Python Colecciones: Listas

Las colecciones son una manera de agrupar varios elementos. En otros notebooks vimos cómo funcionan las listas, que es la colección más usada, pero se trata solo de la punta del iceberg. Con Python tenemos varias manera de almacenar conjuntos de datos, dependiendo del tipo de dato, finalidad, tipo de acceso y rendimiento.

En este primer notebook vamos a repasar las listas y luego continuaremos con los otros tipo de colecciones básicos que tenemos en Python.

Listas

Contenidos

Introducción

al indice

Ya conocemos bastante las listas. Veamos un repaso de lo que podemos hacer con ellas, así como algunas funcionalidades nuevas

In [5]:
```
# Listas de números, strings, booleanos, con elementos repetidos, lista
nums = [6, 2, 8, 3, 4, 5, 5]
months = ["Enero", "Febrero", "Marzo"]
mix = [5, 7, "Abril", True, None, ["Blanco", "Negro"]]
```

Acceso

al indice

Entre corchetes introducimos el indice del elemento al que queremos acceder

In [7]:
```
months = ["Enero", "Febrero", "Marzo"]
print(months[0])
print(months[1])
```

```
Enero
Febrero
```

In [ ]:
```

```

Si planteamos el problema al revés. Tenemos los valores de la lista y lo que queremos es obtener el índice de esos valores dentro de la lista

In [9]:
```
months = ["Enero", "Febrero", "Marzo", "Marzo"]
print(months.index("Febrero"))
```

```
1
```

Slicing: usamos slicing para acceder a varios elementos seguidos de la lista

In [14]:
```
degrees = [22, 34, 15, 26, 18, 22]
sub_degrees = degrees[1:3] # ojo no se incluye el último índice, sino e
print(sub_degrees)
sub_degrees = degrees[1:4]
print(sub_degrees)
sub_degrees = degrees[1:6]
sub_degrees_alt = degrees[1:len(degrees)]
sub_degrees_alt_2 = degrees[1:]
print(sub_degrees)
print(sub_degrees_alt)
print(sub_degrees_alt_2)
```

```
[34, 15]
[34, 15, 26]
[34, 15, 26, 18, 22]
[34, 15, 26, 18, 22]
[34, 15, 26, 18, 22]
```

Modificar elementos

al indice

Las listas son mutables, por lo que podremos modificarlas

In [15]:
```
# Accediendo mediante el indice
months = ["Enero", "Febrero", "Marzo"]
months[1] = "Abril"
print(months)
```

```
['Enero', 'Abril', 'Marzo']
```

Añadir elementos

al indice

Se añade al final de la lista si usamos `append`, o si queremos en un lugar concreto, mediante `insert`

In [17]:
```
# Con append lo añadimos al final de la lista
motorcycles = ['honda', 'yamaha', 'suzuki']
motorcycles.append("kawasaki")
print(motorcycles)
motorcycles.insert(3,"bultaco")
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki', 'kawasaki']
['honda', 'yamaha', 'suzuki', 'bultaco', 'kawasaki']
```

In [18]:
```
# Es muy comun crear una lista vacia, y a lo largo del programa, ir aña
lista_motos = []
lista_motos.append("primera marca")
print(lista_motos)
lista_motos.insert(0,"siguiente_marca")
print(lista_motos)
```

```
['primera marca']
['siguiente_marca', 'primera marca']
```

Eliminar elementos

al indice

Para eliminar elementos se usar `remove`. Si no existe, da error, así que cuidado con esta sentencia.

En ocasiones resulta útil quedarnos con el elemento eliminado. Para ello usamos `pop()`, que elimina el elemento que le indiquemos, y además devuelve ese elemento por lo que podremos guardarlo en una variable para usarlo después.

In [19]:
```
cars = ["VW", "Seat", "BMW", "VW"]
cars.remove("VW")
```

```
print(cars)
```

```
['Seat', 'BMW', 'VW']
```

In [20]:
```
# Eliminar elementos por indice, y guarda ese valor en una variable
cars = ["VW", "Seat", "BMW", "VW"]
eliminado = cars.pop(1)
print(cars)
print(eliminado)
```

```
['VW', 'BMW', 'VW']
Seat
```

ERRORES remove

In [21]:
```
cars = ["VW", "Seat", "BMW", "VW"]
cars.remove("Tesla")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[21], line 2
      1 cars = ["VW", "Seat", "BMW", "VW"]
----> 2 cars.remove("Tesla")

ValueError: list.remove(x): x not in list
```

Métodos: ordenar, tamaño, invertir orden, ocurrencia

al indice

En este apartado veremos los métodos más útiles, pero podrás consultar el resto en este enlace

In [26]:
```
degrees = [22, 34, 15, 26, 18, 22]
#ordenar
degrees.sort()
print(degrees)
degrees.sort(reverse=True)
print(degrees)
#contar numero de elementos
print(len(degrees))
#invertir lista
degrees.reverse()
print(degrees)
# Ocurrencias de un elemento
print(degrees.count(22))
```

```
[15, 18, 22, 22, 26, 34]
[34, 26, 22, 22, 18, 15]
6
[15, 18, 22, 22, 26, 34]
2
```

ERRORES indice

Mucho cuidado cuando accedemos a los elementos de la lista. Es un error muy habitual acceder a un índice que no existe en la lista.

Si estamos accediendo al ultimo elemento, en vez de poner el numero de su indice, poner mejor `-1`, y asi evitamos errores

In [27]:
```
degrees = [22, 34, 15, 26, 18, 22]
print(degrees[123])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[27], line 2
      1 degrees = [22, 34, 15, 26, 18, 22]
----> 2 print(degrees[123])

IndexError: list index out of range
```

Si tenemos este tipo de errores y no sabemos resolverlos, lo mejor es imprimir la longitud de la lista, y todos los elementos.

In [28]:
```
ultimo_elemento = degrees[len(degrees)-1]
print(ultimo_elemento)
```

```
22
```

In [ ]:
```

```