CHECKPOINT 4

1. ¿Cuál es la diferencia entre una lista y una tupla en Python?

¿Qué es?

Lista: Una lista en Python es una colección ordenada de elementos que es mutable, lo que significa que puedes cambiar sus elementos después de haberla creado. Las listas son *iterables* y por tanto se puede acceder a sus elementos mediante indexación,es decir, se puede acceder a cualquier elemento de una lista escribiendo el nombre de la lista y entre corchetes el número de orden en la lista.

Se tiene la posibilidad de agregar elementos a una lista mediante el método append y el método remove elimina un elemento de una lista.

Tupla: son secuencias de elementos similares a las listas, la diferencia principal es que las tuplas no pueden ser modificadas directamente, es decir, una tupla no dispone de los métodos como append o insert que modifican los elementos de una lista. Para acceder a un elemento de una tupla se utilizan los índices. Un índice es un número entero que indica la posición de un elemento en una tupla. El primer elemento de una tupla siempre comienza en el índice 0.

Las tuplas son objetos inmutables. No obstante, las tuplas pueden contener objetos u otros elementos de tipo secuencia, por ejemplo, una lista. Estos objetos, si son mutables, sí se pueden modificar.

¿Qué beneficio nos da?

Lista: Permite agregar, eliminar o modificar elementos, lo que la hace útil cuando necesitas cambiar datos dinámicamente.

Tupla: Debido a su inmutabilidad, las tuplas son más rápidas y seguras, ideales para datos constantes que no necesitan ser modificados.

Errores comunes:

Lista: Modificar la lista mientras se itera sobre ella puede causar errores inesperados.

Tupla: Intentar modificar una tupla (añadir o eliminar elementos) resultará en un error TypeError.

Sintaxis:

Lista: Para definir una lista se utilizan los corchetes, dentro de estos se colocan todos los elementos separados por comas. Ej: mi_lista = [1, 2, 3]

Tupla: Las tuplas se representan escribiendo los elementos entre paréntesis y separados por comas. Una tupla puede no contener ningún elemento, es decir, ser una tupla vacía. Una tupla puede incluir un único elemento, pero para que Python entienda que nos estamos refiriendo a una tupla es necesario escribir al menos una coma**.** mi_tupla = (1,)

¿Cuándo lo utilizo?

Lista: Cuando necesitas una colección de elementos que puedan cambiar durante la ejecución del programa.

Tupla: Cuando necesitas garantizar que los datos permanezcan constantes a lo largo de la ejecución del programa.

2. ¿Cuál es el orden de las operaciones?

¿Qué es?

El orden de las operaciones en Python sigue las reglas matemáticas conocidas como PEMDAS: Paréntesis, Exponentes, Multiplicación y División (de izquierda a derecha), Adición y Sustracción (de izquierda a derecha).

¿Qué beneficio nos da?

Nos ayuda a resolver expresiones complejas de manera consistente, evitando errores en cálculos matemáticos.

¿Cuándo lo utilizo?

Cada vez que realizas cálculos matemáticos en Python, para asegurarte de que las operaciones se ejecutan en el orden correcto. Ignorar el orden de las operaciones, lo que puede llevar a resultados inesperados si no se utilizan paréntesis cuando es necesario.

3. ¿Qué es un diccionario Python?

¿Qué es?

Son una forma de almacenar elementos como lo harías en una lista de Python. Pero, en lugar de acceder a los elementos utilizando su índice, le asignas una clave fija y accedes al elemento utilizando la clave. Los diccionarios Python son mutables, esto quiere decir que no tienen un tamaño predefinido y que su contenido aumenta o disminuye según las necesidades de la aplicación. Todos los datos son también modificables, es decir, se puede añadir, modificar, eliminar y consultar todos los datos de una manera sencilla y rápida.

¿Qué beneficio nos da?

Permite acceder a valores de forma rápida utilizando claves únicas, ideal para manejar datos estructurados y relacionados.

Errores comunes:

Intentar acceder a una clave que no existe en el diccionario genera un error KeyError.

Sintaxis:

Para definir un diccionario deberemos crear la variable utilizando un juego de llaves o mediante el método dict(), ambas opciones son correctas: mi_diccionario = {"nombre": "Juan", "edad": 25} o diccionario = dict()

¿Cuándo lo utilizo?

Los diccionarios en Python es una estructura de datos que puede convertirse en una compleja base de datos, y que incluso puede almacenarse de forma permanente en un sistema de ficheros. La cantidad de aplicaciones de todo tipo que se pueden desarrollar gracias a los diccionarios Python son infinitas.

Se pueden almacenar todos los datos necesarios de una persona, desde su nombre o teléfono hasta su dirección de correo electrónico. También es una estructura de datos adecuada para programar una aplicación de control de stock, que permita manejar de una forma sencilla el inventario de productos de un negocio.

4. ¿Cuál es la diferencia entre el método sort() y la función sorted()?

¿Qué es?

sort(): Es un método que ordena los elementos de una lista en su lugar, modificando la lista original. Sort ordenará de menor a mayor los valores y para los Strings los ordenará de manera alfabética, basta con añadirlo como una extensión a la variable con los paréntesis vacíos.

sorted(): Es una función que toma una lista y devuelve una nueva lista con esos elementos en orden ordenado . La lista original no se modifica.

¿Qué beneficio nos da?

sort(): Es eficiente cuando no necesitas mantener la lista original y quieres ahorrar memoria.

sorted(): Es útil cuando necesitas preservar la lista original sin modificarla.

Errores comunes:

Confundir sort() y sorted() puede llevar a modificar accidentalmente la lista original cuando no era deseado.

Sintaxis:

sort(): mi_lista.sort()

sorted(): lista_ordenada = sorted(mi_lista)

¿Cuándo lo utilizo?

sort(): Cuando quieres ordenar una lista y no necesitas la lista original.

sorted(): Cuando necesitas una nueva lista ordenada sin alterar la original.

5. ¿Qué es un operador de asignación?

Son atajos para escribir otros operadores de manera más corta, y asignar su resultado a la variable inicial. Por ejemplo, el operador += en x+=1 es equivalente a x=x+1. Nos **permiten realizar una operación y almacenar su resultado en la variable inicial**.

Se utiliza un operador de asignación para asignar valores a una variable. Esto generalmente se combina con otros operadores (como aritmética, bit a bit) donde la operación se realiza en los operandos y el resultado se asigna al operando izquierdo.

¿Qué beneficio nos da?

Simplifica la escritura de código cuando necesitas actualizar el valor de una variable en función de su valor actual.

¿Cuándo lo utilizo?

Cuando necesitas actualizar el valor de una variable de manera acumulativa, como en ciclos o acumulación de valores.

Por ejemplo el operador += se utiliza para sumar el valor de la derecha a la variable y asignar el resultado a la variable.Ej: El operador -= se utiliza para restar el valor de la derecha a la variable y asignar el resultado a la variable.

Ej: total = 100

descuento = 20

total -= descuento # Equivalente a total = total - descuento

Después de esta operación, total será igual a 80.