Introducción

Las matemáticas serán parte de casi todos los proyectos que crees a lo largo de tu viaje como desarrollador. Si eres un científico de datos, es posible que estés haciendo cálculos bastante complejos. Si estás creando sitios web, es posible que estés determinando los precios o los artículos en un carrito de compras. Python, como cualquier otro lenguaje de programación, proporciona numerosos operadores y bibliotecas para realizar operaciones matemáticas.

Escenario: Crear un programa para calcular la distancia entre planetas

Imagina que estás creando un programa para calcular la distancia entre planetas. El programa permite a un usuario ingresar las distancias de dos planetas desde el sol, y calcula la distancia entre esos dos planetas. Además, deseas proporcionar la distancia tanto en millas como en kilómetros.

¿Qué aprenderás?

En este módulo, explorarás las capacidades matemáticas básicas de Python.

Aprenderás sobre:

- Los operadores matemáticos disponibles en Python.
- El orden de las operaciones.
- Cómo convertir cadenas en números.

¿Cuál es el objetivo principal?

Aprovechar el poder de los operadores matemáticos cuando creamos un programa Python.

¿Qué son los operadores en Python?

Por lo general, las matemáticas implican alrededor de cuatro operaciones principales: suma, resta, multiplicación y división. Python es compatible con estos cuatro operadores y algunos otros. Exploremos los operadores más comunes que usarás en tus programas.

Adición (Suma)

En python usamos `+` para indicar la adición. Usando `+` entre dos números los suma y proporciona el total.

```
answer = 30 + 12
print(answer)
```

Salida: 42

Los operadores se comportan igual cuando usan números literales (como `42`) o variables.

Sustracción (Resta)

Del mismo modo, Python utiliza `-` para la resta. El uso de `-` entre dos números resta los dos números y proporciona la diferencia.

```
difference = 30 - 12
print(difference)
```

Salida: 18

Multiplicación

En Python `*`, es el operador de multiplicación. Proporciona el producto de dos números:

```
product = 30 * 12
print(product)
```

Salida: 360

División

Por último, se utiliza `/` para la división. Proporciona el cociente de dos números:

```
quotient = 30 / 12
print(quotient)
```

Salida: 2.5

Trabajar con la división

Imagina que necesitas convertir un número de segundos en minutos y segundos para la visualización.

```
seconds = 1042
```

El primer paso es determinar el número de minutos que hay en `1042` segundos. Con `60` segundos en un minuto, puedes dividir y obtener una respuesta de `17.3666667`. El número que te interesa es simplemente `17`. Siempre que desees redondear hacia abajo, utilizando lo que se conoce como *división de piso*. Para realizar la división de piso en Python, utilizamos `//`.

```
seconds = 1042
display_minutes = 1042 // 60
print(display_minutes)
```

Salida: 17

El siguiente paso es determinar el número de segundos. Este es el resto de si se divide por `1042`. Puedes encontrar el resto utilizando el operador `módulo`, que es `%` en Python. El resto de `1042 / 60` es `22`, que es lo que proporcionará el operador del módulo.

```
seconds = 1042
display_minutes = 1042 // 60
display_seconds = 1042 % 60

print(display_minutes)
print(display_seconds)
```

Salida: 17, 22

Orden de funcionamiento (Jerarquía de Operaciones)

Python respeta el orden de operación para las matemáticas. El orden de operación dicta que las expresiones deben evaluarse en el siguiente orden:

- 1.- Paréntesis
- 2.- Exponentes
- 3.- Multiplicación y división
- 4.- Suma y resta

Observa cómo se evalúan los paréntesis antes de cualquier otra operación. Esto permite asegurarte de que el código se ejecuta de manera predecible y que su código sea más fácil de leer y mantener.

Como resultado, es una buena práctica usar paréntesis incluso si el orden de operación se evaluaría de la misma manera sin ellos. En las siguientes dos líneas de código, la segunda es más comprensible porque el paréntesis es una indicación clara de qué operación se realizará primero.

```
result_1 = 1032 + 26 * 2
result_2 = 1032 + (26 * 2)
```

El resultado es el mismo en ambos casos - 1084

Trabajar con números en Python

Más allá de la aritmética central, puedes hacer uso de otras operaciones con números. Es posible que debas realizar el redondeo o convertir cadenas en números.

En el escenario de este módulo, deseas aceptar la entrada de un usuario. La entrada será una cadena en lugar de un número, por lo que deberás convertirla en un número. Además, el usuario puede introducir valores que le den una respuesta negativa, que no querrás mostrar. Es posible que debas convertir la respuesta a su valor absoluto (Recordemos que el valor absoluto hace refencia al valor sin signos, es decir sin negativo).

Afortunadamente, Python proporciona utilidades para estas operaciones.

Convertir cadenas en números

Python admite dos tipos principales de números: enteros (o `int`) y coma flotante (o `float`). La diferencia clave entre los dos es la existencia de un punto decimal; los enteros son números enteros, mientras que los floats contienen un valor decimal.

Cuando conviertes cadenas en números, indicamos el tipo de número que deseamos crear. Debemos decidir si necesitamos un punto decimal. Se utiliza `int` para convertir a un número entero y `float` para convertir a un número de coma flotante.

```
demo_int = int('215')
print(demo_int)
```

Salida: 215

```
demo_float = float('215.3')
print(demo_float)
```

Salida: 215.3

Si utilizas un valor no válido para cualquiera de los dos `int` o `float`, obtendrás un error.

Valores absolutos

Un valor absoluto en matemáticas es el número no negativo sin su signo. El uso de un valor absoluto puede ser útil en diferentes situaciones, incluido nuestro ejemplo de buscar determinar la distancia entre dos planetas. Considera las siguientes matemáticas:

```
a = 39 - 16
b = 16 - 39
```

Observa que la diferencia entre las dos ecuaciones es que los números se invierten. Las respuestas son `23` y `-23`, respectivamente. Cuando estás determinando la distancia entre dos planetas, el orden en el que ingresas los números no importa, porque la respuesta absoluta es la misma.

Convertimos el valor negativo en su valor absoluto utilizando `abs`. Si realiza la misma operación utilizando `abs` (e imprimes las respuestas), notarás que se muestra `23` para ambas ecuaciones.

```
print(abs(39 - 16))
print(abs(16 - 39))
```

Salida: 23, 23

Redondeo

La función de python incorporada llamada `round` también es útil. Usada para redondear al entero más cercano si el valor decimal `.5` es mayor o mayor, o hacia abajo si es menor que `.5`.

```
print(round(14.5))
```

Salida: 15

Biblioteca Math

Python tiene bibliotecas para proporcionar operaciones y cálculos más avanzados. Una de las más comunes es la biblioteca `math`. `math` te permite realizar el redondeo con `floor` y `ceil`, proporcionar el valor de pi, y muchas otras operaciones. Veamos cómo usar esta biblioteca para redondear hacia arriba o hacia abajo.

El redondeo de números permite quitar la parte decimal de un flotador. Puedes elegir redondear siempre hacia arriba al número entero más cercano usando `ceil`, o hacia abajo usando `floor`.

```
from math import ceil, floor

round_up = ceil(12.5)
print(round_up)

round_down = floor(12.5)
print(round_down)
```

Salida: 13, 12

Resumen

Se te encomendó la tarea de crear una aplicación que solicita al usuario las distancias del sol para dos planetas, así como calcular la diferencia entre las dos distancias. Utilizaste los operadores aritméticos en Python y funciones como `abs` e `int` para convertir valores.

Aprendiste sobre:

- Los operadores matemáticos disponibles en Python.
- El orden de las operaciones.
- Cómo convertir cadenas en números.

Curso Propedútico de Python para Launch X - Innovacción Virtual.

Material desarrollado con base en los contenidos de MSLearn y la metáfora de LaunchX, traducción e implementación por: Fernanda Ochoa - Learning Producer de LaunchX.

Redes:

- GitHub: FernandaOchoa
- Twitter: @imonsh
- Instagram: fherz8a