2. Fundamentos

Índice

1. Comentarios
2. Variables
3. Tipos de datos
4. Operadores
5. Conversión de tipos
6. Manejo de cadenas
7. Estructuras de control de flujo

1. Comentarios

Los comentarios sirven para hacer anotaciones al código que no serán procesadas por el intérprete:

```
# Esto es un comentario de una sola línea

"""
Y esto otro
de varias líneas
"""
```

2. Variables

Esto son (asignaciones de) variables:

```
name = 'Espagueti'
lastname = "Volador"
age = 16
```

Se puede mostrar su valor:

```
print(name)
```

Y su tipo:

```
print(type(name))
```

Se puede capturar un valor y asignárselo:

```
name = input("Escribe tu nombre: ")
print(name)
```

Hacer asignaciones múltiples:

```
scorePlayer1 = scorePlayer2 = scorePlayer3 = 5
```

Ejemplos de nombres válidos de variables:

```
fullname = "Simone de Beauvoir"
full_name = "Miyamoto Mushashi"
_full_name = "Ada Lovelace"
fullName = "Richard Feynman"
FULLNAME = "Clara Campoamor"
fullname2 = "Siddartha Gautama"
```

3. Tipos de datos

Python presenta todos estos tipos primitivos de datos:

- str (string):

  ```
  "Abracadabra pata de cabra"
  ```

- int (Integer):

  ```
  42
  ```

- float (Float):

  ```
  3.14159
  ```

- complex (Complex):

  ```
  9 + 3j
  ```

- bool (Boolean):

```
True
False
```

- None (sin valor)

4. Operadores

Dependiendo del tipo de datos, este soportará unos operadores u otros:

- Los *strings*, concatenación:

```
fullname = "Anonymous"
age = 25

# Con el operador:
print("Hola, " + fullname + " tienes " + age + " años")

# Con la notación f-string:
print(f"Adiós, {fullname}, tienes {age} años")
```

- Los numéricos ofrecen operadores aritméticos:
  - Suma: `1 + 2`
  - Resta: `3 - 4`
  - Multiplicación: `6 * 7`
  - División: `10 / 3`
  - Potencia: `2 ** 5`
  - Módulo: `10 % 3`
  - División entera: `10 // 3`
- Los *booleans*, operadores lógicos:
  - Conjunción:

    ```
    True and False
    ```

  - Disyunción:

    ```
    True or False
    ```

  - Negación:

    ```
    not True
    ```

5. Conversión de tipos

Es posible convertir variables de un tipo a otro si es necesario:

```
intValue = 1
floatValue = 2.0
strValue = "34"

floatValueFromInt = float(intValue)
intValueFromFloat = int(floatValue)
intValueFromStr = int(strValue)
```

6. Manejo de cadenas

Python permite strings multilínea:

```
myText = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
```

Para saber la longitud de un string, existe la función `len`:

```
print(len(myText))
```

Se puede comprobar si un *substring* está contenido en otro string con el operador `in`:

```
print("tempor" in myText)
print("totum" in myText)
```

Permite escapar caracteres para que no sean interpretados:

```
boss = "Me dijo que era un tanto \"impredecible\""
print(boss)
```

O usar caracteres especiales:

```
\\ \n \r \t \b
```

A no ser que vayan precedidos por el prefijo de literalidad:

```
print(r"Esto sale tal cual: \n")
```

Se pueden recorrer:

```
for x in "banana":
  print(x)
```

O también obtener un substring de otro:

```
b = "Hola, Mundo"
print(b[:5])
print(b[2:])
```

7. Estructuras de control de flujo

Para las estructuras de control de flujo se usan operadores lógicos (citados más arriba) y operadores de comparación:

```
== != < <= > >=
```

Condicionales

```
a = 1
b = 2

if b > a:
  print("b es mayor que a")
```

Puede haber condicionales de dos o más ramas:

```
a = 1
b = 2
c = 3

if a > b:
  print("a es mayor que b")
else:
  print("b es mayor que a")
```

```
if a > c:
  print("a es mayor que c")
elif a > b:
  print("a es mayor que b")
else:
  print("a es el más pequeño de todos")
```

Bucles

Un bucle debe tener tres elementos:

- Inicialización previa
- Condición de parada
- Cambios en cada iteración

While

```
i = 0

while i < 5:
  print(i)
  i += 1
```

For .. in

```
for i in range(5):
  print(i)

for letter in "abracadabra":
  print(letter)
```

Ambos tipos de bucles permiten el uso de `continue` y `break` para romper el control de flujo:

```
i = 1
while i < 5:
  print(i)
  i += 1

  if i == 3:
    break

fruits = ["cereza", "pomelo", "coco"]

for f in fruits:
```

```
    if f == "pomelo":
      continue

    print(f)
```

Referencias

Comentarios
Variables
Tipos de datos
Cadenas
Números
Lógicos
Operadores
Condicionales
Bucles While
Bucles For

Python Tutor