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

```
v_tupla = (“cadena”, 3.7, 15, “otra cadena”, 75)
```

- **Listas**: son variables que guardan datos de diferentes tipos, pero que sí que pueden ser modificados

```
v_lista = [“cadena”, 3.7, 15, “otra cadena”, 75]
```

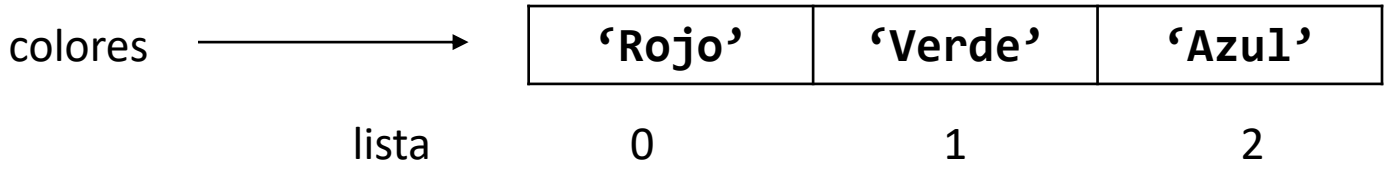
Listas

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
```



Listas

Lista vacía: se representa con [], sin elementos dentro de los corchetes

```
lista_vacía = []
```

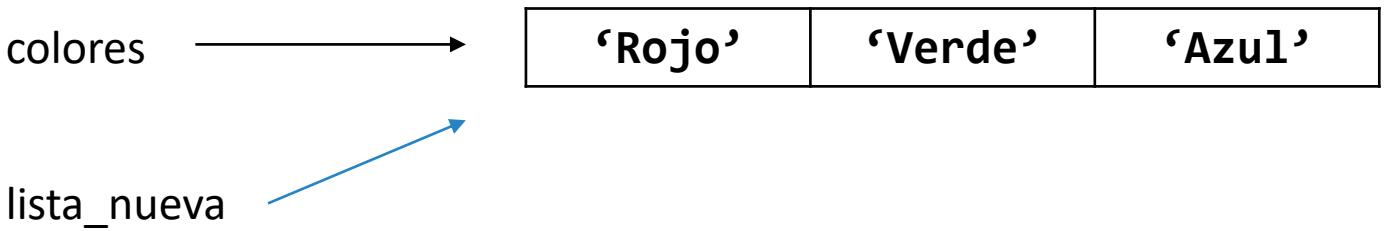
Unir dos listas : **+**

```
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.

```
colores = [‘Rojo’, ‘Verde’, ‘Azul’]

lista_nueva = colores  # no copia la lista
```



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’)
```

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’)
```

Listas

WHILE + len

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
```

Listas

Range

Función que guarda números del 0 a n-1

```
for i in range(50):
    print(i)
```

Se imprimirán los números del 0 al 49

Listas

Métodos en 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.

Listas

Métodos en 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(indice) : quita el elemento y lo devuelve (si hacemos print, por ejemplo). Si no indicamos el índice, quita el ultimo elemento de la lista.

Listas

Métodos en listas

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

Listas

Porciones de lista

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”
```