CHECKPOINT 3

¿Cuáles son los tipos de Datos en Python?

Booleans: Para valores lógicos, que pueden ser True o False, usados habitualmente en condicionales, if True, while True, etc

Numbers: Para valores de tipo numérico, pueden ser, int (enteros), float (coma flotante), Decimal (para operaciones con decimales y mucha precisión)

String: Para cadenas de texto, han de ir entre comillas simples o dobles. Son de carácter inmutable, se puede iterar en ellas a través de su índice. Existen numerosas funciones en Python que permiten operar con ellas.

Bytes and bytesArrays: Para datos binarios, bytes es inmutable, mientras bytesArray se puede mutar.

None: Para indicar que no hay valor o que el valor es nulo, muy similar al null de otros lenguajes de programación. Muy útiles para declarar variables sin darles valor o saber si una variable tiene algún dato asignado.

Lists: Dentro de las secuencias de datos en Python, es una lista ordenada y mutable de elementos. Se declaran con corchetes: lista=['valor1','valor2','valor3']. Se puede acceder a cada valor o iterar sobre ellos a través de su índice: valor_nuevo=lista[0]

Tupples: Dentro de las secuencias de datos, una tupla es una lista ordenada e inmutable de valores. Se declaran con paréntesis: tupla=('valor1','valor2','valor3'). Se puede acceder a ellas u iterar a través de su índice. Muy utilizada para guardar valores que son constantes .

Sets: Dentro de las colecciones de datos en Python, es una colección desordenada y mutable de datos. No pueden tener valores repetidos. Se declaran mediante llaves. Mi_set={3,5,7}

Diccionarios: Dentro de las colecciones de datos en Python, es una colección desordenada y mutable de datos en los que se asocian pares de [valor : clave]. Los valores pueden repetirse pero las claves han de ser únicas para ese diccionario. Se accede a los valores a través de las claves: diccionario[clave] devuelve →[valor]

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

Deberíamos utilizar la convención PEP8, que especifica las principales prácticas de nomenclatura en Python. Por ejemplo, las variables se definen en minúsculas y sin espacios en blanco, utilizando en su lugar el '_', mi_variable, los nombres han de ser lo más descriptivos posibles. En los nombres de clases tampoco se utilizan espacios, pero las palabras empiezan con mayúsculas: MiClase

¿Qué es un Heredoc en Python?

Son cadenas de texto de multiples líneas. Se declaran con tres comillas.

mi_heredoc = """

texto línea 1

texto línea 2

texto línea 3

"""

Son útiles para guardar grandes cadenas de texto. Evitan tener que utilizar caracteres de formateo del estilo /n,/r..etc

Mejoran notablemente la legibilidad del texto.

Habitualmente utilizadas con la función "strip()" para eliminar los espacios en blanco

mi_heredoc.strip()

¿Qué es una interpolación de cadenas?

Es una técnica que permite insertar variables, valores y otro código Python dentro de una cadena de texto. P.ej:

nombre = 'Israel'

edad= 44

Aunque anteriormente se utilizaba el método [.format()] →

mi_interpolacion = 'Me llamo {}, y tengo {} años'.format(nombre,edad)

Actualmente se utiliza el método [f-strings]→

mi_interpolacion = f'Me lllamo{nombre} y tengo {edad} años'

¿Cuándo deberíamos usar comentarios en Python?

En principio los comentarios se deben limitar a los diferentes componentes de la aplicación, o como mucho a alguna aclaración excepcional cuando aun utilizando nombres (de variables, clase, métodos. etc) lo mas descriptivos posibles, no se entienda claramente el propósito que se desea conseguir.

Aunque es una recomendación, y por lo tanto no es obligatorio, aunque si conveniente, no estoy muy de acuerdo con ella, muchas veces pequeños comentarios hacen que no se pierda mucho tiempo intentando entender lo que hizo o quiso hacer otra persona…incluso lo que hiciste tu mismo¡¡

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

Son dos formas completamente diferentes de hacer lo mismo.

Las aplicaciones monolíticas tienen todos sus componentes en un único bloque de código, y se despliegan en un único servidor. Son mas fáciles y baratas de crear, pero en cambio su mantenimiento y escalabilidad es notablemente mas complejo, ya que cualquier cambio en su código, aunque sea un pequeño cambio en un componente, puede afectar a la apliacion entera. Se utilizan principalmente para proyectos sencillos o con pocos recursos.

Por otro lado las apliaciones de tipo microservicios, tienen sus componentes separados y desplegándose cada uno en su propio servidor. Son notablemente mas costosas ya que requieren de mas tiempo para codificar y mas recursos para su despliegue. A cambio tienen también sus ventajas: su mantenimiento es mas sencillo ya que se puede modificar el código de cada componente por separado sin afectar al resto y el fallo de uno de sus componentes no implica el fallo de la aplicación entera. Su escalabilidad es mayor y mas sencilla y permiten que sus componentes puedan funcionar con diferentes tecnologías. Generalmente se utilizan en grandes aplicaciones y son creados por equipos.