CHECKPOINT 3 - Preguntas teóricas

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

Los tipos de datos en Python son los siguientes:

- Booleans: Tienen un valor True o False. Esto, por ejemplo, nos permite marcar si una publicación de blog ha sido publicada o no. La primera letra siempre tiene que ir en mayúsculas (True/False)
- Numbers: Son los números enteros (1,2,3,…,), decimales, fracciones, flotantes, etc.
- Strings: Son las cadenas de caracteres. Pueden ser desde un nombre hasta un documento de HTML. Normalmente se escriben entre comillas.
- Bytes and byte arrays. son un tipo de datos reservado para el desarrollo complejo de Python.
- None: Estos datos son para definir “nulo” o “ninguno”. Son útiles en el caso de que queramos definir una variable pero aún no tengamos los datos de esta.
- List / Tuples / Sets / Dictionaries: Estos entrarían dentro de la categoría de estructura de datos. Nos permiten administrar colecciones de Python.

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

Para seguir buenas prácticas a la hora de escribir variables en Python, se recomienda seguir las PEP 8, que es la guía de estilo de Python. En esta guía se plantean algunas prácticas como escribir siempre en minúsculas y separando las palabras por un guion bajo además de utilizar nombres descriptivos y significativos para poder entender de forma fácil qué vamos a encontrar en la variable.

Ejemplo:
name_surname = ‘Kristine Hudgens’
(Se trata de un ejemplo muy sencillo, en el que vemos que en la variable está incluido el nombre y el apellido de la persona)

3 - ¿Qué es un Heredoc en Python?

Un Heredoc en Python se utiliza para definir cadenas largas de varias líneas o bloques de texto que pueden incluir caracteres especiales. Para utilizarlo es importante emplear siempre tres comillas al inicio y al final de la cadena. Estas pueden ser dobles (“””) o simples (‘”), pero es importante hacerlo de este modo, ya que por omisión Python interpreta que cuando termina una línea horizontal de programación tiene que estar el cierre de la variable. Utilizar tres comillas en estos casos nos permite que no nos devuelva un error.

Ejemplo:
texto_prueba = “””

Si escribimos un texto en Python que contenga varias líneas además de ‘caracteres especiales’, estos pueden interpretarse como parte del código de programación en vez de parte del texto que queremos que se imprima en pantalla
“””

4 - ¿Qué es una interpolación de cadenas?

La interpolación de cadenas es la capacidad de insertar valores de variables dentro de una cadena de texto. Es decir, en el caso de que tengamos un texto que queramos que se imprima siempre igual y que además incluya una variable, deberemos utilizar la interpolación de cadenas.

5 - ¿Cuándo deberíamos usar comentarios en Python?

Es importante emplear bien los comentarios en Python. Un error muy común es tratar de comentarlo todo. Por ejemplo comentar qué hace esa función, pero puede que estos comentarios se queden ahí después de que ese código se modifique. Para ello, en estos casos es mejor utilizar un nombre para la variable que explique bien su función.

Los casos en los que sí es interesante comentar el código es por ejemplo a la hora de organizar archivos grandes. Los comentarios nos permiten localizar de forma rápida donde se encuentra la parte del código que estamos buscando, como puede ser el encabezado. Otro caso interesante podría ser cuando tenemos algo pendiente por hacer o modificar en el código, es decir un TO DO.

Los comentarios se realizan de la siguiente manera:

Aquí va el comentario

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

Las aplicaciones monolíticas son más fáciles de desarrollar y permiten crear una aplicación con funciones básicas. Por ello, suelen ser más rápidas, ya que no tienen que comunicarse con tantos servicios. De todas formas, si queremos ampliarla o realizar modificaciones, es probable que de más fallos y causar problemas en diferentes partes del código.

Las aplicaciones de microservicios por su parte, dividen los procesos y nos permite crear varios servicios dentro de una misma aplicación. Estos podrán desarrollarse y ejecutarse de forma individual y no afectarán al resto de procesos. Por tanto, en caso de haber un fallo en algún punto de la aplicación, no tiene porqué fallar toda ella y eso la convierte en un método mucho más sencillo si queremos desarrollar una aplicación más compleja. Como desventaja, requieren de un mayor tiempo para desarrollarla.

Es decir, ambas son interesantes y tenemos que tener en cuenta cual es la aplicación final a la hora de decidirnos entre una de ellas.