Introducción a la Programación con Python

¡Bienvenido al mundo de la programación con Python! Este documento te guiará a través de los conceptos básicos de este lenguaje de programación, comenzando por los fundamentos y avanzando hacia aplicaciones prácticas. Aprenderás sobre variables, operadores, estructuras condicionales, manejo de errores, y cómo convertir tipos de datos. Prepárate para dar tus primeros pasos en el apasionante mundo de la codificación.

 **by Daniel Labrador Benito**

Variables y Operadores Matemáticos

Las variables son como contenedores que almacenan datos en un programa. En Python, puedes crear una variable simplemente asignándole un valor usando el signo igual (=). Por ejemplo, `edad = 25` crea una variable llamada `edad` y le asigna el valor 25. Los operadores matemáticos te permiten realizar cálculos con variables y valores.

Estos son algunos operadores matemáticos comunes en Python:

- Suma (+)
- Resta (-)
- Multiplicación (*)
- División (/)
- Módulo (%)
- Exponenciación (**)

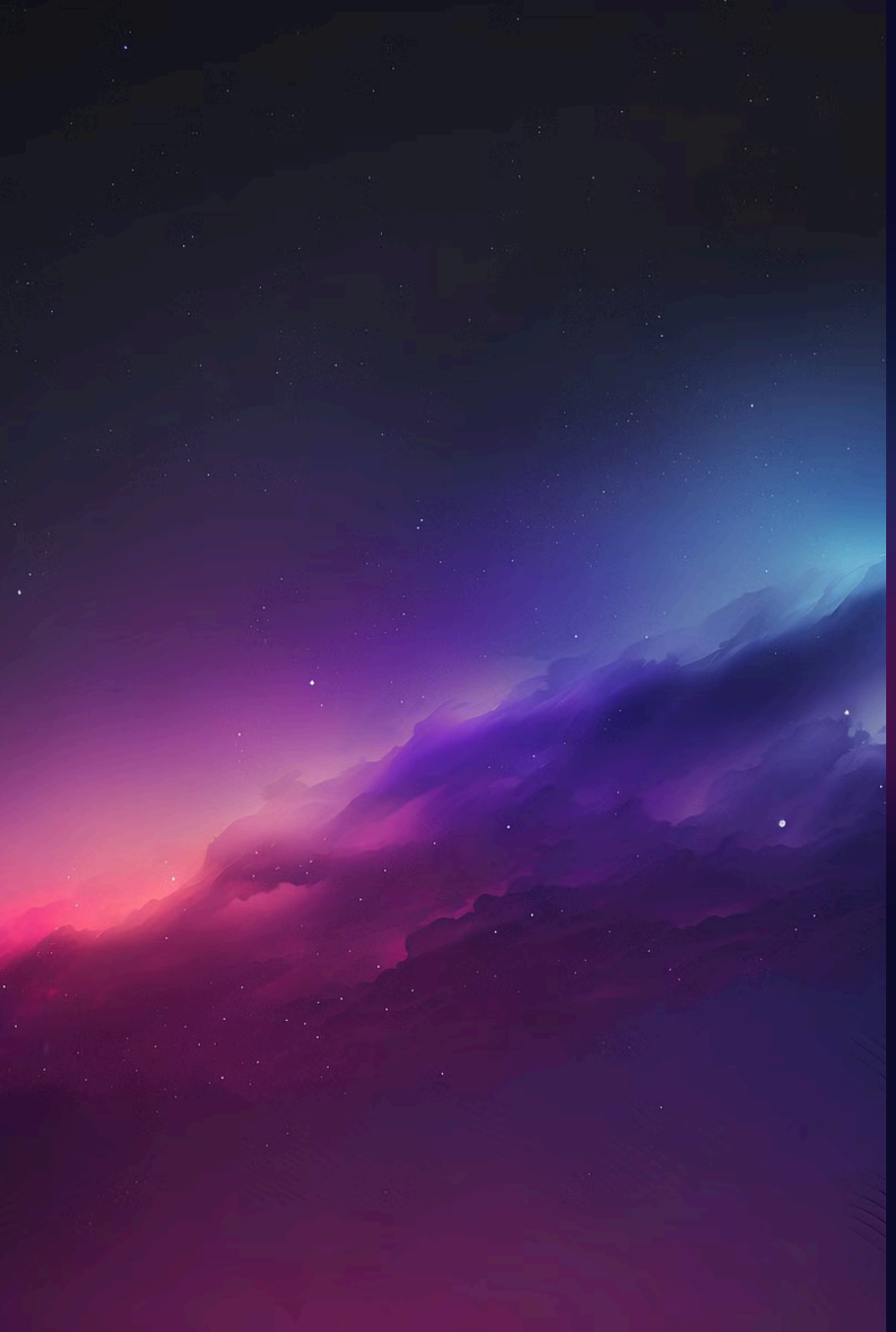
Puedes usar estos operadores para realizar operaciones matemáticas como la suma de dos números: `suma = 10 + 5` o el cálculo del módulo de 10 entre 3: `modulo = 10 % 3`

Operadores Matemáticos

Los operadores matemáticos en Python te permiten realizar operaciones aritméticas y comparar valores. Ya hemos visto algunos ejemplos, pero aquí te detallo un poco más:

- Suma (+): Agrega dos valores.
- Resta (-): Resta un valor de otro.
- Multiplicación (*): Multiplica dos valores.
- División (/): Divide un valor entre otro.
- Módulo (%) : Calcula el residuo de una división.
- Exponenciación (**): Eleva un valor a una potencia.
- Igualdad (==): Compara si dos valores son iguales.
- Desigualdad (!=): Compara si dos valores son diferentes.
- Mayor que (>): Compara si un valor es mayor que otro.
- Menor que (<): Compara si un valor es menor que otro.
- Mayor o igual que (>=): Compara si un valor es mayor o igual que otro.
- Menor o igual que (<=): Compara si un valor es menor o igual que otro.

Recuerda que los operadores matemáticos se utilizan para realizar cálculos y comparar valores, mientras que los operadores de asignación se utilizan para asignar valores a las variables.



Impresión de Datos con print()

La función `print()` es una herramienta esencial en Python para mostrar información en la consola. Te permite mostrar texto, variables y resultados de cálculos. Para utilizarla, simplemente escribe `print()` seguido del valor que deseas mostrar entre paréntesis. Por ejemplo, `print("Hola mundo!")` mostrará la frase "Hola mundo!" en la consola.

También puedes utilizar `print()` para mostrar el valor de una variable. Por ejemplo, si tienes la variable `edad = 25`, puedes mostrar su valor con `print(edad)`, lo que mostrará "25" en la consola.

La función `print()` es muy versátil y te permitirá visualizar los resultados de tus programas y verificar si tu código está funcionando correctamente.

Estructuras Condicionales

Las estructuras condicionales te permiten ejecutar diferentes partes de código dependiendo de si una condición se cumple o no. En Python, la estructura condicional más común es el `if-else`.

La sintaxis es la siguiente:

```
if condicion:
  # Código a ejecutar si la condición es verdadera
else:
  # Código a ejecutar si la condición es falsa
```

Por ejemplo, si quieres verificar si un número es mayor que 10, puedes usar el siguiente código:

```
numero = 15
if numero > 10:
  print("El número es mayor que 10")
else:
  print("El número no es mayor que 10")
```

En este caso, la condición `numero > 10` es verdadera, por lo que se ejecutará el código dentro del bloque `if`. Las estructuras condicionales son fundamentales para crear programas con lógica y tomar decisiones basadas en las condiciones de tu código.

Manejo de Errores con try-except

En la programación, es común que se produzcan errores durante la ejecución del código. El bloque `try-except` te permite manejar estos errores de manera eficiente, evitando que el programa se detenga abruptamente.

La sintaxis es la siguiente:

```
try:
  # Código que puede generar un error
except Exception as error:
  # Código a ejecutar si se produce un error
```

Por ejemplo, si intentas dividir un número entre cero, se producirá un error. El bloque `try-except` te permite manejar este error y mostrar un mensaje al usuario:

```
try:
  resultado = 10 / 0
except ZeroDivisionError as error:
  print("Error: No se puede dividir entre cero.")
```

El bloque `try` contiene el código que puede generar un error. Si se produce un error, se ejecutará el bloque `except`, mostrando el mensaje de error. El manejo de errores es esencial para crear programas robustos que puedan manejar situaciones inesperadas y evitar que se detengan.

Conversión de Tipos de Datos (String a int)

En Python, los datos tienen diferentes tipos, como números enteros (int), números de punto flotante (float), cadenas de texto (str), y más. A veces, necesitas convertir un tipo de dato a otro para realizar operaciones específicas. Por ejemplo, puedes convertir una cadena de texto que representa un número a un número entero usando la función `int()`.

Si tienes la cadena `numero_str = "10"`, puedes convertirla a un número entero con `numero_int = int(numero_str)`. Ahora, `numero_int` tendrá el valor 10 como un entero. Esta conversión te permite realizar operaciones matemáticas con el valor.

Recuerda que debes asegurarte de que la cadena de texto sea una representación válida de un número para evitar errores.

Aplicaciones Prácticas

¡Ahora que hemos aprendido los fundamentos de Python, vamos a ver algunas aplicaciones prácticas de lo que hemos visto hasta ahora!

Ejemplo 1: Sumas Sencillas

Puedes usar las variables y los operadores matemáticos para realizar sumas simples en Python. Por ejemplo, si quieres sumar dos números, puedes escribir el siguiente código:

```
numero1 = 10
numero2 = 5
suma = numero1 + numero2
print(suma)
```

Este código mostrará "15" en la consola, ya que es el resultado de la suma de `numero1` y `numero2`.

Ejemplo 2: Condicionales Sencillos

Podemos usar una estructura condicional `if-else` para verificar si un número es par o impar. Si el número es divisible por 2, es par; de lo contrario, es impar. El código sería algo así:

```
numero = 12
if numero % 2 == 0:
  print("El número es par")
else:
  print("El número es impar")
```

Este código mostrará "El número es par" en la consola, ya que 12 es divisible por 2. ¡Así de fácil es usar los conocimientos que has aprendido para crear programas simples y prácticos!

Conclusión y Pasos a Seguir

¡Enhorabuena! Has dado tus primeros pasos en el mundo de la programación con Python. Has aprendido conceptos clave como variables, operadores, estructuras condicionales, manejo de errores y conversión de tipos de datos. Estos son los pasos a seguir para seguir aprendiendo y practicar:

1. Revisa el código que has aprendido y experiméntalo. Prueba diferentes valores y juega con el código para comprender mejor cómo funciona.
2. Investiga otros tipos de datos en Python, como números de punto flotante (float), cadenas de texto (str), listas (list) y diccionarios (dict). Comienza a explorar el poder de Python para manipular diferentes tipos de datos.
3. Practica la creación de programas simples. Intenta resolver problemas sencillos utilizando los conceptos que has aprendido. Puedes buscar ejemplos en línea o crear tus propios desafíos.
4. ¡No te desanimes! La programación es una habilidad que se desarrolla con la práctica. A medida que escribas más código, tu comprensión del lenguaje irá mejorando y te sentirás más cómodo con Python.

¡Bienvenido al emocionante mundo de la programación! Sigue explorando, practicando y divirtiéndote con Python. ¡El cielo es el límite!