

# **Python: listas**

**Sistemas de gestión empresarial – 148fa (DAM)**

# Listas vs tuplas

- **Tuplas:** variables que guardan datos que no pueden ser modificados. Los datos pueden ser de diferentes tipos.
- **Listas:** son variables que guardan datos de diferentes tipos, pero que sí que pueden ser modificados.

```
# Lista
mi_lista = [1, 2, 3]
mi_lista[0] = 10 # Se puede modificar

# Tupla
mi_tupla = (1, 2, 3)
# mi_tupla[0] = 10 # Esto daría un error porque es inmutable
```

# Listas

- Los elementos de las listas se escriben dentro de los corchetes [].
- El comportamiento de las listas es parecido al de las cadenas de caracteres o strings:
  - Función len(lista) : devuelve la longitud de la lista
  - Acceder a un elemento de la lista: mi\_lista[indice]
  - El primer elemento de la lista se encuentra en el índice 0

```
colores = ['Rojo', 'Verde', 'Azul']
```

```
print len(colores)    # 3
```

```
print colores[0]    # rojo
```

```
print colores [2]    # azul
```

colores

lista

'Rojo'	'Verde'	'Azul'
0	1	2

# Listas

- Lista vacía: se representa con [], sin elementos dentro de los corchetes
- Unir dos listas : +

```
lista_vacia = []  
lista1 = [1, 2]  
lista2 = [3, 4]  
  
lista3 = lista1 + lista2  
print(lista3) # [1, 2, 3, 4]
```

# Listas

- Si asignamos la lista creada previamente a una variable, no crea una copia de la lista.
- Esta nueva variable, considerada de tipo lista, señalará la misma lista en la memoria (REFERENCIA).

```
colores = ['Rojo', 'Verde', 'Azul']
```

```
lista_nueva = colores # no copia la lista
```

colores →

lista\_nueva →

'Rojo'	'Verde'	'Azul'
--------	---------	--------

# Listas

- Para copiar el contenido de una lista en otra, podemos usar el método copy:

```
lista_original = [1, 2, 3]
lista_copia = lista_original.copy()
print(lista_copia)  # Resultado: [1, 2, 3]
```

# Recorrer listas

- **FOR-IN:** Podemos recorrer los elementos de la lista utilizando el bucle FOR.

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

```
for numero in lista_numeros:
```

```
    if (numero % 2 == 0):
```

```
        print ('El número',  
numero, ' es par')
```

```
    else:
```

```
        print ('El número',  
numero, ' es impar')
```

# Recorrer listas

- **WHILE:** De esta manera podemos controlar el índice y los saltos que queramos hacer sobre él.

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

```
i = 0
```

```
while i < len(lista_numeros):
```

```
    print(lista_numeros[i])
```

```
    i = i+2
```



# Buscar elementos en listas

- **IN:** Podemos buscar un elemento en una lista utilizando IN

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

if 3 in lista_numeros:

    print('El número se encuentra
en la lista')

else:

    print ('El número NO se
encuentra en la lista')
```

# Porciones de listas

- Podemos utilizar rangos de índices para acceder a porciones de una lista. Pero el índice que se escoja por la derecha será uno más del que realmente abarque. Es decir, si queremos coger los elementos 2 y 3 de una lista (cuyos índices serán 1 y 2) debemos escoger los índices desde el 1 al 3.

```
mi_lista = ["Elemento1", 2, "Elemento3", 4]
```

```
print(mi_lista[1:3]) # devolverá 2 y "Elemento3"
```

# Función range()

- Función que guarda números del 0 a n-1:

```
for i in range(0,50):
```

```
    print(i)
```

```
# También se imprimirán los números del 0 al 49.
```

```
# No llega a 50
```

# Métodos de listas

- **lista.append(elemento)** : añade un elemento al final de la lista. No devuelve una nueva lista, tan sólo modifica la original
- **lista.insert(indice, elemento)** : inserta el elemento en el índice que se indique, y desplaza los demás elementos a la derecha
- **lista.extend(lista2)** : añade todos los elementos de la lista2 al final de la lista original. La función extend() es lo mismo que utilizar + o +=
- **list.index(elem)** : busca el elemento dentro de la lista y devuelve su índice. Lanza el error "ValueError" si el elemento no se encuentra en la lista.

```
# Empezamos con una lista inicial
mi_lista = [1, 2, 3]

# 1. lista.append(elemento)
# Añade un elemento al final de la lista
mi_lista.append(4)
print(mi_lista) # Resultado: [1, 2, 3, 4]

# 2. lista.insert(indice, elemento)
# Inserta el elemento 0 en el índice 1 y desplaza los demás elementos a la derecha
mi_lista.insert(__index: 1, __object: 0)
print(mi_lista) # Resultado: [1, 0, 2, 3, 4]

# 3. lista.extend(lista2)
# Añade todos los elementos de lista2 al final de mi_lista
otra_lista = [5, 6, 7]
mi_lista.extend(otra_lista)
print(mi_lista) # Resultado: [1, 0, 2, 3, 4, 5, 6, 7]

# 4. lista.index(elem)
# Devuelve el índice del primer elemento que coincida con "3"
indice = mi_lista.index(3)
print(indice) # Resultado: 3 (el índice donde está el primer "3")
```

# Métodos de listas

- **lista.remove(elemento)** : busca la primera instancia del elemento que se indica y lo elimina. Lanza “ValueError” si no se encuentra
- **lista.sort()** : ordena la lista, pero no la devuelve.
- **lista.reverse()** : invierte el orden de los elementos de la lista.
- **lista.pop(índice)** : quita el elemento y lo devuelve (si hacemos print, por ejemplo). Si no indicamos el índice, quita el ultimo elemento de la lista.
- **lista.clear()** : Quita todos los elementos de la lista, dejándola vacía.
- **lista.count(elemento)** : Devuelve el número de veces que aparece el elemento que señalamos

```
# Lista inicial
mi_lista = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3]

# 1. lista.remove(elemento)
# Busca y elimina la primera instancia de "5"
mi_lista.remove(5)
print(mi_lista) # Resultado: [3, 1, 4, 1, 9, 2, 6, 5, 3]

# 2. lista.sort()
# Ordena la lista de menor a mayor
mi_lista.sort()
print(mi_lista) # Resultado: [1, 1, 2, 3, 3, 4, 5, 6, 9]

# 3. lista.reverse()
# Invierte el orden de los elementos en la lista
mi_lista.reverse()
print(mi_lista) # Resultado: [9, 6, 5, 4, 3, 3, 2, 1, 1]

# 4. lista.pop(índice)
# Quita el elemento en el índice 2 y lo devuelve
elemento = mi_lista.pop(2)
print(elemento) # Resultado: 5 (el elemento eliminado)
print(mi_lista) # Resultado: [9, 6, 4, 3, 3, 2, 1, 1]

# 5. lista.clear()
# Elimina todos los elementos de la lista
mi_lista.clear()
print(mi_lista) # Resultado: []

mi_lista = [1, 2, 3, 1, 4, 1, 5]

# 6. lista.count(elemento)
# Cuenta cuántas veces aparece "1" en la lista
veces = mi_lista.count(1)
print(veces) # Resultado: 3
```