CHECKPOINT 3

¿Cuáles son los tipos de Datos en Python?

Primitivos

- int – Enteros (ejemplo: 10)
- float – Números decimales (ejemplo: 3.14)
- bool – Booleanos: True o False
- str – Cadenas de texto (ejemplo: "Hola")
- NoneType – Valor nulo (None)

Compuestos

- list – Lista ordenada y mutable (ejemplo: [1, 2, 3])
- tuple – Tupla ordenada e inmutable (ejemplo: (1, 2))
- set – Conjunto sin duplicados (ejemplo: {1, 2, 3})
- dict – Diccionario con pares clave-valor (ejemplo: {"a": 1})

Otros

- range – Secuencia numérica (ejemplo: range(5))
- bytes, bytearray, memoryview – Datos binarios
- complex – Números complejos (ejemplo: 2 + 3j)

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

Las variables y funciones se escriben en snake_case, usando letras minúsculas y separando palabras con guiones bajos. Las constantes se escriben en mayúsculas con guiones bajos para separar palabras. Las clases utilizan PascalCase, comenzando cada palabra con mayúscula sin separadores.

Ejemplos:

```
# Variables y funciones

velocidad_angular = 120.5

def calcular_momento():


# Constantes

GRAVEDAD = 9.81


# Clases
```

class MotorElectrico:

¿Qué es un Heredoc en Python?

Python no tiene una característica heredoc como otros lenguajes, pero logra la misma funcionalidad usando triple comillas (""" o '''). Esta sintaxis permite crear cadenas multilínea que preservan exactamente el formato original, incluyendo saltos de línea y espacios. Es útil para documentación, plantillas de texto o cualquier texto que necesite mantener su estructura.

Ejemplos:

especificacion = """

Motor: 1500 RPM

Torque: 45 Nm

Voltaje: 220V AC

"""

¿Qué es una interpolación de cadenas?

La interpolación de cadenas permite insertar valores de variables o expresiones directamente dentro de un string de forma dinámica. Python ofrece varias técnicas: f-strings (la más moderna y eficiente), el método format(), y el formateo con %.

Ejemplos:

potencia = 750

eficiencia = 0.85

 # F-strings

*reporte = f"Potencia: {potencia}W, Eficiencia: {eficiencia*100}%"*

¿Cuándo deberíamos usar comentarios en Python?

Los comentarios deben usarse para mejorar la comprensión del código. Son útiles para explicar lógica compleja, documentar decisiones de diseño, advertir sobre comportamientos especiales, y proporcionar contexto que no es obvio del código. De esta forma, los desarrolladores también pueden utilizar los métodos y las clases sin conocer su funcionamiento interno.

Sin embargo, debe evitarse comentar código auto explicativo. Un buen código debería ser legible por sí mismo mediante nombres descriptivos de variables y funciones.

Ejemplos:

 # Comentarios útiles

 # ADVERTENCIA: Temperatura máxima 80°C

def control_motor():

 # Comentarios innecesarios

contador += 1 # Incrementar contador

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

Las aplicaciones monolíticas son sistemas construidos como una sola unidad desplegable, donde todos los componentes están interconectados y se ejecutan en el mismo proceso. Por el contrario, los microservicios son una arquitectura donde la aplicación se divide en servicios pequeños e independientes que se comunican a través de la red.

Ejemplos:

Arquitectura Monolítica: Una tienda online construida como una aplicación única donde todos los componentes (usuarios, productos, pagos, inventario) están integrados en un solo proyecto. Cuando se necesita actualizar cualquier funcionalidad, se debe desplegar toda la aplicación completa.

Arquitectura de Microservicios: La misma tienda se divide en servicios independientes, cada uno con su responsabilidad específica y base de datos propia. Cada servicio se puede actualizar y desplegar por separado sin afectar al resto del sistema.