CHECKPOINT 3 – PREGUNTAS TEÓRICAS

1. Cuales son los tipos de Datos en Python?

- **Números**:
    - **Enteros** (int): Números sin parte decimal, por ejemplo, 4, -2.
    - **Flotantes** (float): Números con parte decimal, por ejemplo, 3.14, -0.001.
    - **Complejos** (complex): Números con parte real e imaginaria, por ejemplo, 1+2j.
- **Cadenas de texto** (str):
    - Secuencias de caracteres, por ejemplo, "Hola, Txufi!", "El saxo es maravilloso".
- **Booleanos** (bool):
    - Valores de verdad, True o False.
- **Listas** (list):
    - Colecciones ordenadas y mutables de elementos, por ejemplo, [1, 2, 3], ["a", "b", "c"].
- **Tuplas** (tuple):
    - Colecciones ordenadas e inmutables de elementos, por ejemplo, (1, 2, 3), ("a", "b", "c").
- **Conjuntos** (set):
    - Colecciones desordenadas de elementos únicos, por ejemplo, {1, 2, 3}, {"a", "b", "c"}.
- **Diccionarios** (dict):
    - Colecciones de pares clave-valor, por ejemplo, {"nombre": "Juan", "edad": 30}.
- **NoneType**:
    - Representa la ausencia de valor, None.

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

La convención: PEP 8, como sigue:

- **Nombres de variables**:
    - Usa **minúsculas** y separa las palabras con guiones bajos (_).
    - Ejemplo:una_variable, cantidad_de_perros.
- **Nombres de funciones**:
    - Similar a las variables, usa **minúsculas** y guiones bajos.
    - Ejemplo: calcular_promedio(), obtener_datos().
- **Nombres de clases**:
    - Usa el **Estilo CamelCase**, donde cada palabra empieza con una letra mayúscula y no se usan guiones bajos.
    - Ejemplo: ClaseEjemplo, UsuarioAdministrador.
- **Nombres de constantes**:
    - Usa **mayúsculas** y separa las palabras con guiones bajos.
    - Ejemplo: PI, MAXIMO_VALOR.
- **Módulos y paquetes**:
    - Usa **minúsculas** y guiones bajos para los nombres de módulos.
    - Ejemplo: mi_modulo.py, paquete_utilidades.

3. Qué es un Heredoc en Python?

Un Heredoc es una forma de definir una cadena de texto que puede abarcar múltiples líneas, se pueden utilizar cadenas de texto multilínea con comillas triples (''' o """).

Ejemplo:

texto_multilinea = """

Esto es una cadena de texto

Con múltiples líneas.

Y me gusta escribir líneas.

"""

4. Qué es una interpolación de cadenas?

La interpolación de cadenas en Python es una técnica que permite insertar valores de variables dentro de una cadena de texto. Existen varias formas de hacerlo:

1. Usando el operador %

```
nombre = "Juan"
edad = 30
mensaje = "Hola, %s. Tienes %d años." % (nombre, edad)
print(mensaje)
```

2. Usando el método str.format()

```
nombre = "Juan"
edad = 30
mensaje = "Hola, {}. Tienes {} años.".format(nombre, edad)
print(mensaje)
```

3. Usando f-strings

```
nombre = "Juan"
edad = 30
mensaje = f"Hola, {nombre}. Tienes {edad} años."
print(mensaje)
```

5. Cuando deberíamos usar comentarios en Python?

Los comentarios son especialmente útiles en situaciones como:

- **Explicar Lógica Compleja:**
- **Notas para Colaboradores:** Cuando trabajas con un equipo, los comentarios agilizan la comprensión del propósito de cada fragmento de código.

- **Recordatorios para el Futuro:** Los comentarios pueden servir para recordar cambios pendientes en el código.
- **Documentación de Funciones:** para documentar las, especialmente si son funciones complejas.

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

Cada microservicio puede escalarse de manera independiente según sus necesidades. Las monolíticas son difíciles de escalar de manera independiente. Todo el sistema debe escalarse, incluso si solo una parte necesita más recursos.

Las aplicaciones de microservicio son más fáciles de mantener y actualizar, ya que los servicios son más pequeños y manejables. La aplicación monolítica a medida que crece, se vuelve más compleja y difícil de mantener.

El microservicio permite el uso de diferentes tecnologías y lenguajes para diferentes servicios. La aplicación monolítica es menos flexible para adoptar nuevas tecnologías o lenguajes

La aplicación de microservicio es más complejo de desarrollar y gestionar debido a la necesidad de coordinar múltiples servicios. La monolítica es más fácil de desarrollar y probar en las etapas iniciales.

El microservicio Requiere una infraestructura más avanzada para gestionar despliegues y comunicación entre servicios. La monolítica requiere un solo archivo o paquete para desplegar.

En la aplicación de microservicio puede haber una sobrecarga de comunicación entre servicios, lo que puede afectar el rendimiento. En la monolítica presenta menos sobrecarga de comunicación entre componentes