M2C4-Python-Assignment

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

Ambos son estructuras de datos que contienen una lista de elementos ordenados.

Las listas se representan con su contenido entre corchetes y las tuplas se representan con su contenido entre paréntesis. En ambas los elementos se separan mediante comas.

Su principal diferencia es que las listas son mutables, se pueden modificar (añadir, reemplazar o eliminar) sus elementos después de crearlas, mientras que las tuplas son inmutables y sus elementos no se pueden modificar una vez creadas.

Ejemplos de tuplas:

```
tupla_integrales = (1, 2, 3)
tupla_texto = ('a', 'b', 'c')
tupla_booleana = (True, False)
```

No se puede editar ni el orden ni el contenido

Ejemplos de listas:

```
lista_integrales = [1, 2, 3, 4, 5]
lista_texto = ['a', 'b', 'c', 'd', 'e']
lista_booleana = [True, False, True]
```

El contenido y su orden es editable

2-¿Cuál es el orden de las operaciones?

El orden de las operaciones en Python es igual al orden jerárquico de las operaciones aritméticas.

Primero se realizarian las operaciones dentro de los paréntesis, luego las operaciones de exponentes, las de multiplicación y división y por último las de suma y resta.

Ejemplo:

```
resultado_operacion = (1 + 2) ** 2 / 3 - (4 * 5)
print(resultado_operacion) #-17.0
```

1. *(3) ** 2 / 3 - (20)*
2. *9/3-20*
3. *3.0-20*
4. *-17.0*

3-¿Qué es un diccionario Python?

Un diccionario es una estructura de datos que almacena datos en parejas compuestas por una clave y su valor.

- Las claves deben ser objetos inmutables, mientras que los valores pueden ser datos mutables.
- Los diccionarios no tienen un orden definido.
- Son mutables, se pueden modificar elementos de un diccionario después de crearlo añadiendo, reemplazando o eliminando elementos.
- Las claves deben ser únicas, no se pueden repetir.

Sintaxis: nombre_diccionario = {clave: valor, clave: valor, clave: valor}

Ejemplo:

```
mi_diccionario = {
    "valor1": 21,
    "valor2": 42
}
primer_numero = mi_diccionario["valor1"] #los valores están
indexados por sus claves
print(primer_numero) #21
```

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

El método sorted() nos devuelve ordenados los elementos de una lista pero sin modificar el orden original de la lista.

```
mi_lista = [5, 1, 3, 2, 4]
mi_lista_ordenada = sorted(mi_lista)
print(mi_lista) #[5, 1, 3, 2, 4]
print(mi_lista_ordenada) #[1, 2, 3, 4, 5]
```

La lista "mi_lista" no cambia

El método sort() reorganizaría los elementos de la lista pero cambiando el orden original de la lista.

```
mi_lista = [5, 1, 3, 2, 4]
mi_lista.sort()
print(mi_lista) #[1, 2, 3, 4, 5]
```

La lista cambia

5-¿Qué es un operador de reasignación?

Los operadores de asignación o reasignación son un símbolo o combinación de símbolos que se utilizan para asignar un valor a una variable. El operador de asignación más usado es el signo igual (=), que se utiliza para asignar un valor inicial a una variable.

```
mi_variable = 21 #21 sería el valor de la variable mi_variable
mi_variable = 12 #reasignación, ahora mi_variable tendría el valor
de 12
```

Con los operadores de asignación compuestos (combinando operadores con el símbolo igual) podemos realizar operaciones aritméticas y de asignación en una sola expresión, simplificando así el código y haciéndolo más legible y eficaz.

Con el operador compuesto “+=” por ejemplo, se suma el valor de la derecha al valor actual de la variable y reasigna su valor con el resultado de la operación.

```
mi_variable = 12
mi_variable += 9 #Sería igual a: mi_variable = mi_variable + 9
print(mi_variable) #ahora mi variable es 21
```