Listas en Python

Las listas son un tipo de colección ordenada y son el equivalente en otros lenguajes de programación a lo que se conoce como arreglos o vectores.
Las listas pueden tener enteros, booleanos, cadenas de texto, flotantes o inclusive listas. Los elementos de una lista van encerrados entre corchetes ([]) y separados por comas cada uno de ellos de la siguiente forma:

```
lista = [1, "dos", False, [45, "cien"]]
```

Como podemos ver, nuestra lista tiene los siguientes elementos: un entero (1), una cadena de texto ("dos"), un booleano (False) y otra lista con dos elementos (un entero 45 y una cadena de texto "cien").

Cada elemento de una lista cuenta con un índice. Este índice comienza en el 0 para el elemento numero 1.

Ejemplo para acceder a un elemento de la lista

```
num = lista[1]
```

Lo que hicimos en el anterior ejemplo fue asignarle a la variable num el elemento con índice uno de la lista (lista), en este caso ("dos"), por lo tanto la variable num va a contener la cadena de texto "dos".
Ahora, para que nos muestre el contenido de nuestra nueva variable, solo tendremos que pedirlo con un print:

```
print num
```

Ahora, como hacemos para acceder a los elementos de una lista que están dentro de otra lista?

Ejemplo: quiero acceder al primer elemento de la lista que esta dentro de la lista:

```
lista = [1, "dos", False, [45, "cien"]]
```

El elemento 4 de la lista es otra lista. Los indices se empiezan a contar desde el 0, por lo tanto el indice del elemento 4 es el 3. Y como el elemento 4 también es una lista voy a poder acceder a sus elementos por sus indices, en este caso al indice 0 porque queremos interactuar con el primer elemento.

```
num = lista[3][0] #Ahora num vale 45
```

Si nosotros queremos ver solo una porción de la lista tendríamos que hacer lo siguiente:
[inicio:fin]

Por ejemplo, para ver los primeros tres elementos de la lista

```
lista = [1, “dos”, False, [45, “cien”]]
ejemplo = lista[0:3]
```

Aqui lo que hacemos es asignarle a ejemplo los primeros 3 elementos de la lista (lista), o sea, los elementos desde el indice 0 al 3, este ultimo no se incluye.

Si en lugar de `ejemplo = lista[0:3]` hacemos `ejemplo = lista[0:]`
Asigna desde el elemento 0 hasta el final de la lista porque no indicamos fin. Lo mismo seria si no indicamos un inicio: `ejemplo = lista[:3],` asignamos desde el comienzo.

Si nosotros queremos ver algunos elementos, ejemplo: uno si uno no hacemos lo siguiente:

[inicio:fin:salto]

```
lista = [1, “dos”, False, [45, “cien”]]

ejemplo = lista[0:3:2] #Con el 2 saltea de uno en uno

print ejemplo
```

También podemos modificar una fracción de una lista de la siguiente manera:

```
lista = [1, “dos”, False, [45, “cien”]]

lista[0:2] = [5, "mil"]
```

Si imprimimos la lista nos queda de la siguiente manera:

```
5, "mil", False, [45, "cien"]
```

Como los dos primeros elementos de la lista (lista) son 1 y "dos", se van a modificar por los elementos que hemos indicado nosotros: 5 y "mil".

También podemos trabajar con indices negativos, muy útiles a la hora de saber nuestro ultimo elemento de la lista, recorrer una lista de atrás hacia adelante, etc. Vamos a ver un ejemplo:

```
nombres = ["Diego", "Carlos", "Martin", "Lorena", "Natalia"]

print nombres[-1] #Nos devuelve el ultimo elemento de nuestra
lista
```

Funciones y métodos para trabajar con listas

La función len() cuenta los elementos de una lista, ejemplo:

```
>>>estudiantes = ["Jose", "Raul", "Marcelo"]
>>>len(estudiantes)
3
```

Añadir elementos a una lista con el método insert(indice, objeto)
El método insert espera un indice y el valor a agregar, ejemplo:

```
>>>estudiantes = ["Jose", "Raul", "Marcelo"]
>>>estudiantes.insert(0, "Maria")
>>>print estudiantes
["Maria", "Jose", Raul", "Marcelo"]
```

Añadir elementos a una lista con el método append(objeto)

```
>>>estudiantes = ["Jose", "Raul", "Marcelo"]
>>>estudiantes.append("Mariela")
>>>print estudiantes
["Jose", "Raul", "Marcelo", "Mariela"]
```

Añadir varios elementos a una lista con el método extend(segmento)

```
>>>estudiantes = ["Jose", "Raul", "Marcelo"]
>>>estudiantes.extend(["Dario", "Natalia"])
>>>print estudiantes
["Jose", "Raul", "Marcelo", "Dario", "Natalia"]
```

Buscar elementos de una lista con el método index(objeto)

```
>>>estudiantes = ["Jose", "Raul", "Marcelo"]
>>>estudiantes.index("Raul")
1 #Retorna el índice del elemento "Raul"
```

Eliminar un elemento de la lista con el método remove(objeto)

```
>>>estudiantes = ["Jose", "Raul", "Marcelo"]
>>>estudiantes.remove("Raul")
>>>print estudiantes
["Jose", "Marcelo"]
```

Con el método split() podemos convertir una cadena de caracteres en una lista delimitada por los índices que indiquemos. Si no se indica nada se utilizan los espacios:

```
>>>"Hola mi nombre es Diego".split()
["Hola","mi","nombre","es","Diego"]

>>>"1-2-3-4-5-6".split("-")
["1","2","3","4","5","6"]
```

Buscar y saber si hay elemento duplicados con el método set()

```
>>>estudiantes = ["Martin", "Jose", "Raul", "Jose"]
>>>estudiantes_unicos = list(set(estudiantes))
>>>print estudiantes_unicos
["Raul", "Jose", "Martin"]
```

Veamos un ejemplo con una función:

```
def duplicado(lista):
    new_list = list(set(lista))

    if len(new_list) != len(lista):
        return True #Retorna True si hay duplicados
    else:
        return False #Retorna False si no hay duplicados
```

Con el método pop(), podremos eliminar y mostrar el ultimo elemento de la lista, ejemplo:

```
>>>estudiantes = ["Jose", "Raul", "Marcelo"]
>>>estudiantes.pop()
"Marcelo"
>>>estudiantes
["Jose", "Raul"]
```

Saber cuantas veces se repite un elemento de una lista con el método count(objeto)

```
>>>estudiantes = ["Jose", "Raul", "Marcelo", "Raul"]
>>>estudiantes.count("Raul")
2
```

Invertir una lista con el método reverse()

```
>>>estudiantes = ["Jose", "Raul", "Marcelo", "Raul"]
>>>estudiantes.reverse()
>>>print estudiantes
["Raul", "Marcelo", "Raul", "Jose"]
```