PREGUNTAS TEÓRICAS

1. **¿Cuáles son los 4 tipos de datos en Python?**

Los 4 tipos de datos en Phyton a los que hace mención, son los trabajados en el ejercicio:

- Boolean: Sus valores son verdadero (True) o falso (False). Se debe tener cuidado y respetar la sintaxis de la primera letra mayúscula, ya que si no daría error.
- String: Se trata de una secuencia de caracteres, normalmente delimitados por comillas (dobles o simples). Puede ser desde una palabra, a una frase, o incluso un documento HTML completo.
- Number: números desde int (enteros), floats, fracciones, etc.
- List: similares a array, es una lista donde se guardan todos los ítems que queramos.

Sin embargo, en Python hay más tipos de datos:

- Bytes and byte arrays: es un arreglo inmutable de bytes.
- None o null: útil cuando queremos definir una variable, pero aún no queremos asignarle ningún valor.
- Sets,tuples: similares a las list pero con características únicas. Los Sets son una colección no ordenada de valores únicos y los tuples, colecciones de datos idénticos o distintos clasificados con un índice y que no pueden ser modificados
- Diccionaries: es una estructura de datos que permite almacenar cualquier tipo de información, desde cadenas de texto o caracteres hasta números enteros, con decimales, listas e incluso otros diccionarios.

2. **¿Qué tipo de convención de nomenclatura debemos usar para las variables en Python?**

La mejor práctica al nombrar las variables en Phyton es usar “snake case convention”, la cual consiste en escribir en minúscula separando las distintas palabras con guiones bajos _ . Por ejemplo: prueba_variable.

La “camel case convention”, es otra manera que suele ser usada en otros lenguajes ya que consiste en separar las distintas palabras por el uso de mayúsculas al inicio de éstas, por ejemplo: pruebaVariable. El problema resultante con esta última opción, es que en Phyton esta nomenclatura se usa para nombrar las clases entonces puede llevar a equívocos y a fallos y más cuando otras personas externas leen el código.

Lo más importante es usar siempre la misma nomenclatura en todo el código.

3. ¿Qué es un heredoc de Python?

Un heredoc en Python es un string de varias líneas, un multiple-line string. En estos casos la sintaxis cambia y hay que usar poniendo el texto a comentar entre tres comillas o dobles comillas para que no de error.

```
content =
"""
Prueba de multilínea
Prueba de multilínea
Prueba de multilínea
Prueba de multilínea
"""
```

Los heredoc también nos dan la opción de poner la función strip(), la cual elimina cualquier exceso de caracteres en la línea eliminando los saltos de línea y los espacios en el resultado.

4. ¿Qué es la interpolación de cadenas?

La interpolación de cadenas o string interpolation nos permite introducir código Python dentro de un string. Sirve para interpretar Phyton dentro de cadenas pudiendo intercalar dentro de un texto valores de una variable que previamente hemos definido y asignado, o incluso operaciones.

```
name = 'Vanesa'
greeting = f'Hello, {name}'
print(greeting)
```

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

Se deben usar únicamente para explicar instrucciones o describir algo.

Es importante remarcar que lo ideal es nombrar cada componente ya sean variables, clases, etc. de manera descriptiva para que no haya necesidad de poner comentarios y limpiar nuestro código simplificándolo e incluso quitando peso a nuestro proyecto.

También hay que tener en cuenta que hay que tener mucho cuidado con el uso de comentarios ya que pueden dar pie a muchos errores. Por ejemplo, si ponemos un comentario en un apartado del código explicando el funcionamiento de éste, y tras ello procedemos a realizar un cambio en este código, pero no en el comentario, este comentario no se adaptaría a la nueva utilidad, por lo cual, tendríamos un comentario que no se corresponde con el código, y esto podría ser muy confuso para otra persona que tuviese que interpretarlo.

Para facilitar la lectura del código y su comprensión, hay algunas convenciones standard. Por ejemplo #TODO, esto se usa en todos los lenguajes y es un símbolo universal para decir que aún no se ha hecho y hay que volver a dicho punto para trabajar en ello más adelante.

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

- Monolíticas: todo el sistema funciona como un todo, en una única aplicación. La ventaja es que es más rápido de desarrollar y más sencillo de realizar. La desventaja, en cambio es que el desarrollo de nuevas funciones puede ser más difícil porque todo está interconectado. Asimismo, debido a esta interconexión, el mantenimiento también se complica porque los errores también afectan a la totalidad del sistema, generando que un único error haga que deje de funcionar la app / programa.
- Microservicios: está formada por distintas aplicaciones dentro del sistema, cada una desarrollada de forma independiente, conectando sus servicios entre si mediante APIS. La desventaja sería que son más costosas de desarrollar en tiempo y recursos y más complicadas. La ventaja en cambio es que tiene mayor escalabilidad generando que sea más sencillo ir añadiendo módulos según la demanda de la necesidad concreta, otra ventaja es que hay menor peligro de que la app / programa deje de funcionar completamente ya que, al estar formada por bloques, es más fácil detectar los fallos y aislarlos.