `"inf"` como argumento.

∞

```
print(float("inf"))  # infinito positivo
print(float("-inf"))  # infinito negativo
```

Salida:

```
inf
-inf
```

Una forma alternativa de hacerlo es usando la constante `inf` del módulo `math`.

```
import math

print(math.inf)
print(-math.inf)
```

Salida:

```
inf
-inf
```

Ambas formas de hacerlo entregan exactamente el mismo resultado.

```
import math

print(math.inf == float('inf'))
print(-math.inf == float('-inf'))
```

Salida:

```
True
True
```

1.4. NaN

La constante `nan` o NaN, acrónimo proveniente del inglés Not a Number (no es un número) no proviene directamente de las matemáticas, pero tiene usos bastante útiles en programación. Representa un número "ilegal".

```
import math

print(math.nan)
```

Salida:

```
nan
```

Generalmente se usa para expresar resultados imposibles de calcular, como raíces negativas o indeterminaciones.

Esta constante se almacena como un float.

```
import math

print(type(math.nan))
```

Salida:

```
<class 'float'>
```

1.5. Funciones matemáticas predefinidas

Python viene con algunas funciones matemáticas sencillas, que se pueden usar sin necesidad de importar el módulo `math`. En algunos casos, el módulio `math` provee versiones más completas de estas funciones.

Para obtener la distancia entre un número y el `0` (su valor absulto), puede usarse la función `abs()`.

```
print(abs(42))
print(abs(-42))
```

Salida:

```
42
42
```

1.6. Funciones matemáticas

La función `fabs()` retorna el valor absoluto de un número, al igual que `abs()`, pero siempre retorna un float.

```
import math

print(math.fabs(-5))
print(abs(-5))
```

Salida:

```
5.0
5
```

Como ya se ha visto, la función `sqrt()`, abreviación del inglés square root, retorna la raíz cuadrada de un número positivo o `0`. El resultado obtenido siempre será la raíz positiva.

```
import math

print(math.sqrt(16))
print(math.sqrt(1.5))
print(math.sqrt(0))
print(math.sqrt(math.e))
```

Salida:

```
4.0
1.224744871391589
0.0
1.6487212707001282
```

No se puede usar con números negativos.

```
import math

print(math.sqrt(-1))
```

Salida:

```
Traceback (most recent call last):
  File "<stdin>", line 3, in <module>
    print(math.sqrt(-1))
ValueError: math domain error
```

La función `factorial()` retorna el factorial de un número.

```
import math

print(math.factorial(0))  # 1, por definición
print(math.factorial(1))  # 1
print(math.factorial(2))  # 2*1
print(math.factorial(3))  # 3*2*!
print(math.factorial(4))  # 4*3*2*1
print(math.factorial(5))  # 5*4*3*2*1
```

Salida:

```
1
1
2
6
24
120
```

El factorial no está definido para los números negativos.

```
import math

print(math.factorial(-1))
```

Salida:

```
Traceback (most recent call last):
  File "<stdin>", line 3, in <module>
    print(math.factorial(-1))
ValueError: factorial() not defined for negative values
```

1.7. Funciones de redondeo

Para redondear un número a un determinado número de decimales, puede usarse la función `round()`.

```
from math import pi

print(round(pi, 0))
print(round(pi, 1))
print(round(pi, 2))
print(round(pi, 4))
print(round(pi, 8))
```

Salida:

```
3.0
3.1
3.14
3.1416
3.14159265
```

1.8. Grados y radianes

Al trabajar con ángulos, es común medirlos usando grados o radianes. El módulo `math` tiene funciones para convertir valores entre estas unidades.

El método `radians()` convierte de grados a radianes.

```
import math

print(math.radians(360))  # 2pi
print(math.radians(180))  # pi
print(math.radians(100.05))
print(math.radians(-20))
```

Salida:

```
6.283185307179586
3.141592653589793
1.7462019166203266
-0.3490658503988659
```

El método `degrees()` convierte de radianes a grados.

```
import math

print (math.degrees(math.pi))
print (math.degrees(math.tau))
print (math.degrees(0))
print (math.degrees(3 * math.pi / 4))
print (math.degrees(7 * math.pi / 4))
```

Salida:

```
180.0
360.0
0.0
135.0
315.0
```

1.9. Funciones trigonométricas

El módulo `math` tiene funciones trigonométricas, similares a las de una calculadora. Estas funciones sólo funcionan con radianes. Para usarlas con grados, primero deben convertirse a radianes con el método `radians()`.

Funciones trigonométricas básicas:

```
import math

print(math.sin(math.pi))  # seno
print(math.cos(0))  # coseno
print(math.tan(math.pi / 4))  # tangente

print(1 / math.cos(0))  # secante
print(1 / math.sin(math.pi))  # cosecante
print(1 / math.tan(math.pi / 4))  # cotangente
```

Salida:

```
1.2246467991473532e-16
1.0
0.9999999999999999
1.0
8165619676597685.0
1.0000000000000002
```

Funciones trigonométricas inversas:

```python
import math

print(math.asin(1))  # arcoseno
print(math.acos(0))  # arcocoseno
print(math.atan(1))  # arcotangente

print(1 / math.acos(0))  # arcosecante
print(1 / math.asin(1))  # arcocosecante
print(1 / math.atan(1))  # arcocotangente
```

Salida:

```
1.5707963267948966
1.5707963267948966
0.7853981633974483
0.6366197723675814
0.6366197723675814
1.2732395447351628
```

Funciones trigonométricas hiperbólicas:

```python
import math

print(math.sinh(math.pi))  # seno hiperbólico
print(math.cosh(0))  # coseno hiperbólico
print(math.tanh(math.pi / 4))  # tangente hiperbólica

print(1 / math.cosh(0))  # secante hiperbólica
print(1 / math.sinh(math.pi))  # cosecante hiperbólica
print(1 / math.tanh(math.pi / 4))  # cotangente hiperbólica
```

Salida:

```
11.548739357257746
1.0
0.6557942026326724
1.0
0.08658953753004696
1.5248686188220641
```

Se debe recordar que los resultados de estas funciones se obtienen por aproximaciones y un computador no puede representar todos los números decimales con precisión perfecta, lo que significa que puede haber pequeños errores de aproximación en los resultados.

1.10. Números finitos e infinitos

1.11. Números complejos