MC2C3

1. ¿Cuáles son los tipos de Datos en Python?

Son como categorías en las que se clasifican los valores que podemos usar en un programa. Estos determinan qué operaciones se pueden realizar con los datos y cómo se almacenan en la memoria.

TIPOS DE DATOS BÁSICOS

Números (Numbers) int, float

- Enteros (int) Números sin decimales
  edad = 49
- Decimales (float) Números con decimales
  precio = 9.50

Cadenas de texto/Strings (str) secuencias de caracteres entre comillas (“ o ‘)

- Nombre = “Ione”
  mensaje = ‘Hola, ¿Cómo estás?’

Buena práctica: Comillas dobles (“) para cadenas que contienen comillas simples (‘) Ejm:

- sentence = “ Le dijo: ‘Hola’”

str.upper() Convierte los caracteres de una cadena a mayúsculas. Se puede combinar con **strip()** (elimina espacios y saltos de linea) o **replace()** (reemplazar texto). No modifica la cadena original, solo devuelve una nueva.

str.capitalize() Convierte el primer carácter de la cadena a mayúscula y el resto a minúsculas. No modifica la cadena original, devuelve una nueva. Para formatear nombre propios, mensajes o textos.

str.lower() Convierte todos los caracteres de una cadena a minúsculas.

.title() Convierte la primera letra de cada palabra dentro de la cadena en mayúsculas.

Booleans (bool) representan valores: True o False
Para condiciones y comparaciones (**if else**).

Colecciones list, tuple, set, dict.

Bytes: bytes, byte array

Ninguno: None
Dato especial que representa la ausencia de valor o un valor nulo.
Uso principal: Indicar que una variable no tiene un valor asignado o que una función no devuelve nada.

- Inicializar una variable sin valor

Se puede usar para definir una variable sin asignar valor inicial. Cuando no sabemos qué valor tendrá la variable más adelante.
Resultado = None # No sabemos que valor tendrá.

Se puede asignar un valor más tarde:
resultado = 42 # Valor asignado

- Valor de retorno en funciones.

Si una función no devuelve un valor, Python devuelve None por defecto.

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

snake_case: Usar letras minúsculas y separar las palabras con guiones bajos (_).
Ejm: nombre_usuario, edad_persona, total_ventas.

Se recomienda el uso de nombres descriptivos que indiquen claramente el propósito de la variable y mantener el mismo estilo en todo el código.

3. **¿Qué es un Heredoc en Python?**

Es una forma de definir las cadenas de texto largas o multilinea en un lenguaje de programación de manera sencilla y legible, preservando saltos de línea y formato. En Python se usan las triples comillas (“””) o (‘’’).

Ejm Heredoc en Python:

texto_largo = “””
blablablablablablablablablabla.
blablablablablablablablablabla.
blablablablablablablablablabla.
“””
print(texto_largo)
blablablablablablablablablabla.
Blablablablablablablablablabla.
Blablablablablablablablablabla.

.strip(“\n) Elimina saltos de línea al principio y al final.

.strip() Elimina saltos de línea y espacios en blanco al principio y al final.

.lstrip() Elimina saltos de linea y espacios al principio.

.rstrip() Elimina saltos de línea y espacios al final.

.repr() Devuelve el texto tal cual lo representa la computadora para así estar seguros de dónde están los saltos de línea. Cada salto esta representado por \n (new line character) y así se puede ver como se generan los Heredocs.

\nblablablablablablablablabla\n
blablablablablablablablablabla\n
blablablablablablablablablabla\n

4. **¿Qué es una interpolación de cadenas?**

Es una manera de insertar valores a variables o expresiones dentro de una cadena de texto para no tener que unir estas manualmente.

f-string (Introducida en Python 3.6) es la manera más recomendada de hacer interpolación de cadenas. Se agrega f antes de la cadena y se usan curly brackets {} para insertar variables o expresiones.

Ejm:
nombre = “Ione”
edad = “49”
mensaje = f”Hola, {nombre}. Tienes {edad} años.”
print(mensaje) # Hola, Ione. Tienes 49 años.

.format() Usa llaves {} como marcadores de posición y luego pasas los valores con .format(), alternativa más antigua pero en uso.
Ejm:
nombre = “Ione”
edad = “49”
mensaje = “Hola, {}. Tienes {} años.”.format(nombre, edad)
print(mensaje) # Hola, Ione. Tienes 49 años.

5. **¿Cuándo deberíamos usar comentarios en Python?**

- Explicar el propósito del código, si no es obvio a simple vista.
- Cuando se toma una decisión especifica, explica el por qué.
- Para documentar funciones, clases o módulos.
- Código temporal o depuración.
- Desactivar código temporalmente.

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

Monolítica: Fácil de desarrollar pero difícil de escalar. Ejm:

- Tienda en línea donde el frontend, backend y base de datos están en un solo servidor.
- Si el servidor falla, la tienda deja de funcionar.

Microservicios: Servicios independientes, escalables pero más complejos. Ejm: tienda en linea donde:

- Un servicio maneja la autentificación.
- Otro servicio gestiona el catálogo de productos.
- Otro servicio procesa los pagos.

Si el servicio de pagos falla, el resto de la tienda sigue funcionando.