Introducción al cómputo con Python

Introducción al cómputo con Python

Python es un lenguaje de programación de alto nivel, interpretado, de propósito general, multiplataforma y de código abierto. Es uno de los lenguajes de programación más populares del mundo, utilizado para una amplia gama de aplicaciones, desde ciencia de datos y aprendizaje automático hasta desarrollo web y creación de juegos.

En esta introducción, aprenderemos los fundamentos del cómputo con Python. Comenzaremos con una breve descripción del lenguaje y luego exploraremos los conceptos básicos de la programación, como variables, tipos de datos, operadores y expresiones. También veremos cómo escribir funciones y cómo usar módulos para organizar nuestro código.

¿Qué es Python?

Python es un lenguaje de programación interpretado, lo que significa que no necesita ser compilado antes de ejecutarse. Esto lo hace más fácil de aprender y usar que los lenguajes compilados, como C++ o Java.

Python es también un lenguaje de propósito general, lo que significa que se puede utilizar para una amplia gama de aplicaciones. Esto lo hace una buena opción para principiantes, ya que no es necesario aprender un lenguaje diferente para cada tipo de aplicación.

Fundamentos de la programación

En esta sección, aprenderemos los fundamentos de la programación con Python. Comenzaremos con una descripción de las variables, que se utilizan para almacenar datos en la memoria. Luego exploraremos los tipos de datos, que definen el tipo de datos que se pueden almacenar en una variable.

También veremos cómo usar operadores y expresiones para manipular datos. Los operadores son símbolos que se utilizan para realizar operaciones matemáticas o lógicas. Las expresiones son combinaciones de variables, operadores y valores que se evalúan para obtener un resultado.

Variables

En Python, las variables son nombres que se utilizan para referirse a valores almacenados en la memoria. Las variables se crean cuando se les asigna un valor por primera vez. Para asignar un valor a una variable se utiliza el operador de igualdad (=).

Por ejemplo, el siguiente código crea una variable llamada `x` y le asigna el valor 10:

```
x = 10
```

En este ejemplo, el nombre de la variable es `x` y el valor asignado es 10. El tipo de datos de la variable `x` es `int`, que representa un número entero.

Las variables se pueden usar para almacenar cualquier tipo de dato, incluidos números, cadenas, listas, tuplas, diccionarios y objetos.

Nombres de variables

Los nombres de las variables en Python deben cumplir con las siguientes reglas:

- El nombre debe comenzar con una letra o guión bajo (_).
- El nombre puede contener letras, números y guiones bajos.
- El nombre no puede contener espacios.
- El nombre no puede ser una palabra clave reservada de Python.

Tipos de datos

Los tipos de datos definen el tipo de datos que se pueden almacenar en una variable. Python tiene una variedad de tipos de datos, incluidos:

- `int`: números enteros
- `float`: números decimales
- `str`: cadenas de texto
- `list`: listas de valores
- `tuple`: tuplas de valores
- `dict`: diccionarios de claves y valores
- `object`: objetos

Operaciones con variables

Las variables se pueden usar en operaciones matemáticas, lógicas y de comparación. Por ejemplo, el siguiente código suma los valores de las variables `x` e `y`:

```
x = 10
y = 20

z = x + y

print(z)
```

Este código imprime el siguiente resultado:

```
30
```

Ejemplos de variables

Aquí hay algunos ejemplos de variables en Python:

Nombre de variable	Tipo de dato	Valor
x	int	10

Nombre de variable	Tipo de dato	Valor
y	float	3.14
name	str	"John Doe"
numbers	list	[1, 2, 3, 4, 5]
coordinates	tuple	(1, 2, 3)
person	dict	{"name": "John Doe", "age": 30}
car	object	<class 'Car'>

Funciones

En Python, las funciones son bloques de código que se pueden reutilizar. Las funciones se definen usando la palabra clave `def` seguida del nombre de la función, los parámetros de la función y el código de la función.

Estructura de una función

La estructura básica de una función en Python es la siguiente:

```
def nombre_funcion(parametros):
    # Código de la función
    return valor_retorno
```

Nombre de la función

El nombre de la función debe ser un identificador válido de Python.

Parámetros

Los parámetros son variables que se pasan a la función como entrada. Los parámetros se definen entre paréntesis después del nombre de la función.

Código de la función

El código de la función es el conjunto de instrucciones que se ejecutan cuando se llama a la función.

Valor de retorno

El valor de retorno es el valor que se devuelve de la función. El valor de retorno se especifica usando la palabra clave `return`.

Ejemplo de función

El siguiente ejemplo define una función llamada `suma()` que suma dos números:

```
def suma(x, y):
    return x + y
```

```
print(suma(10, 20))
```

Este código imprime el siguiente resultado:

```
30
```

Ejemplos de funciones

Aquí hay algunos ejemplos de funciones en Python:

```
# Función que imprime un mensaje
def mensaje():
    print("Hola, mundo!")

mensaje()

# Función que calcula el área de un círculo
def area_circulo(r):
    return 3.14 * r * r

print(area_circulo(10))

# Función que devuelve el factorial de un número
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

print(factorial(5))
```

Clases

Explicación de las clases en Python

En Python, las clases son una forma de crear nuevos tipos de datos. Las clases se definen usando la palabra clave `class` seguida del nombre de la clase, los atributos de la clase y los métodos de la clase.

Estructura de una clase

La estructura básica de una clase en Python es la siguiente:

```
class nombre_clase:
    # Atributos de la clase
    # Métodos de la clase
```

Atributos de la clase

Los atributos de la clase son variables que se comparten entre todas las instancias de la clase. Los atributos de la clase se definen usando la palabra clave `self` seguida del nombre del atributo.

Métodos de la clase

Los métodos de la clase son funciones que se asocian a una clase. Los métodos de la clase se definen usando la palabra clave `def` seguida del nombre del método, los parámetros del método y el código del método.

Ejemplo de clase

El siguiente ejemplo define una clase llamada `Persona` que tiene dos atributos, `nombre` y `edad`, y un método, `saludar()`:

```
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def saludar(self):
        print(f"Hola, mi nombre es {self.nombre} y tengo {self.edad} años.")

persona = Persona("Juan Pérez", 30)
persona.saludar()
```

Este código imprime el siguiente resultado:

```
Hola, mi nombre es Juan Pérez y tengo 30 años.
```

Ejemplos de clases

Aquí hay algunos ejemplos de clases en Python:

```
# Clase que representa un círculo
class Circulo:
    def __init__(self, radio):
        self.radio = radio

    def area(self):
        return 3.14 * self.radio * self.radio

circulo = Circulo(10)
print(circulo.area())

# Clase que representa un libro
class Libro:
    def __init__(self, titulo, autor, paginas):
```

```
        self.titulo = titulo
        self.autor = autor
        self.paginas = paginas

    def leer(self):
        for pagina in range(self.paginas):
            print(f"Página {pagina + 1}")

libro = Libro("El Quijote", "Miguel de Cervantes", 100)
libro.leer()
```

Explicación

Las clases se utilizan para crear nuevos tipos de datos que pueden usarse para representar objetos del mundo real. Los atributos de la clase representan las propiedades de un objeto, y los métodos de la clase representan las acciones que puede realizar un objeto.

En el ejemplo de la clase `Persona`, los atributos `nombre` y `edad` representan las propiedades de una persona. El método `saludar()` representa la acción de saludar a alguien.

Las clases son una herramienta poderosa que se puede usar para crear aplicaciones complejas.

Términos clave

- Clase: una plantilla para crear objetos
- Atributo: una variable que se asocia a una clase o a un objeto
- Método: una función que se asocia a una clase o a un objeto
- Instancia: un objeto creado a partir de una clase

Ejercicios

- Define una clase llamada `Coche` que tenga los atributos `marca`, `modelo` y `color`.
- Define un método llamado `arrancar()` que haga que el coche arranque.
- Crea una instancia de la clase `Coche` y llama al método `arrancar()`.

Control de flujo

El control de flujo es una parte fundamental de la programación, ya que permite controlar el orden en el que se ejecutan las instrucciones de un programa.

En Python, existen dos tipos básicos de control de flujo:

- **Control condicional:** permite ejecutar un bloque de instrucciones solo si se cumple una condición.
- **Control iterativo:** permite ejecutar un bloque de instrucciones repetidamente hasta que se cumpla una condición.

Control condicional

El control condicional se utiliza para ejecutar un bloque de instrucciones solo si se cumple una condición. En Python, la condición se evalúa como un valor booleano.

Existen dos tipos básicos de control condicional en Python:

- **Sentencia if:** permite ejecutar un bloque de instrucciones si la condición se evalúa como `True`.
- **Sentencia elif:** permite ejecutar un bloque de instrucciones si la condición se evalúa como `True` y no se ha ejecutado ningún bloque de instrucciones if o elif anterior.
- **Sentencia else:** permite ejecutar un bloque de instrucciones si ninguna de las condiciones anteriores se evalúa como `True`.

Ejemplo

El siguiente código utiliza una sentencia if para imprimir un mensaje solo si la variable `edad` es mayor de 18:

```
edad = 20

if edad > 18:
    print("Eres mayor de edad.")
```

Este código imprimirá el siguiente mensaje:

```
Eres mayor de edad.
```

Control iterativo

El control iterativo se utiliza para ejecutar un bloque de instrucciones repetidamente hasta que se cumpla una condición. En Python, existen dos tipos básicos de control iterativo:

- **Bucle for:** permite iterar sobre una colección de elementos.
- **Bucle while:** permite ejecutar un bloque de instrucciones repetidamente mientras se cumpla una condición.

Bucle for

El bucle for se utiliza para iterar sobre una colección de elementos. La colección puede ser una lista, una tupla, un diccionario o un conjunto.

Ejemplo

El siguiente código utiliza un bucle for para imprimir los números del 1 al 10:

```
for numero in range(1, 11):
    print(numero)
```

Este código imprimirá los siguientes números:

```
1
2
```

```
3
4
5
6
7
8
9
10
```

Bucle while

El bucle while se utiliza para ejecutar un bloque de instrucciones repetidamente mientras se cumpla una condición.

Ejemplo

El siguiente código utiliza un bucle while para imprimir los números del 1 al 10:

```
numero = 1

while numero <= 10:
    print(numero)
    numero += 1
```

Este código imprimirá los mismos números que el código anterior.

Términos clave

- **Control de flujo:** permite controlar el orden en el que se ejecutan las instrucciones de un programa.
- **Control condicional:** permite ejecutar un bloque de instrucciones solo si se cumple una condición.
- **Control iterativo:** permite ejecutar un bloque de instrucciones repetidamente hasta que se cumpla una condición.
- **Sentencia if:** permite ejecutar un bloque de instrucciones si la condición se evalúa como `True`.
- **Sentencia elif:** permite ejecutar un bloque de instrucciones si la condición se evalúa como `True` y no se ha ejecutado ningún bloque de instrucciones if o elif anterior.
- **Sentencia else:** permite ejecutar un bloque de instrucciones si ninguna de las condiciones anteriores se evalúa como `True`.
- **Bucle for:** permite iterar sobre una colección de elementos.
- **Bucle while:** permite ejecutar un bloque de instrucciones repetidamente mientras se cumpla una condición.

Ejercicios

- Escribe un programa que use una sentencia if para imprimir un mensaje si el usuario ingresa un número mayor que 10.
- Escribe un programa que use un bucle for para imprimir los números del 1 al 100.
- Escribe un programa que use un bucle while para imprimir los números del 1 al 100, pero solo los números pares.

Estructuras de datos

Las estructuras de datos son una parte fundamental de la programación, ya que permiten almacenar y organizar datos de manera eficiente. En Python, existen cuatro estructuras de datos comunes:

- **Listas:** las listas son una secuencia de elementos que pueden ser de cualquier tipo de dato.
- **Tuplas:** las tuplas son una secuencia de elementos que no se pueden modificar.
- **Diccionarios:** los diccionarios son una colección de pares clave-valor.
- **Conjuntos:** los conjuntos son una colección de elementos únicos.

Listas

Las listas son una de las estructuras de datos más comunes en Python. Las listas se definen usando corchetes []. Los elementos de una lista pueden ser de cualquier tipo de dato, incluso listas, tuplas, diccionarios y conjuntos.

Ejemplo

```
numeros = [1, 2, 3, 4, 5]
nombres = ["Juan", "Pedro", "María"]

print(numeros)
# [1, 2, 3, 4, 5]

print(nombres)
# ["Juan", "Pedro", "María"]
```

Tuplas

Las tuplas son similares a las listas, pero no se pueden modificar. Las tuplas se definen usando paréntesis (). Los elementos de una tupla pueden ser de cualquier tipo de dato, incluso listas, tuplas, diccionarios y conjuntos.

Ejemplo

```
numeros = (1, 2, 3, 4, 5)
nombres = ("Juan", "Pedro", "María")

print(numeros)
# (1, 2, 3, 4, 5)

print(nombres)
# ("Juan", "Pedro", "María")
```

Diccionarios

Los diccionarios son una estructura de datos que se utiliza para almacenar pares clave-valor. Las claves de un diccionario deben ser únicas, pero los valores pueden ser de cualquier tipo de dato.

Ejemplo

```
persona = {"nombre": "Juan", "edad": 30}

print(persona)
# {"nombre": "Juan", "edad": 30}

print(persona["nombre"])
# Juan

print(persona["edad"])
# 30
```

Conjuntos

Los conjuntos son una estructura de datos que se utiliza para almacenar elementos únicos. Los elementos de un conjunto no tienen un orden específico.

Ejemplo

```
numeros = {1, 2, 3, 4, 5}
nombres = {"Juan", "Pedro", "María"}

print(numeros)
# {1, 2, 3, 4, 5}

print(nombres)
# {"Juan", "Pedro", "María"}
```

Términos clave

- **Estructura de datos:** una forma de almacenar y organizar datos.
- **Lista:** una secuencia de elementos que pueden ser de cualquier tipo de dato.
- **Tupla:** una secuencia de elementos que no se pueden modificar.
- **Diccionario:** una colección de pares clave-valor.
- **Conjunto:** una colección de elementos únicos.

Ejercicios

- Crea una lista que contenga los números del 1 al 10.
- Crea una tupla que contenga los nombres de tus amigos.
- Crea un diccionario que contenga la edad de tus amigos.
- Crea un conjunto que contenga los colores del arcoíris.

Resumen

Los conceptos vistos hasta aquí son los siguientes:

- **Variables:** son nombres que se utilizan para referirse a valores almacenados en la memoria.
- **Funciones:** son bloques de código que se pueden reutilizar.
- **Clases:** son una forma de crear nuevos tipos de datos.
- **Control de flujo:** permite controlar el orden en el que se ejecutan las instrucciones de un programa.
- **Estructuras de datos:** son una forma de almacenar y organizar datos.

Estos conceptos son fundamentales para la programación en Python. Una vez que los domines, podrás empezar a escribir programas más complejos y eficientes.

Aquí hay un resumen de cada concepto:

Variables

- Las variables se definen usando el operador ``=.
- El tipo de datos de una variable se determina por el valor que se le asigna.
- Las variables se pueden usar para almacenar cualquier tipo de dato, incluidos números, cadenas, listas, tuplas, diccionarios y conjuntos.

Funciones

- Las funciones se definen usando la palabra clave `def`.
- El cuerpo de una función se define entre llaves `{}`.
- Las funciones se pueden llamar usando el operador `()`.

Clases

- Las clases se definen usando la palabra clave `class`.
- Los atributos de una clase se definen usando la palabra clave `self`.
- Los métodos de una clase se definen usando la palabra clave `def`.
- Las instancias de una clase se crean usando el operador `()`.

Control de flujo

- El control condicional permite ejecutar un bloque de instrucciones solo si se cumple una condición.
- El control iterativo permite ejecutar un bloque de instrucciones repetidamente hasta que se cumpla una condición.

Estructuras de datos

- Las listas son una secuencia de elementos que pueden ser de cualquier tipo de dato.
- Las tuplas son una secuencia de elementos que no se pueden modificar.
- Los diccionarios son una colección de pares clave-valor.
- Los conjuntos son una colección de elementos únicos.