

Checkpoint 3

Preguntas teóricas:

- ¿Cuáles son los tipos de Datos en Python?

Booleans

Tipo de dato que representa valores de verdad, es decir, True o False.

```
es_mayor = True
print(type(es_mayor)) # <class 'bool'>
print(5 > 3) # True
```

Numbers

Representa valores numéricos y se divide en int (enteros), float (decimales) y complex (números complejos).

```
num_entero = 10
num_decimal = 3.14
num_complejo = 2 + 3j
print(type(num_entero), type(num_decimal), type(num_complejo))
```

Strings

Secuencia de caracteres representada entre comillas simples o dobles.

```
texto = "Hola, Python"
print(type(texto)) # <class 'str'>
```

Bytes and Byte Arrays

Datos binarios inmutables (bytes) o mutables (bytearray).

```
dato_bytes = b"Hola"
dato_bytearray = bytearray(5)
print(type(dato_bytes), type(dato_bytearray))
```

None

Representa la ausencia de un valor o un valor nulo en Python.

```
valor_nulo = None
print(type(valor_nulo)) # <class 'NoneType'>
```

Lists

Colección ordenada y mutable de elementos, que pueden ser de distintos tipos.

```
lista = [1, 2, 3, "Python", 4.5]
print(type(lista)) # <class 'list'>
```

Tuples

Colección ordenada e inmutable de elementos.

```
tupla = (10, 20, "Hola")
print(type(tupla)) # <class 'tuple'>
```

Sets

Colección desordenada de valores únicos, sin elementos duplicados.

```
conjunto = {1, 2, 3, 4, 4, 5}
print(type(conjunto)) # <class 'set'>
print(conjunto) # {1, 2, 3, 4, 5}
```

Dictionaries

Colección de pares clave-valor, donde cada clave es única.

```
diccionario = {"nombre": "Juan", "edad": 25, "ciudad": "Madrid"}
print(type(diccionario)) # <class 'dict'>
print(diccionario["nombre"]) # Juan
```

- **¿Qué tipo de convención de nomenclatura deberíamos utilizar para las variables en Python?**

Siguiendo el estándar PEP 8, que es la guía oficial de estilo de Python, la nomenclatura recomendada es la `snake_case`.

En la cual se usa **minúsculas** y se separan las palabras con **guiones bajos**.

```
nombre_usuario = "Juan"
edad_persona = 25
```

- **¿Qué es un Heredoc en Python?**

Es una forma de definir cadenas de texto en múltiples líneas sin necesidad de usar `\n` en cada línea. Se pueden utilizar tanto comillas simples triples (`'''`) como comillas dobles triples (`"""`).

```
texto_multilinea = """Esto es un ejemplo
de un string multilínea,
similar a un heredoc en otros lenguajes."""
print(texto_multilinea)
```

- **¿Qué es una interpolación de cadenas?**

La interpolación de cadenas es una técnica que permite insertar valores dentro de una cadena de texto de manera dinámica, por ejemplo, usando f-strings.

```
nombre = "Juan"
edad = 25
mensaje = f"Hola, mi nombre es {nombre} y tengo {edad} años."
print(mensaje)
```

- **¿Cuándo deberíamos usar comentarios en Python?**

Los comentarios en Python se deben usar para explicar código complejo o poco intuitivo, documentar funciones, clases y módulos mediante docstrings, marcar partes del código que requieren mejoras o revisiones y desactivar temporalmente partes del código durante pruebas. Pero, no se deben usar para comentar lo obvio, ni abusar de los comentarios en exceso. La regla es escribir código claro y utilizar comentarios solo cuando realmente sean necesarios para mejorar la comprensión.

- **¿Cuáles son las diferencias entre aplicaciones monolíticas y de microservicios?**

Las aplicaciones monolíticas son estructuras únicas donde todas las funcionalidades están integradas y se despliegan como un solo bloque. Son más simples al principio, pero difíciles de escalar y mantener a medida que crecen. En cambio, las aplicaciones de microservicios están divididas en servicios independientes que se comunican entre sí, lo que permite escalabilidad, actualizaciones y mantenimiento más fáciles. Sin embargo, requieren una infraestructura más compleja y gestión eficiente de la comunicación entre servicios.