Tuplas y Diccionarios en Python

Tuplas en Python

¿Qué es una tupla?

Una **tupla** es una colección **ordenada** e **inmutable** de elementos. Esto significa que, una vez que se crea una tupla, no se puede modificar. Se definen usando paréntesis ().

Características principales de una tupla:

- **Inmutables**: No se pueden modificar después de su creación.
- **Ordenadas**: Los elementos tienen un orden específico que no cambia.
- **Permiten duplicados**: Los elementos pueden repetirse.

Ejemplos de creación de tuplas:

```python
# Crear una tupla con varios elementos
mi_tupla = (10, 20, 30)

# Crear una tupla con diferentes tipos de datos
tupla_varios_tipos = ("texto", 100, 4.5, True)

# Crear una tupla vacía
tupla_vacia = ()

# Crear una tupla de un solo elemento (importante incluir la coma)
tupla_un_elemento = (50,)
```

Acceso a los elementos de una tupla:

Puedes acceder a los elementos de una tupla usando índices (como en las listas).

```python
mi_tupla = ("manzana", "banana", "cereza")

# Acceder al primer elemento
print(mi_tupla[0])  # Salida: manzana

# Acceder al último elemento
print(mi_tupla[-1])  # Salida: cereza
```

Operaciones comunes con tuplas:

1. **Desempaquetar tuplas**: Puedes asignar los elementos de una tupla a variables.

```
mi_tupla = ("rojo", "verde", "azul")
r, v, a = mi_tupla

print(r)  # Salida: rojo
print(v)  # Salida: verde
print(a)  # Salida: azul
```

2. **Concatenar tuplas**: Como las tuplas son inmutables, para "modificarlas" puedes crear una nueva tupla concatenando otras.

```
tupla1 = (1, 2, 3)
tupla2 = (4, 5, 6)

tupla_concatenada = tupla1 + tupla2
print(tupla_concatenada)  # Salida: (1, 2, 3, 4, 5, 6)
```

3. **Verificar la existencia de un elemento en una tupla**:

```
mi_tupla = ("manzana", "banana", "cereza")
print("banana" in mi_tupla)  # Salida: True
```

4. **Tamaño de una tupla**:

```
mi_tupla = (1, 2, 3, 4, 5)
print(len(mi_tupla))  # Salida: 5
```

Convertir una lista en una Tupla

Puedes convertir una lista en una tupla utilizando la función integrada tuple(). A continuación te muestro cómo hacerlo:

```
# Crear una lista
mi_lista = [1, 2, 3, 4, 5]

# Convertir la lista en una tupla
mi_tupla = tuple(mi_lista)

print(mi_tupla)  # Salida: (1, 2, 3, 4, 5)
```

Explicación:

- La función tuple() toma un iterable (como una lista) y lo convierte en una tupla.
- El contenido de la lista permanece igual, pero la estructura cambia a una tupla, que es inmutable.

Ejemplo con una lista de diferentes tipos de datos:

```python
mi_lista = ["hola", 42, 3.14, True]

# Convertir la lista en una tupla
mi_tupla = tuple(mi_lista)

print(mi_tupla)  # Salida: ('hola', 42, 3.14, True)
```

Este método es útil si necesitas la inmutabilidad de los datos en lugar de la mutabilidad de las listas.

Diccionarios en Python

¿Qué es un diccionario?

Un **diccionario** es una colección **desordenada**, **mutable** e **indexada** de pares **clave-valor**. En lugar de acceder a los elementos por un índice numérico, se accede a ellos mediante claves.

Características principales de un diccionario:

- **Mutables**: Se pueden modificar después de su creación.
- **No permiten claves duplicadas**.
- **Claves únicas**: Cada clave debe ser única, aunque los valores pueden repetirse.

Ejemplos de creación de diccionarios:

```python
# Crear un diccionario vacío
mi_diccionario = {}

# Crear un diccionario con algunos pares clave-valor
diccionario_colores = {"rojo": "#FF0000", "verde": "#00FF00", "azul":
"#0000FF"}

# Diccionario con diferentes tipos de valores
persona = {"nombre": "Ana", "edad": 25, "es_estudiante": True}
```

Acceso a los valores de un diccionario:

Puedes acceder a los valores de un diccionario usando sus claves.

```python
persona = {"nombre": "Ana", "edad": 25, "es_estudiante": True}

# Acceder al valor de la clave "nombre"
print(persona["nombre"])  # Salida: Ana

# Acceder al valor de la clave "edad"
print(persona["edad"])  # Salida: 25
```

Operaciones comunes con diccionarios:

1. **Agregar y modificar elementos**:

```
persona = {"nombre": "Ana", "edad": 25}

# Agregar una nueva clave-valor
persona["ciudad"] = "Madrid"

# Modificar un valor existente
persona["edad"] = 26

print(persona)
# Salida: {'nombre': 'Ana', 'edad': 26, 'ciudad': 'Madrid'}
```

2. **Eliminar un elemento**:

```
persona = {"nombre": "Ana", "edad": 25, "ciudad": "Madrid"}

# Eliminar una clave-valor usando pop()
persona.pop("ciudad")

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

3. **Recorrer un diccionario**:

```
diccionario_colores = {"rojo": "#FF0000", "verde": "#00FF00", "azul":
"#0000FF"}

# Recorrer claves y valores
for clave, valor in diccionario_colores.items():
    print(f"La clave es {clave} y el valor es {valor}")

# Salida:
# La clave es rojo y el valor es #FF0000
# La clave es verde y el valor es #00FF00
# La clave es azul y el valor es #0000FF
```

4. **Verificar si una clave existe**:

```
persona = {"nombre": "Ana", "edad": 25}

if "nombre" in persona:
    print("La clave 'nombre' existe en el diccionario.")
```

5. **Obtener el valor de una clave sin causar un error si no existe**:

```
persona = {"nombre": "Ana", "edad": 25}

# Usar get() para evitar un KeyError si la clave no existe
ciudad = persona.get("ciudad", "No especificada")
print(ciudad)  # Salida: No especificada
```

6. **Diccionarios anidados** (diccionarios dentro de otros diccionarios):

```
alumnos = {
    "alumno1": {"nombre": "Pedro", "edad": 20},
    "alumno2": {"nombre": "Laura", "edad": 22},
}

# Acceder a los datos del alumno2
print(alumnos["alumno2"]["nombre"])  # Salida: Laura
```

Ejemplos prácticos combinando tuplas y diccionarios

a- **Crear un diccionario con tuplas como claves**: A veces es útil usar tuplas como claves en un diccionario, ya que las tuplas son inmutables y pueden ser utilizadas como claves, mientras que las listas no pueden.

```
# Diccionario con tuplas como claves
puntos = {(0, 0): "origen", (1, 2): "punto A", (3, 5): "punto B"}

print(puntos[(1, 2)])  # Salida: punto A
```

b- **Contar la frecuencia de elementos en una lista usando un diccionario**:

```
# Lista de palabras
palabras = ["manzana", "banana", "manzana", "cereza", "banana",
"cereza", "cereza"]

# Contar la frecuencia de cada palabra
frecuencia = {}

for palabra in palabras:
    if palabra in frecuencia:
        frecuencia[palabra] += 1
    else:
        frecuencia[palabra] = 1

print(frecuencia)
# Salida: {'manzana': 2, 'banana': 2, 'cereza': 3}
```

Conclusión

- Las **tuplas** son ideales para almacenar colecciones de datos que no deben cambiar.
- Los **diccionarios** son útiles para almacenar pares clave-valor donde las claves deben ser únicas y puedes modificar los valores según sea necesario.
- Ambas estructuras son esenciales en Python y se pueden usar de manera eficiente en diversos escenarios.

Trabajo Práctico: Tuplas y Diccionarios en Python

Ejercicios con Tuplas

Ejercicio 1: Desempaquetado de tuplas

Dada la siguiente tupla:

```
punto = (10, 20)
```

a) Desempaqueta los valores de la tupla en las variables x y y.
b) Imprime el valor de cada variable.

Ejercicio 2: Concatenación de tuplas

Tienes las siguientes dos tuplas:

```
tupla1 = (1, 2, 3)
tupla2 = (4, 5, 6)
```

a) Crea una nueva tupla que sea la concatenación de tupla1 y tupla2.
b) Imprime la nueva tupla.

Ejercicio 3: Índices y slicing en tuplas

Dada la siguiente tupla:

```
frutas = ("manzana", "banana", "cereza", "naranja", "kiwi")
```

a) Imprime el segundo elemento de la tupla.
b) Imprime los tres primeros elementos de la tupla.
c) Imprime el último elemento de la tupla usando un índice negativo.

Ejercicio 4: Verificar existencia de un elemento

Dada la tupla:

```
colores = ("rojo", "verde", "azul", "amarillo")
```

a) Verifica si el color "morado" está presente en la tupla.
b) Verifica si el color "azul" está presente en la tupla.
c) Imprime el resultado de ambas verificaciones.

Ejercicio 5: Contar elementos en una tupla

Dada la siguiente tupla:

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

a) Cuenta cuántas veces aparece el número 4 en la tupla.
b) Imprime el resultado.

Ejercicios con Diccionarios

Ejercicio 6: Crear y acceder a un diccionario

Crea un diccionario que contenga información sobre una persona, incluyendo las claves: nombre, edad, y ciudad. Luego:

a) Imprime el valor asociado a la clave nombre.
b) Imprime el valor asociado a la clave edad.

Ejercicio 7: Modificar un diccionario

Dado el siguiente diccionario:

```
persona = {"nombre": "Juan", "edad": 30, "ciudad": "Madrid"}
```

a) Cambia el valor de la clave edad a 31.
b) Agrega una nueva clave profesión con el valor "Ingeniero".
c) Imprime el diccionario modificado.

Ejercicio 8: Eliminar elementos de un diccionario

Dado el diccionario:

```
mascota = {"nombre": "Firulais", "especie": "Perro", "edad": 5}
```

a) Elimina la clave edad del diccionario.
b) Imprime el diccionario después de eliminar esa clave.

Ejercicio 9: Contar palabras en un texto

Dado el siguiente texto:

```
texto = "manzana naranja manzana pera pera pera naranja manzana"
```

a) Escribe un programa que cuente cuántas veces aparece cada palabra en el texto utilizando un diccionario.
b) Imprime el diccionario resultante.

Ejercicio 10: Diccionario anidado

Dado el siguiente diccionario de alumnos:

```
alumnos = {
    "alumno1": {"nombre": "Carlos", "edad": 20, "nota": 8.5},
    "alumno2": {"nombre": "Lucía", "edad": 22, "nota": 9.0},
    "alumno3": {"nombre": "Ana", "edad": 21, "nota": 7.5},
}
```

a) Imprime el nombre del alumno2.
b) Imprime la nota del alumno3.

Ejercicio Final

Sistema de Gestión de Inventario de Tienda

Descripción:

Una tienda de tecnología necesita llevar un registro de su inventario. Cada producto tiene asociado un código, una descripción, y un precio. Además, el sistema debe ser capaz de:

1. Mostrar todos los productos disponibles.
2. Buscar un producto por su código.
3. Modificar el precio de un producto.
4. Eliminar un producto del inventario.
5. Mostrar todos los productos cuyo precio esté en un rango dado por el usuario.

Para realizar este ejercicio, usa las siguientes estructuras:

- **Tuplas** para almacenar los detalles de cada producto (código, descripción, precio).
- **Diccionarios** para almacenar el inventario, donde el código del producto será la clave y el valor será una tupla con la descripción y el precio.

Instrucciones:

1. **Crear un diccionario** que contenga el inventario inicial con al menos 5 productos.
    - Cada clave será un código de producto (ej. "A001", "A002").
    - Cada valor será una tupla con la descripción del producto y su precio.
2. Implementar las siguientes funciones:
    - mostrar_inventario(inventario): Muestra todos los productos del inventario con su código, descripción y precio.

- buscar_producto(inventario, codigo): Busca un producto por su código. Si existe, muestra su descripción y precio.
- modificar_precio(inventario, codigo, nuevo_precio): Modifica el precio de un producto dado su código.
- eliminar_producto(inventario, codigo): Elimina un producto del inventario usando su código.
- productos_por_rango_de_precio(inventario, min_precio, max_precio): Muestra todos los productos cuyo precio esté entre min_precio y max_precio.

Ejemplo de Inventario:

```
inventario = {
    "A001": ("Laptop", 1500),
    "A002": ("Mouse", 25),
    "A003": ("Teclado", 45),
    "A004": ("Monitor", 300),
    "A005": ("Impresora", 120)
}
```

Ejemplo de salida esperada:

- **Mostrar inventario:**

```
mostrar_inventario(inventario)
```

Salida:

```
Código: A001, Descripción: Laptop, Precio: $1500
Código: A002, Descripción: Mouse, Precio: $25
Código: A003, Descripción: Teclado, Precio: $45
Código: A004, Descripción: Monitor, Precio: $300
Código: A005, Descripción: Impresora, Precio: $120
```

- **Buscar producto por código:**

```
buscar_producto(inventario, "A003")
```

Salida:

```
Producto encontrado: Teclado, Precio: $45
```

- **Modificar el precio de un producto:**

```
modificar_precio(inventario, "A004", 350)
```

Salida:

```
El precio del producto con código A004 ha sido actualizado a $350
```

- **Eliminar un producto del inventario:**

```
eliminar_producto(inventario, "A002")
```

Salida:

```
El producto con código A002 ha sido eliminado del inventario.
```

- **Mostrar productos dentro de un rango de precios:**

```
productos_por_rango_de_precio(inventario, 100, 500)
```

Salida:

```
Productos en el rango de precio entre $100 y $500:
Código: A004, Descripción: Monitor, Precio: $350
Código: A005, Descripción: Impresora, Precio: $120
```