Python para Principiantes

Este es la primera parte de un tutorial de prueba inspirado en Learn X in Y Minutes: Python.

Variables y tipos de datos

Asignación de variables

En Python, se utilizan igualdad (`=`) para asignar un valor a una variable.

```
x = 5  # asigna 5 a x
y = "hello"  # asigna la cadena "hello" a y
```

Tipos de datos

Python tiene varios tipos de datos integrados:

- `int`: números enteros, como 1, 2, 3, etc.
- `float`: números decimales, como 1.0, 2.5, etc.
- `str`: cadenas de texto, como “hello”, ‘hello’, etc.
- `bool`: valores booleanos, como True o False
- `list`: listas, como [1, 2, 3], [“a”, “b”, “c”], etc.
- `tuple`: tuplas, como (1, 2, 3), (“a”, “b”, “c”), etc.
- `dict`: diccionarios, como {“name”: “John”, “age”: 30}, etc.

Operadores

Aritméticos

Python admite los siguientes operadores aritméticos:

- `+`: suma
- `-`: resta
- `/`: división
- `*`: multiplicación
- `%`: módulo (resto de la división)
- `**`: exponente

```
x = 5
y = 3
print(x + y)  # imprime 8
print(x - y)  # imprime 2
print(x * y)  # imprime 15
print(x / y)  # imprime 1.6666666666666667
print(x % y)  # imprime 2
print(x ** y)  # imprime 125
```

Comparación

Python admite los siguientes operadores de comparación:

- `==`: igual
- `!=`: no igual
- `>`: mayor que
- `<`: menor que
- `>=` : mayor o igual
- `<=` : menor o igual

```
x = 5
y = 3
print(x == y)  # imprime False
print(x != y)  # imprime True
print(x > y)  # imprime True
print(x < y)  # imprime False
print(x >= y)  # imprime True
print(x <= y)  # imprime False
```

Control de flujo

Condicional if

La estructura de control de flujo `if` se utiliza para ejecutar un bloque de código si se cumple una condición.

```
x = 5
if x > 10:
    print("x es mayor que 10")
else:
    print("x es menor o igual que 10")  # imprime "x es menor o igual que 10"
```

Bucle for

La estructura de control de flujo `for` se utiliza para iterar sobre una secuencia (como una lista o una cadena).

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
# imprime:
# apple
# banana
# cherry
```

Funciones

Las funciones permiten agrupar un bloque de código que se puede reutilizar varias veces.

```
def greet(name):
    print("Hello, " + name + "!")

greet("John")  # imprime "Hello, John!"
```

Módulos

Los módulos permiten agrupar funciones y variables relacionadas en un solo archivo.

```
import math
print(math.pi)  # imprime 3.14159265359
```

Bonus

Listas

Las listas son colecciones ordenadas de elementos.

```
numbers = [1, 2, 3, 4, 5]
print(numbers[0])  # imprime 1
print(numbers[-1])  # imprime 5
numbers.append(6)  # agrega 6 al final de la lista
print(numbers)  # imprime [1, 2, 3, 4, 5, 6]
```

Diccionarios

Los diccionarios son colecciones de pares clave-valor.

```
person = {"name": "John", "age": 30}
print(person["name"])  # imprime "John"
print(person["age"])  # imprime 30
person["city"] = "New York"  # agrega una nueva clave-valor
print(person)  # imprime {"name": "John", "age": 30, "city": "New York"}
```