


UT. Python III

Ejemplos mezclas listas, tuplas, ...

ÍNDICE DE CONTENIDO

1.Ejemplo 1: Listas dentro de tuplas

4

	LISTA	TUPLA	CONJUNTO	DICCIONARIO
Características	Datos heterogéneos. Acepta repetidos. Elementos mutables y accesibles por índice.	Datos heterogéneos. Acepta repetidos. Elementos inmutables y accesibles por índice.	Datos heterogéneos. Elementos de tipos inmutables, no accesibles por índice. No admite repetidos.	Claves de tipos inmutables, únicas (no admite repetidas). Valores de cualquier tipo, accesibles por clave, admiten repetidos. Pares heterogéneos.
Crear	<code>[]</code> ó <code>list()</code>	<code>()</code> ó <code>tuple()</code>	<code>set()</code>	<code>{}</code> ó <code>dict()</code>
Inicializar con datos	<ul style="list-style-type: none"> <code>[elemento1, elemento2, elemento3]</code> <code>list(iterable)</code> para crear con los elementos de una secuencia o contenedor. 	<ul style="list-style-type: none"> <code>elemento1, elemento2, elemento3</code> (los paréntesis son opcionales) <code>tuple(iterable)</code> para crear con los elementos de una secuencia o contenedor. 	<ul style="list-style-type: none"> <code>{elemento1, elemento2, elemento3}</code> <code>set(iterable)</code> para crear con los elementos de una secuencia o contenedor. 	<ul style="list-style-type: none"> <code>{clave1: valor1, clave2: valor2, clave3: valor3}</code> <code>dict([(c1,v1), (c2,v2)])</code>
Insertar elementos	<ul style="list-style-type: none"> <code>lista.append(elemento)</code> agrega elemento al final <code>lista.insert(posición, elemento)</code> inserta elemento en posición del índice. 	Sólo es posible inicializarla en la creación. Luego no se puede agregar, porque es inmutable.	<code>conjunto.add(elemento)</code> donde elemento es un solo dato.	<ul style="list-style-type: none"> <code>dicc[clave]=valor</code> <code>dicc.update({c3:v3, c4:v4})</code>
Acceder a un elemento	<ul style="list-style-type: none"> <code>lista[indice]</code> lectura/escritura. <code>lista[inicio:fin:paso]</code> Iterando por sus elementos. 	<ul style="list-style-type: none"> <code>tupla[indice]</code> sólo lectura <code>tupla[inicio:fin:paso]</code> Iterando por sus elementos. 	Iterando por sus elementos (no soporta índices). No es posible modificar elementos.	<ul style="list-style-type: none"> <code>dicc[clave]</code> <code>dicc.get(clave, val_defecto)</code> Iterando por sus elementos.
Iterar usando el índice	<code>for i in range(len(lista)):</code> <code>print(lista[i])</code>	<code>for i in range(len(tupla)):</code> <code>print(tupla[i])</code>	No tiene índice.	No tiene índice (el "índice" son las claves).
Iterar con for para iterables	<code>for elemento in lista</code>	<code>for elemento in tupla</code>	<code>for elemento in conjunto</code>	<ul style="list-style-type: none"> <code>for clave in dicc.keys()</code> <code>for valor in dicc.values()</code> <code>for c,v in dicc.items()</code>
Pertenencia	<code>elemento in lista</code>	<code>elemento in tupla</code>	<code>elemento in conjunto</code>	<ul style="list-style-type: none"> <code>clave in dicc.keys()</code> <code>value in dicc.values()</code>
Eliminar elemento	<ul style="list-style-type: none"> <code>del lista[indice]</code> <code>lista.remove(elemento)</code> elimina la primera ocurrencia 	No es posible porque son inmutables.	<ul style="list-style-type: none"> <code>conjunto.remove(elemento)</code> <code>conjunto.discard(elemento)</code> 	<code>del dicc[clave]</code>
Cantidad de elementos	<code>len(lista)</code>	<code>len(tupla)</code>	<code>len(conjunto)</code>	<code>len(dicc)</code>
Vaciar	<code>lista.clear()</code>	No es posible porque son inmutables.	<code>conjunto.clear()</code>	<code>dicc.clear()</code>

1.EJEMPLO 1: LISTAS DENTRO DE TUPLAS

Define una tupla que contenga tres listas diferentes. Accede y modifica un elemento dentro de una de las listas.

```
tupla_con_listas = ([1, 2, 3], [4, 5, 6], [7, 8, 9]) → una tupla con tres listas
```

```
elemento = tupla_con_listas[1][1] → Accedo al 2º elemento de la 2ª lista
```

```
print("Elemento accedido:", elemento) → Elemento accedido: 5
```

```
tupla_con_listas[0][2] = 10 → Modifico el 3º elemento de la 1ª lista
```

```
print("Tupla después de la modificación:", tupla_con_listas)
```

2.EJEMPLO 2: TUPLAS CON DICCIONARIOS

Define una tupla que contenga dos diccionarios diferentes. Accede y modifica un valor dentro de uno de los diccionarios.

```
tupla_con_diccionarios = ({"nombre": "Alice", "edad": 30}, {"nombre": "Bob", "edad": 25})  
→ Defino una tupla con dos diccionarios
```

```
nombre = tupla_con_diccionarios[0]["nombre"]  
→ Accedo al valor asociado a la clave 'nombre' del 1º diccionario
```

```
print("Nombre accedido:", nombre) → Nombre accedido: Alice
```

```
tupla_con_diccionarios[1]["edad"] = 26  
→ Modifico el valor asociado a la clave 'edad' del 2º diccionario
```

```
print("Tupla después de la modificación:", tupla_con_diccionarios)
```

3.EJEMPLO 3: LISTAS CON DICCIONARIOS

Define una lista que contenga dos diccionarios. Accede y modifica un valor dentro de uno de los diccionarios.

```
lista_con_diccionarios = [{"ciudad": "Sevilla", "pais": "España"}, {"ciudad": "Lisboa", "pais": "Portugal"}]
```

→ Defino una lista con dos diccionarios

```
ciudad = lista_con_diccionarios[1]["ciudad"]
```

→ Accedo al valor asociado a la clave 'ciudad' del 2º diccionario

```
print("Ciudad accedida:", ciudad) → Ciudad accedida: Lisboa
```

```
lista_con_diccionarios[0]["pais"] = "España"
```

→ Modifico el valor asociado a la clave 'pais' del 1º diccionario

```
print("Lista después de la modificación:", lista_con_diccionarios)
```

4.EJEMPLO 4: CONJUNTO CON TUPLAS

Define un conjunto que contenga varias tuplas. Accede a un elemento dentro de una de las tuplas.

```
conjunto_con_tuplas = {"manzana", "roja"}, {"plátano", "amarillo"}, {"uva", "morado"}
```

→ Defino un conjunto con varias tuplas

```
lista_de_tuplas = list(conjunto_con_tuplas)
```

→ Convierto el conjunto en lista para acceder a los elementos

```
fruta_color = lista_de_tuplas[0][1] → Accedo al 2º elemento de la 1ª tupla
```

```
print("Color accedido:", fruta_color) → Color accedido: XXX
```

Resultado: depende del orden, podría ser "roja", "amarillo" o "morado"

Nota: Los conjuntos no tienen orden, por lo que al convertir a una lista, el orden puede variar

```
$ python3 La_Mezcla.py
Ejemplo 1: Listas dentro de tuplas
Elemento accedido: 5
Tupla después de la modificación: ([1, 2, 10], [4, 5, 6], [7, 8, 9])
Ejemplo 2: Tupla con diccionarios
Nombre accedido: Alice
Tupla después de la modificación: ({'nombre': 'Alice', 'edad': 30}, {'nombre': 'Bob', 'edad': 26})
$ python3 La_Mezcla.py
*****
Ejemplo 1: Listas dentro de tuplas
*****
Elemento accedido: 5
Tupla después de la modificación: ([1, 2, 10], [4, 5, 6], [7, 8, 9])
*****
Ejemplo 2: Tupla con diccionarios
*****
Nombre accedido: Alice
Tupla después de la modificación: ({'nombre': 'Alice', 'edad': 30}, {'nombre': 'Bob', 'edad': 26})
*****
Ejemplo 3: Listas con diccionarios
*****
Ciudad accedida: Lisboa
Lista después de la modificación: [{'ciudad': 'Sevilla', 'pais': 'España'}, {'ciudad': 'Lisboa', 'pais': 'Portugal'}]
*****
Ejemplo 4: Conjunto con Tuplas
*****
Color accedido: amarillo
$ █
```