listas-tuplas-diccionarios

February 23, 2024

0.1 Tuplas

[3]:
```
# Una tupla es un objeto inmutable, no pueden cambiar los datos una vez␣
↪definidos

tupla = (1, 2, "Hola a todos", ("Segunda tupla elemento 1", "Mucho contenido"),␣
↪"Ultimo valor")
print (tupla)
```

```
(1, 2, 'Hola a todos', ('Segunda tupla elemento 1', 'Mucho contenido'), 'Ultimo
valor')
```

[6]:
```
# Acceso a tuplas
# Inicia en la posición 0 en adelante, y -1 indica el último valor
print(tupla[0]) # posición 0

print(tupla[-1]) # Ultima posicion

print(tupla[1:3]) # se puede usar X:Y para indicar de rango X hasta Y

print(tupla[-3]) # Hola a todos
```

```
1
Ultimo valor
(2, 'Hola a todos')
Hola a todos
```

[20]:
```
# Adicional a eso, si seguimos haciendo más pequeño el numero negativo como -2
print(tupla[-2]) # Penúltima posicion
# esta tupla resultante es el penúltimo elemento de la tupla principal
```

```
('Segunda tupla elemento 1', 'Mucho contenido')
```

[10]:
```
print(tupla[:])
```

```
(1, 2, 'Hola a todos', ('Segunda tupla elemento 1', 'Mucho contenido'), 'Ultimo
valor')
```

[21]:
```
# Asignar valores de una tupla a variables
tuplaXYZ = ("x", "y", "z")
x, y, z = tuplaXYZ
print("Estos son los valores de la tuplaXYZ: ", tuplaXYZ)
print("Valor de x : ", x)
print("Valor de y : ", y)
print("Valor de z : ", z)
```

```
Estos son los valores de la tuplaXYZ:  ('x', 'y', 'z')
Valor de x :  x
Valor de y :  y
Valor de z :  z
```

[11]:
```
# Crear una tupla a partir de 2 tuplas
tupla1 = (1, 2, 3)
tupla2 = (4, 5, 6)
tupla3 = tupla1 + tupla2
print(tupla3)
```

```
(1, 2, 3, 4, 5, 6)
```

[14]:
```
# Contar las ocurrencias de un elemento en una tupla
tupla_concurrencia = (1, 1, 1, 3, 3, 2, 2, 2, 2)
print(tupla_concurrencia.count(2))
```

```
4
```

[15]:
```
# Mostrar el índice de la primera ocurrencia de un elemento
print(tupla_concurrencia.index(2))
```

```
5
```

0.2 Listas

[16]:
```
# Una lista es un objeto mutable, se pueden cambiar los datos después de␣
 ↪definidos

lista = [1, 2, "Hola a todos", ["Segunda lista elemento 1", "Mucho contenido"],␣
 ↪"Ultimo valor"]
print(lista)
```

```
[1, 2, 'Hola a todos', ['Segunda lista elemento 1', 'Mucho contenido'], 'Ultimo
valor']
```

[17]:
```
# Acceso a listas
# Inicia en la posición 0 en adelante, y -1 indica el último valor
print(lista[0])  # posición 0
```

```
print(lista[-1])  # Ultima posicion

print(lista[1:3])  # se puede usar X:Y para indicar de rango X hasta Y
```

```
1
Ultimo valor
[2, 'Hola a todos']
```

[18]:
```
# Adicional a eso, si seguimos haciendo más pequeño el numero negativo como -2
print(lista[-2])  # Penúltima posicion
```

```
['Segunda lista elemento 1', 'Mucho contenido']
```

[20]:
```
# Asignar valores de una lista a variables
listaXYZ = ["x", "y", "z"]
x, y, z = listaXYZ
print("Estos son los valores de la listaXYZ: ", listaXYZ)
print("Valor de x : ", x)
print("Valor de y : ", y)
print("Valor de z : ", z)
```

```
Estos son los valores de la listaXYZ:  ['x', 'y', 'z']
Valor de x :  x
Valor de y :  y
Valor de z :  z
```

[21]:
```
# Crear una lista a partir de 2 listas
lista1 = [1, 2, 3]
lista2 = [4, 5, 6]
lista3 = lista1 + lista2
print(lista3)
```

```
[1, 2, 3, 4, 5, 6]
```

[30]:
```
# Contar las ocurrencias de un elemento en una lista
lista_concurrencia = [1, 1, 1, 3, 3, 2, 2, 2, 2]
print(lista_concurrencia.count(3))
```

```
2
```

[31]:
```
# Mostrar el índice de la primera ocurrencia de un elemento
print(lista_concurrencia.index(2))
```

```
5
```

0.3 Listas vs Tuplas mutabilidad

0.3.1 Lista mutable

[22]:
```
# Lista mutable
lista_mut = [1, 2, 3, 4, 5]
print("Lista original:", lista_mut)

# Modificar un elemento de la lista
lista_mut[2] = 10
print("Lista después de la modificación:", lista_mut)

# Añadir un nuevo elemento a la lista
lista_mut.append(6)
print("Lista después de agregar un elemento:", lista_mut)

# Eliminar un elemento de la lista
del lista_mut[1]
print("Lista después de eliminar un elemento:", lista_mut)
```

```
Lista original: [1, 2, 3, 4, 5]
Lista después de la modificación: [1, 2, 10, 4, 5]
Lista después de agregar un elemento: [1, 2, 10, 4, 5, 6]
Lista después de eliminar un elemento: [1, 10, 4, 5, 6]
```

[24]:
```
# Agregar el contenido de una lista a otra
lista_mut_2 = [6, 7, 8]
print("Lista original mut 2:", lista_mut_2)

# Lista mutable
lista_mut = [1, 2, 3, 4, 5]

print("Lista original:", lista_mut)

# Modificar un elemento de la lista
lista_mut.extend(lista_mut_2)
print("Lista después de la modificación:", lista_mut)
```

```
Lista original mut 2: [6, 7, 8]
Lista original: [1, 2, 3, 4, 5]
Lista después de la modificación: [1, 2, 3, 4, 5, 6, 7, 8]
```

0.3.2 Tupla inmutable

[23]:
```
# Tupla inmutable
tupla_imm = (1, 2, 3, 4, 5)
print("\nTupla original:", tupla_imm)
```

```python
# Intentar modificar un elemento de la tupla generará un error
try:
    tupla_imm[2] = 10  # Esto generará un TypeError
except TypeError as e:
    print("Error al intentar modificar la tupla:", e)

# Intentar agregar un nuevo elemento a la tupla generará un error
try:
    tupla_imm.append(6)  # Esto generará un AttributeError
except AttributeError as e:
    print("Error al intentar agregar un elemento a la tupla:", e)

# Intentar eliminar un elemento de la tupla generará un error
try:
    del tupla_imm[1]  # Esto generará un TypeError
except TypeError as e:
    print("Error al intentar eliminar un elemento de la tupla:", e)
```

```
Tupla original: (1, 2, 3, 4, 5)
Error al intentar modificar la tupla: 'tuple' object does not support item
assignment
Error al intentar agregar un elemento a la tupla: 'tuple' object has no
attribute 'append'
Error al intentar eliminar un elemento de la tupla: 'tuple' object doesn't
support item deletion
```

0.3.3 Iterar sobre listas

[44]:
```python
# Crear una lista
mi_lista = [1, 2, 3, 4, 5]

# Iterar sobre la lista e imprimir cada elemento
for elemento in mi_lista:
    print(elemento)
```

```
1
2
3
4
5
```

0.3.4 Iterar sobre tuplas

[45]:
```python
# Crear una tupla
mi_tupla = (10, 20, 30, 40, 50)
```

```python
# Iterar sobre la tupla e imprimir cada elemento
for elemento in mi_tupla:
    print(elemento)
```

```
10
20
30
40
50
```

0.4 Diccionarios

1. Diccionario Básico:

[25]:
```python
estudiante = {
    "nombre": "Juan",
    "edad": 20,
    "curso": "Matemáticas"
}
print(estudiante)
```

```
{'nombre': 'Juan', 'edad': 20, 'curso': 'Matemáticas'}
```

2. Diccionario Anidado:

[35]:
```python
universidad = {
    "estudiantes": {
        "Juan": {"edad": 20, "curso": "Matemáticas"},
        "Ana": {"edad": 22, "curso": "Historia"},
        "2018300001": estudiante
    },
    "profesores": {
        "Dr. Pérez": {"departamento": "Matemáticas"},
        "Prof. Gómez": {"departamento": "Historia"}
    }
}
print(universidad)
```

```
{'estudiantes': {'Juan': {'edad': 20, 'curso': 'Matemáticas'}, 'Ana': {'edad':
22, 'curso': 'Historia'}}, 'profesores': {'Dr. Pérez': {'departamento':
'Matemáticas'}, 'Prof. Gómez': {'departamento': 'Historia'}}}
```

3. Uso de Diccionario para Contar Ocurrencias:

[36]:
```python
frase = "python es divertido"
conteo_letras = {}
for letra in frase:
    if letra.isalpha():
        letra = letra.lower()
        conteo_letras[letra] = conteo_letras.get(letra, 0) + 1
```

```
print(conteo_letras)
```

```
{'p': 1, 'y': 1, 't': 2, 'h': 1, 'o': 2, 'n': 1, 'e': 2, 's': 1, 'd': 2, 'i': 2,
'v': 1, 'r': 1}
```

4. Diccionario con Listas:

[37]:
```
curso = {
    "nombre": "Programación",
    "alumnos": ["Juan", "Ana", "Carlos"],
    "profesor": "Dr. Rodríguez"
}
print(curso)
```

```
{'nombre': 'Programación', 'alumnos': ['Juan', 'Ana', 'Carlos'], 'profesor':
'Dr. Rodríguez'}
```

5. Diccionario con Tuplas:

[38]:
```
contactos = {
       "amigos": [("Juan", "123456"), ("Ana", "789012")],
       "familia": [("Mamá", "111111"), ("Papá", "222222")]
   }
print(contactos)
```

```
{'amigos': [('Juan', '123456'), ('Ana', '789012')], 'familia': [('Mamá',
'111111'), ('Papá', '222222')]}
```

6. Diccionario con Funciones:

[43]:
```
def cuadrado(x):
    return x * x

operaciones = {
    "cuadrado": cuadrado,
    "doble": lambda x: x * 2,
    "mitad": lambda x: x / 2
}

# Imprimir el resultado de la función "doble"
print(operaciones["cuadrado"](5))  # Salida: 10

# Imprimir el resultado de la función "doble"
print(operaciones["doble"](5))  # Salida: 10

# Imprimir el resultado de la función "mitad"
print(operaciones["mitad"](8))  # Salida: 4.0
```

```
25
10
4.0
```

7. Diccionario con Diccionarios Internos:

[40]:
```python
libros = {
    "libro1": {"titulo": "Python 101", "autor": "A. Smith"},
    "libro2": {"titulo": "Data Science Intro", "autor": "B. Johnson"}
}
print(libros)
```

```
{'libro1': {'titulo': 'Python 101', 'autor': 'A. Smith'}, 'libro2': {'titulo':
'Data Science Intro', 'autor': 'B. Johnson'}}
```

8. Diccionario para Configuración:

[41]:
```python
configuracion = {
    "idioma": "es",
    "tema": "oscuro",
    "tamaño_fuente": 14
}
print(configuracion)
```

```
{'idioma': 'es', 'tema': 'oscuro', 'tamaño_fuente': 14}
```

Estos son solo ejemplos para ilustrar diversas formas en las que puedes usar diccionarios en Python. Los diccionarios son flexibles y pueden adaptarse a una variedad de situaciones en las que necesitas mapear claves a valores.

0.4.1 9. Iteracion

[2]:
```python
mi_diccionario = {"clave1": 1, "clave2": 2, "clave3": 3}

# Longitud del diccionario
cantidad_elementos = len(mi_diccionario)
print(f"Cantidad de elementos en el diccionario: {cantidad_elementos}")

# Verificar si una clave está presente
if "clave2" in mi_diccionario:
    print("La clave 'clave2' está presente en el diccionario")
```

```
Cantidad de elementos en el diccionario: 3
La clave 'clave2' está presente en el diccionario
```

[ ]:
```python
# Agregar un nuevo elemento
mi_diccionario["nueva_clave"] = 4
print("Diccionario después de agregar 'nueva_clave':", mi_diccionario)

# Actualizar un valor existente
```

```
mi_diccionario["clave1"] = 10
print("Diccionario después de actualizar 'clave1':", mi_diccionario)

# Iterar sobre claves
for clave in mi_diccionario:
    print(clave)

# Iterar sobre valores
for valor in mi_diccionario.values():
    print(valor)

# Iterar sobre pares clave-valor
for clave, valor in mi_diccionario.items():
    print(clave, valor)
```

[ ]:
```
# Eliminar un elemento por clave
clave_a_eliminar = "clave3"
if clave_a_eliminar in mi_diccionario:
    del mi_diccionario[clave_a_eliminar]
    print(f"Elemento con clave '{clave_a_eliminar}' eliminado. Diccionario␣
 ↪resultante:", mi_diccionario)

# Iterar sobre claves
print("\nIterar sobre claves:")
for clave in mi_diccionario:
    print(clave)

# Iterar sobre valores
print("\nIterar sobre valores:")
for valor in mi_diccionario.values():
    print(valor)

# Iterar sobre pares clave-valor
print("\nIterar sobre pares clave-valor:")
for clave, valor in mi_diccionario.items():
    print(clave, valor)
```