Colecciones Python: Tuplas

Contenidos

- Introducción
- Listas y tuplas

Introducción

al indice

Muy similares a las listas. Si en Notebooks anteriores definíamos a las listas como ordenadas y mutables, las tuplas son ordenadas e inmutables.

- **Inmutables**: una vez creada la tupla con sus elementos iniciales, no se puede modificar
- **Ordenadas**: podemos acceder a sus elementos a través del indice y reordenar la tupla según queramos

Las tuplas se suelen usar para pequeñas colecciones de datos que no van a cambiar a lo largo del programa, como es el caso de las constantes.

Si las listas se construían mediante corchetes `[]`, las tuplas lo hacen con los paréntesis `()`

In [3]:
```
tupla_ejemplo =  (3, "Texto")
print(tupla_ejemplo)
print(type(tupla_ejemplo))

print("El primer elemento es", tupla_ejemplo[0])
print("El primer elemento es", tupla_ejemplo[1])
```

```
(3, 'Texto')
<class 'tuple'>
El primer elemento es 3
El primer elemento es Texto
```

Los paréntesis ya se usan para reordenar operaciones `5 * (3 + 4)`, por lo que hay que añadir una coma cuando definamos una tupla con un único elemento, sino Python lo interpretará como un número.

In [6]:
```
mala_tupla = (24)
print(type(mala_tupla), mala_tupla)
buena_tupla =(24,)
print(type(buena_tupla), buena_tupla)
```

```
<class 'int'> 24
<class 'tuple'> (24,)
```

In [7]:
```
# Anidación tupla
tupla_anid = (1, 2, 3, ("A", "B", "C"))
print(tupla_anid)
print(len(tupla_anid))
```

```
(1, 2, 3, ('A', 'B', 'C'))
4
```

In [8]:
```
print(tupla_anid[2])
print(tupla_anid[3][1])
```

```
3
B
```

In [9]:
```
# Son iterables
for i in tupla_anid:
    print(i)
```

```
1
2
3
('A', 'B', 'C')
```

In [10]:
```
# Slicing
tupla_bis = tupla_anid[2:]
print(tupla_bis)
```

```
(3, ('A', 'B', 'C'))
```

ERRORES tuplas

Cuidado que las tuplas son inmutables, y una vez creadas no las podrás modificar después

In [11]:
```
tupla_error = (1,2,3,4,5)
tupla_error[3] = 23
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 2
      1 tupla_error = (1,2,3,4,5)
----> 2 tupla_error[3] = 23

TypeError: 'tuple' object does not support item assignment
```

In [12]:
```
# Si queremos añadir elementos, podemos meterlos en otra tupla y sumarl
tupla_1 = (1,2,3)
tupla_2 = (4,5,6)

print(tupla_1 + tupla_2)
tupla_3 = tupla_1 + tupla_2
tupla_1 = tupla_1 + tupla_2
print(tupla_3)
print(tupla_1)
```

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

Listas y tuplas

al indice

Podemos combinar listas y tuplas que no tendremos ningún problema, siempre y cuando respetemos las propiedades de cada tipo de dato.

In [14]:
```
lista_ejemplo = ["Fresas", ("Naranjas", "Limones"), "Kiwis"]
tupla_lista = ("Fresas", ["Naranjas", "Limones"], "Kiwis")
print(lista_ejemplo, type(lista_ejemplo))
print(tupla_lista, type(tupla_lista))
```

```
['Fresas', ('Naranjas', 'Limones'), 'Kiwis'] <class 'list'>
('Fresas', ['Naranjas', 'Limones'], 'Kiwis') <class 'tuple'>
```

In [ ]: