M2C4 Python Assignment

Preguntas teóricas :

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

La principal diferencia entre una lista y una tupla en Python es que las listas son **mutables** , lo que significa que puedes modificar sus elementos después de haberla creado (puedes agregar , eliminar o cambiar elementos).

En cambio, las tuplas son **inmutables,** lo que significa que una vez que se crea una tupla, no puedes modificar sus elementos (no puedes agregar, eliminar ni cambiar los valores).

1. **Lista** : Mutable (Puedes cambiar sus elementos).
    a. **Ser mutable**, lo que significa que sus elementos pueden cambiarse después de ser creados. Puedes agregar , eliminar o modificar elementos.
    b. **Utilizar corchetes [ ]** para su creación.
    c. **Permitir elementos duplicados** , ya que cada elemento tiene una posición específica dentro de la lista.
    d. **Tener acceso por índice** , lo que significa que puedes acceder a cualquier elemento usando su posición.
    e. **Rendimiento ,** más lenta por que permite modificaciones.
    f. **Seguridad,** menos segura, porque sus valores pueden cambiar.

    g. **Ejemplo de lista :** mi_lista = [1, 2, 3, “Python”, True]

2. **Tupla** : Inmutable (No puedes cambiar sus elementos).
    a. **Ser inmutable,** lo que significa que no puedes agregar, eliminar ni modificar elementos.
    b. **Utilizar paréntesis ( )** para su creación.
    c. **Tener acceso por índice,** igual que las listas.
    d. **El rendimiento,** es más rápida , ya que es fija.
    e. **Seguridad ,** es mas segura ya que sus valores no cambian.

    f. **Ejemplo de una tupla :** mi_tupla = (1 , 2 , 3, “python” , True)

M2C4 Python Assignment

2- Cuál es el orden de las operaciones ?

El orden de las operaciones en matemáticas y programación (en Python también) se refiere al conjunto de reglas que determinan el orden en el que se deben realizar las operaciones dentro de una expresión.

1. Paréntesis ( ) : Las operaciones dentro de paréntesis se realizan primero.

2. Exponente ** : La potencia o exponenciación se realiza después de los paréntesis .

3. Multiplicación, División, Módulo y División entera * | % | / : Estas operaciones se realizan de izquierda a derecha.

4. Suma y Resta + | - : Se realizan después de las multiplicaciones y divisiones , también de izquierda a derecha.

5. Operadores de comparación (por ejemplo, == , != , < , > , <=, >=): Estas operaciones se realizan después de las sumas y restas.

6. Operadores lógicos : and, or, not se evalúan después de las comparaciones, en el siguiente orden :
    a. Primero not.
    b. Luego and.
    c. Finalmente or.
7. **Ejemplo**
    a. resultado = 5 + 3 * 2 ** 2 / 4 -1
        i. Exponente 2** 2 = 4
        ii. Multiplicación: 3 * 4 = 12
        iii. Division : 12 / 4 = 3
        iv. Suma : 5 + 3 = 8
        v. Resta : 8 - 1 = 7

        vi. Salida : 7.0

Esto ayuda a garantizar que las expresiones se resuelvan de manera consistente en todos los lenguajes de programación.

M2C4 Python Assignment

3 - ¿Qué es un diccionario Python ?

Un diccionario en Python es una estructura de datos que almacena información en pares **clave - valor** . Cada elemento del diccionario tiene una clave única . que se utiliza para acceder al valor correspondiente. Es una estructura que nos permite asociar datos de manera fácil y rápida .

Es como una agenda telefónica , donde cada nombre (clave) tiene asociado un número de teléfono (valor).

Características clave de los diccionarios en Python :

1. **Desordenados:** No tienen un orden específico.
2. **Mutables:** Puedes agregar, modificar y eliminar elementos.
3. **Acceso rápido :** La búsqueda de un valor a través de su clave es muy eficiente.

4. **Ejemplo de Diccionario :**
    a. mi_diccionario = {
    'nombre' : "Kevin",
    'edad' : "28",
    'Profesión' : 'Electricista'
    }

    print (mi_diccionario['nombre']) # Salida : Kevin , Aquí podemos seleccionar qué dato queremos que nos imprima de nuestro diccionario.

4- Cuál es la diferencia entre el método ordenado y la función de ordenación ?

La diferencia entre el método sort( ) y la función sorted( ) en Python es importante y está relacionada con como trabaja con las listas y su comportamiento.

1. sort ( ) :
    a. Es un método de las listas en Python
    b. Modifica la lista original en su lugar (es decir, la lista se ordena directamente y no se crea una nueva lista).
    c. No devuelve ningún valor (devuelve None), solo ordena la lista en su lugar.
    d. Solo se puede usar con listas, no con otros tipos de colecciones como tuplas y diccionarios.
        i. **Ejemplo de sort ( ) :**
            1. **numbers = [3, 1, 4, 1, 5, 9]**
            numbers.sort()
            print(numbers) # Salida: [1, 1, 3, 4, 5, 9]

M2C4 Python Assignment

2. sorted ( ) :
    a. Es una función que puede usarse con cualquier iterable (listas , tuplas , diccionarios, etc..)
    b. Devuelve una nueva lista ordenada , sin modificar el iterable original.
    c. Puede ser usada con cualquier tipo de secuencia (no solo listas), lo que le da mayor flexibilidad.

        i. **Ejemplo de sorted ( ) :**
            1. **numbers = [3, 1, 4, 1, 5, 9]**
               nueva_lista = sorted(numbers)
               print(nueva_lista) # Salida: [1, 1, 3, 4, 5, 9]
               print(numbers) # La lista original no cambia

3. Resumen de las diferencias :
    a. sort ( ) :
        i. Modificar la lista original.
        ii. Solo se puede usar con listas.
        iii. No devuelve nada (devuelve None).
    b. sorted ( ) :
        i. Devuelve una nueva lista ordenada.
        ii. Puede usarse con cualquier iterable (listas , tuplas , diccionarios, etc.).
        iii. No modifica el iterable original.

M2C4 Python Assignment

5- Que es un operador de reasignación ?

Es un tipo de operador que se utiliza para modificar el valor de una variable.

Los operadores de resignación combinan una operación aritmética con la reasignación del valor a la misma variable, permitiendo realizar cálculos y actualizar el valor de una variable de manera compacta y eficiente.

Ejemplos de operadores de reasignación en Python :

1. **Suma y reasignación (+=) :** Se usa para sumar el valor actual de la variable con otro valor y reasignar el resultado a la misma variable.

2. **Resta y reasignación (-=):** Se usa para restar el valor actual de la variable con otro valor y reasignar el resultado.

3. **Multiplicación y reasignación (*=):** Se usa para multiplicar el valor actual de la variable por otro valor y reasignar el resultado.

4. **División y reasignación (/=):** Se usa para dividir el valor actual de la variable por otro valor y reasignar el resultado.
5. **División entera y reasignación (//=):** Se usa para realizar una división entera y reasignar el resultado.

6. **Módulo y reasignación (%=):** Se usa para obtener el residuo de la división de la variable por otro valor y reasignar el resultado.

7. **Exponenciación y reasignación (**=):** Se usa para elevar el valor actual de la variable a una potencia y reasignar el resultado.