Checkpoint 3

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

- **bool**: tipo de dato que puede ser "**True"** o "**False**"
- Numéricos: hay varias opciones, los tipos más comunes son
    - **int**: números enteros ej. 10
    - **float**: números con decimales ej. 5.64
- **string** o texto: es una cadena de caracteres ej. "Hola mundo"
- **None:** la variable no tiene ningún valor. Por ejemplo, se puede utilizar en formularios, la variable será ***none*** hasta que se le asigne un valor.
- Valores binarios:
    - **byte**
    - **bytearray**
- **list:** es una lista de elementos que pueden ser diferentes tipos de datos ej. [1, 2, 3] o ["Uno", "Dos", "Tres"]
- **tuple:** lista inmutable de elementos, es decir, no se puede cambiar ej. (1, 2, 3) o ("Uno", "Dos", "Tres")
- **dict:** diccionario de valores, se definen tanto la clave como el valor ej. {"nombre": "Ainhoa", "edad": 27}

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

Se recomienda usar letras minúsculas y separar las palabras con guiones bajos ej. (post_count).

Es importante que los nombres de las variables sean descriptivos, es decir, que se entienda lo que representa la variable, así, la compresión de código es más fácil. Por ejemplo, si se quiere crear una lista de nombres la variable se podrías llamar ***names_ls***.

3. ¿Qué es un Heredoc en Python?

Los *Heredoc* son las frases multilínea. En Python hay diferentes maneras para crear un texto que tenga varias líneas:

Comillas triples (''' o """):

```
texto = '''Este es un ejemplo de una cadena de texto heredoc.
Podemos escribir todas las líneas que queramos.
Esto se puede realizar de distintas maneras.'''
```

```
texto = """Este es un ejemplo de una cadena de texto heredoc.
Podemos escribir todas las líneas que queramos.
Esto se puede realizar de distintas maneras."""
```

Con \n para saltos de línea:

```
texto = "Este es un ejemplo de una cadena de texto heredoc.\nPodemos escribir todas las líneas que queramos."
```

4. ¿Qué es una interpolación de cadenas?

La interpolación de cadenas permite introducir variables dentro de una cadena de texto. Permite generar cadenas dinámicas, lo que hace que el código sea más fácil de mantener. A continuación, se muestras dos maneras de hacer una interpolación.

Ejemplo 1:

```
nombre = 'Ainhoa'
edad = 27
texto = f'Hola, mi nombre es {nombre} y tengo {edad} años.'
```

Ejmplo 2:

```
nombre = 'Ainhoa'
edad = 27
texto = 'Hola, mi nombre es {0} y tengo {1} años.'.format(nombre, edad)
```

5. ¿Cuándo deberíamos usar comentarios en Python?

Los comentarios deben usarse solo para facilitar la compresión del código. Se debe explicar el "porque" del código y no el "que", ya que el "qué" o "cómo" pueden ir cambiando.

Los comentarios también pueden ser útiles para explicar alguna parte del código donde la lógica sea compleja o difícil de entender, por ejemplo, si hay cálculos o fórmulas que sean difíciles de entender.

Por otro lado, también se pueden poner comentarios sobre los puntos pendientes. Estos comentarios empiezan con "**TODO:**" y a continuación se explica lo que falta por hacer.

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

Monolíticas: su función esta basada en una única aplicación, es decir, toda la funcionalidad está contenida en un solo bloque, en un mismo servidor o sistema de archivos. La mayor ventaja de este tipo de aplicaciones es que es rápido de desarrollar (se puede crear la aplicación con funcionalidades básicas y luego ir avanzando). Sin embargo, es difícil de mantener ya que si se hace un cambio en una parte del sistema puede afectar a otras partes. Además, si se realiza un cambio, hay que compilar toda la aplicación, lo que puede ser ineficiente.

Microservicios: la arquitectura de estas aplicaciones se divide en diferentes servicios. Estos servicios son independientes y cada uno realiza una función específica, es decir, cada servicio es responsable de una parte del sistema. Las ventajas de estas aplicaciones son la flexibilidad, facilidad de escalar la aplicación (se puede mejorar solo un microservicio en vez de la aplicación completa) y la facilidad de mantenimiento. Por otro lado, entre las desventajas están la duplicación de lógica (algunas funcionalidades pueden estar duplicadas entre los diferentes servicios) y la complejidad de la gestión (los microservicios tienen que estar comunicados entre sí: esto requiere el uso de diferentes herramientas).