**FUNCIONES Y METODOS EN LISTAS**

**FUNCIONES**

len(lista): Devuelve el número de elementos en la lista.

sum(lista): Retorna la suma de todos los elementos numéricos de la lista.

max(lista): Devuelve el elemento más grande de la lista.

min(lista): Devuelve el elemento más pequeño de la lista.

sorted(lista): Retorna una nueva lista ordenada sin modificar la original.

list(iterable): Convierte un iterable en una lista

enumerate()– Permite recorrer la lista obteniendo índices y valores

**METODOS**

lista.append(elemento): Agrega un elemento al final de la lista.

lista.extend(otra\_lista): Agrega los elementos de otra lista al final.

lista.insert(posición, elemento): Inserta un elemento en una posición específica.

lista.remove(elemento): Elimina la primera aparición del elemento en la lista.

lista.pop([índice]): Elimina y devuelve el elemento en la posición indicada (por defecto, el último).

lista.clear(): Elimina todos los elementos de la lista.

lista.index(elemento): Devuelve el índice de la primera aparición del elemento.

lista.count(elemento): Devuelve cuántas veces aparece un elemento en la lista.

lista.sort([reverse=False]): Ordena la lista en su lugar.

lista.reverse(): Invierte el orden de los elementos en la lista.

lista.copy(): Devuelve una copia de la lista

**FUNCIONES Y MÉTODOS EN CONJUNTOS**

***Añadir y eliminar elementos***

conjunto.add(elemento): Agrega un elemento al conjunto.

conjunto.remove(elemento): Elimina un elemento, pero genera error si no existe.

conjunto.discard(elemento): Elimina un elemento sin generar error si no existe.

conjunto.pop(): Elimina y devuelve un elemento aleatorio del conjunto.

conjunto.clear():Vacía el conjunto.

print(conjunto)

***Operaciones entre conjuntos***

set.union(otro\_set): Retorna un nuevo conjunto con la unión de ambos.

set.intersection(otro\_set): Retorna la intersección de ambos conjuntos.

set.difference(otro\_set): Retorna la diferencia (elementos presentes en el primero pero no en el segundo).

set.symmetric\_difference(otro\_set): Retorna los elementos que están en uno u otro conjunto, pero no en ambos.

print(set1.union(set2))

print(set1.intersection(set2))

print(set1.difference(set2))

print(set1.symmetric\_difference(set2))

Tuplas ( tuple) en Python

Las tuplas son estructuras ordenadas e inmutables , lo que significa que no se pueden modificar después de su creación. Son útiles cuando necesitas garantizar que los datos no cambien.

Funciones y métodos en Tuplas

🔹 Acceso a elementos

Podemos acceder a los elementos de una tupla mediante índices:

pitón

Copiar

Editar

tupla = (10, 20, 30, 40)

print(tupla[1]) # Salida: 20

print(tupla[-1]) # Salida: 40 (índice negativo cuenta desde el final)

🔹 Conteo y búsqueda de elementos

count(elemento): Cuenta cuantas veces aparece un elemento.

index(elemento): Devuelve el índice de la primera aparición del elemento.

Ejemplo

pitón

Copiar

Editar

tupla = (5, 10, 15, 10, 20)

print(tupla.count(10)) # Salida: 2

print(tupla.index(15)) # Salida: 2

🔹 Conversión entre listas y tuplas

A veces necesitamos convertir entre listas y tuplas para modificar los datos.

Ejemplo

pitón

Copiar

Editar

lista = [1, 2, 3, 4]

tupla = tuple(lista) # Convierte una lista en tupla

print(tupla) # Salida: (1, 2, 3, 4)

nueva\_lista = list(tupla) # Convierte una tupla en lista

nueva\_lista.append(5) # Ahora podemos modificar la lista

print(nueva\_lista) # Salida: [1, 2, 3, 4, 5]

🔹 Desempaquetado de Tuplas

Podemos asignar los valores de una tupla a múltiples variables.

Ejemplo

pitón

Copiar

Editar

coordenadas = (10, 20)

x, y = coordenadas # Asigna 10 a x y 20 a y

print(x, y) # Salida: 10 20

También podemos usar \*para capturar múltiples valores:

pitón

Copiar

Editar

numeros = (1, 2, 3, 4, 5)

a, \*b, c = numeros

print(a) # Salida: 1

print(b) # Salida: [2, 3, 4]

print(c) # Salida: 5

Funciones y Métodos para Trabajar con Diccionarios en Python (dict)

Los diccionarios en Python son estructuras de datos que almacenan pares clave-valor. Son mutables y permiten acceso rápido a los valores a través de sus claves.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

🔹 Funciones Globales para Diccionarios

Estas funciones operan sobre diccionarios sin ser métodos específicos.

1️⃣ len(diccionario) Devuelve el número de elementos en el diccionario.

• 🔹 Ejemplo:

python

CopiarEditar

dic = {'a': 10, 'b': 20, 'c': 30}

print(len(dic)) # Salida: 3

2️⃣ dict(iterable)

• 📌 Crea un diccionario a partir de una lista de tuplas clave-valor.

• 🔹 Ejemplo:

python

CopiarEditar

lista = [('nombre', 'Ana'), ('edad', 25)]

dic = dict(lista)

print(dic) # Salida: {'nombre': 'Ana', 'edad': 25}

3️⃣ max(diccionario) y min(diccionario) Devuelven la clave máxima y mínima del diccionario (según orden alfabético o numérico de las claves).

• 🔹 Ejemplo:

python

CopiarEditar

dic = {'a': 10, 'c': 30, 'b': 20}

print(max(dic)) # Salida: 'c'

print(min(dic)) # Salida: 'a'

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

🔹 Métodos de Diccionarios

Los métodos de diccionarios permiten manipular y obtener información de ellos.

1️⃣ .keys() Devuelve una vista de todas las claves.

• 🔹 Ejemplo:

python

CopiarEditar

dic = {'a': 10, 'b': 20, 'c': 30}

print(dic.keys()) # Salida: dict\_keys(['a', 'b', 'c'])

2️⃣ .values() Devuelve una vista de todos los valores.

• 🔹 Ejemplo:

python

CopiarEditar

print(dic.values()) # Salida: dict\_values([10, 20, 30])

3️⃣ .items() Devuelve una vista de los pares clave-valor en tuplas.

• 🔹 Ejemplo:

python

CopiarEditar

print(dic.items()) # Salida: dict\_items([('a', 10), ('b', 20), ('c', 30)])

4️⃣ .get(clave, valor\_por\_defecto) Obtiene un valor dado su clave. Si no existe, devuelve el valor por defecto.

• 🔹 Ejemplo:

python

CopiarEditar

print(dic.get('b')) # Salida: 20

print(dic.get('d', 'No existe')) # Salida: 'No existe'

5️⃣ .update(otro\_diccionario) Fusiona otro diccionario dentro del actual, sobrescribiendo claves repetidas.

• 🔹 Ejemplo:

python

CopiarEditar

dic.update({'b': 50, 'd': 40})

print(dic) # Salida: {'a': 10, 'b': 50, 'c': 30, 'd': 40}

6️⃣ .pop(clave, valor\_por\_defecto) Elimina y devuelve el valor asociado a una clave.

• 🔹 Ejemplo:

python

CopiarEditar

valor = dic.pop('b')

print(valor) # Salida: 50

print(dic) # Salida: {'a': 10, 'c': 30, 'd': 40}

7️⃣ .popitem()

• 📌 Elimina y devuelve un par clave-valor al azar (en Python 3.7+ elimina el último).

• 🔹 Ejemplo:

python

CopiarEditar

par = dic.popitem()

print(par) # Salida: ('d', 40)

print(dic) # Salida: {'a': 10, 'c': 30}

8️⃣ .setdefault(clave, valor\_por\_defecto) Si la clave existe, devuelve su valor. Si no, la agrega con un valor por defecto.

• 🔹 Ejemplo:

python

CopiarEditar

print(dic.setdefault('c', 100)) # Salida: 30 (ya existía)

print(dic.setdefault('e', 50)) # Salida: 50 (clave nueva añadida)

print(dic) # Salida: {'a': 10, 'c': 30, 'e': 50}

9️⃣ .clear()

• 📌 Elimina todos los elementos del diccionario.

• 🔹 Ejemplo:

python

CopiarEditar

dic.clear()

print(dic) # Salida: {}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_