Checkpoint 3

¿Cuáles son los tipos de Datos en Python?

Los tipos son los siguientes:

- Booleans
- Numbers
- Strings
- Bytes and byte arrays
- None
- Lists
- Tuples
- Sets
- Dictionaries

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

Según la propia guía de estilo de Python (PEP8), los nombres de las variables deben estar en minúsculas, con las palabras separadas por guiones bajos según sea necesario para mejorar la legibilidad.

No recomienda utilizar las letras «l» (letra «ele» minúscula), «O» (letra «oh» mayúscula) ni «I» (letra «i» mayúscula) como nombres de variables de un solo carácter, para evitar confusiones.

¿Qué es un Heredoc en Python?

Heredoc o *Here Document* es la forma de representar un bloque de texto en código, que permite definir cadenas (strings) multilínea y que se entienda como un todo.

Ejemplo:

```
txt = """Así usamos las comillas.
De esta manera el texto multilínea se entenderá
correctamente, todas las líneas que haga falta.
“""

print(txt)
```

¿Qué es una interpolación de cadenas?

Es el proceso de insertar variables, expresiones o modificadores directamente en una cadena/string utilizando brackets {}, pudiendo así combinar hard code (la cadena) con la parte dinámica del código (lo que se encuentra entre los brackets).

Ejemplo del curso:

```
name = 'Kristine'
product = 'Python elearning course'

email_content = f"""
Hi {name}

Thank you for purchasing {product}

Regards,

Sales Team
"""

print(email_content)
```

¿Cuándo deberíamos usar comentarios en Python?

Se usa cuando aún no hemos terminado una parte del código o para organizarnos bien, para explicarlo, hacerlo más legible o evitar su ejecución cuando se está testeando, pero no es recomendable dejar comentarios una vez terminado, se considera mala práctica porque el código debería ser claro y entenderse sin necesidad de comentarios.

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

El sistema de las aplicaciones monoliticas es un modelo tradicional de programa de software, construido como una unidad unificada, autónoma e independiente de otras aplicaciones, lo contrario que las aplicaciones de microservicios, que se basan en una serie de servicios que se implementan de forma independiente.

Ambas tienen sus pros y sus contras; por ejemplo los monolitos pueden ser convenientes en las primeras etapas de un proyecto para facilitar la gestión del código y la implementación, porque permiten que todo el contenido del monolito se publique a la vez, aunque para realizar un cambio en una aplicación monolitica es necesario actualizarla accediendo a la base de código y compilando e implementando una versión actualizada de la interfaz del lado del servicio. Esto hace que las actualizaciones sean restrictivas y requieran mucho tiempo.

Por otro lado, es más fácil aislar y corregir fallos y errores en servicios individuales (en aplicaciones de microservicios), y permiten una implementación independiente rápida y sencilla de funciones individuales sin la amenaza de provocar la caída de toda la aplicación. También añaden mayor complejidad en comparación con una arquitectura monolítica, ya que existen más servicios en más lugares creados por múltiples equipos y puede existir una falta de estandarización, ya que sin una plataforma común, puede haber una diferencias de lenguajes, estándares de registro y monitorización.