¿Cuáles son los tipos de Datos en Python?

Booleans, numbers, strings, bytes, none, lists, tuples, sets, dictionaries.

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

Snake case, las letras en minúscula y usar _ como separador entre palabras.

¿Qué es un Heredoc en Python?

Es una forma de escribir cadenas de texto en múltiples líneas usando comillas o apóstrofes triples (‘’’/”””).

Python mantendrá el posicionamiento del texto con sus espacios y saltos de línea tal cual estén escritos dentro del heredoc.

```
ejemplo = ‘’’
Esto es un ejemplo de texto
usando heredoc
‘’’
```

Al usar la variable ejemplo, se mantendrán los saltos de línea del comienzo y del final.

Aparte de esto, se pueden utilizar los heredocs como comentarios multilínea en vez del #.

¿Qué es una interpolación de cadenas?

Es la capacidad de usar código dinámicamente dentro de cadenas de texto.

Por ejemplo, tenemos las variables:

```
nombre = “Ana”
usuario = “UserAna”
antiguedad = 23
```

Si queremos crear un saludo que responda independiente de los valores de las variables podemos hacerlo de esta forma:

```
saludo = f”Hola {nombre}, tu usuario es {usuario} y hace {antiguedad} semanas que eres miembro”
```

Existe una alternativa usando el método format:

```
saludo = ”Hola {0}, tu usuario es {1} y hace {2} semanas que eres miembro”.format(nombre, usuario, antigüedad)
```

¿Cuándo deberíamos usar comentarios en Python?

Para añadir documentación o contexto adicional al código.

Cuando no seamos capaz de crear una nomenclatura lo suficientemente descriptiva de lo que hace una variable o función.

Cuando queramos añadir un TODO o recordatorio de lo que hacer más adelante.

Para separar bloques enteros de código y hacerlo más legible.

Cuando se desee desactivar partes del código temporalmente.

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

Una aplicación monolítica contiene toda su funcionalidad en un sólo servicio mientras que las basadas en microservicios separan sus funcionalidades en, como su nombre sugiere, diferentes servicios de menor tamaño que interactúan entre sí.

Esto hace que desarrollar una aplicación monolítica sea más simple y más rápido y, al eliminar el tiempo de respuesta entre servicios, también puede potencialmente hacer que la aplicación sea más rápida.

Por contra, hace que sea muy difícil escalar la aplicación si por ejemplo el número de usuarios aumenta y también complica el desarrollo cuando existen múltiples personas trabajando en ella.

Con las aplicaciones de microservicios ocurre al revés, inicialmente es más complicado poner en marcha todo porque hay que preparar todas las diferentes partes, pero una vez solventado esto desaparecen las desventajas anteriores.

No obstante, si el proyecto es pequeño y/o no hay mucha gente desarrollándolo, una aplicación en microservicios podría no ser la opción más adecuada ya que se perdería tiempo con la puesta en marcha que podría ser utilizado en el propio desarrollo.