**Operaciones con listas.**

Tenemos una lista de 5 elementos, podemos referenciar un elemento en concreto de la lista.

>>> Lista[0]  
'a'  
>>>Lista[1]  
'b'  
>>>Lista[4]  
'3'

Referirnos a cada elemento individual de la lista mediante su posición dentro de ella, así el primer elemento de una lista (no vacía) lo tendremos siempre en la posición 0, el segundo en la 1, etc. Como es nuestro caso en una lista de cinco elementos, estos estarán referenciados del 0 al 4.

También podemos usar **índices negativos** para acceder a sus elementos. Si el índice es negativo, este empieza a contar a partir del último elemento. En nuestro ejemplo los elementos irán de -1 a -5, donde -1 nos da el último elemento de la lista y -5 el primero.

>>>Lista[-1]  
'3'  
>>>Lista[-4]  
'b'  
>>>Lista[-5]  
'a'

Para saber cuantos elementos hay en una lista disponemos de la función **len**

>>>len(Lista)  
5

Los elementos no son inmutables, por lo que podemos **cambiar el valor** de los elementos.

>>> la=[1, 2, 3, 4, 5]  
>>> la  
[1, 2, 3, 4, 5]  
>>> la[2] = la[1] + la[3] + 5  
>>> la  
[1, 2, 11, 4, 5]

Podemos obtener un subconjunto de elementos de una lista especificando dos índices (inicio y fin) **slicing**, el resultado será una nueva lista que contendrá los elementos que se encuentren en el rango de los índices.

>>>Lista[2:4]  
['alfa', 'beta']

Obtenemos una nueva lista con los elementos del rango especificado. El rango va desde el elemento 2 (primer índice) hasta el elemento 3 (segundo índice -1). Como observareis el primer índice si que se incluye en el rango pero el segundo se queda fuera [2:4) por lo que solo obtenemos los elementos 2 y 3.

>>>Lista[2:2]  
[]  
>>>Lista[2:3]  
['alfa']

aquí también podemos usar los índices negativos.

>>> Lista[2:-1]  
['alfa', 'beta']

>>> Lista[-3:-1]  
['alfa', 'beta']

Obtendremos los dos primeros elementos de la lista.

>>> Lista[0:2]  
['a', 'b']

Extraemos el primer elemento de la lista.

>>> Lista[1:5]  
['b', 'alfa', 'beta', '3']

Veamos otra forma de realizar esto. **(slice assignament)**

>>> # dos primeros elementos  
... Lista[:2]  
['a', 'b']  
>>> # todos menos el primero  
... Lista[1:]  
['b', 'alfa', 'beta', '3']  
>>> # toda la lista  
... Lista[:]  
['a', 'b', 'alfa', 'beta', '3']

Las listas son dinámicas por lo que podemos introducir nuevos elementos en ellas **append.**

>>> Lista.append("nou")  
>>> Lista  
['a', 'b', 'alfa', 'beta', '3', 'nou']

**append,** añade un único elemento al final de la lista, también podemos insertarlo en una posición en concreto **insert**.

>>> Lista.insert(3,"insert")  
>>> Lista  
['a', 'b', 'alfa', 'insert', 'beta', '3', 'nou']

Tenemos esta otra forma de añadir uno o mas elementos

>>> la=[1,2,3,4,5]  
>>> la[2:2]=['a', 'b']  
>>> la  
[1, 2, 'a', 'b', 3, 4, 5]

Podemos añadir los elementos de otra lista concatenar (**extend**).

>>> Lista.extend(['1','2'])  
>>> Lista  
['a', 'b', 'alfa', 'insert', 'beta', '3', 'nou', '1', '2']

En este último ejemplo tenemos que pararnos a reflexionar un poco, fijaros que lo que hemos hecho ha sido introducir en nuestra lista los elementos que contenía una segunda lista, si queréis ver lo que quiero decir probar a ejecutar la siguiente instrucción.

>>> len(Lista)  
9  
>>> Lista.append(['1','2'])  
>>> Lista  
['a', 'b', 'alfa', 'insert', 'beta', '3', 'nou', '1', '2', ['1', '2']]  
>>> len(Lista)  
10

Fijaros que lo que hemos hecho ha sido añadir un único elemento (de tipo lista) a la lista. Ahora hemos abierto un nuevo frente, todos los elementos de nuestra lista eran del mismo tipo, con la última instrucción hemos añadido un elemento de tipo lista, esto es posible ya que las listas en python pueden contener elementos de diferentes tipos.

>>> Lista=[1, 'a', ['juan', 'pepe'], 'a']  
>>> Lista  
[1, 'a', ['juan', 'pepe'], 'a']  
>>> len(Lista)  
4

Es posible el anidamiento.

>>> l1 = [1, 3, 5]  
>>> l2 = ['a', 'b', **l1**, 'c']  
>>> len(l2)  
4  
>>> l2  
['a', 'b', [1, 3, 5], 'c']  
>>> l2[2][1]  
3  
>>> l2[2].append(7)  
>>> l2  
['a', 'b', [1, 3, 5, 7], 'c']  
>>> l1  
[1, 3, 5, 7]

Fijaros que en la segunda linea, el tercer elemento es una lista, y fijaros que l2[2] se refiere a l1, tal vez sea un poco complicado si no estáis muy duchos, pero creo que vale la pena que perdáis un poco el tiempo en observarlo y comprenderlo.

Podemos borrar un elemento de la Lista (**remove**).

>>> Lista  
[1, 'a', ['juan', 'pepe'], 'a']  
>>> Lista.remove('a')  
>>> Lista  
[1, ['juan', 'pepe'], 'a']

Igual que en la búsqueda, elimina la primera aparición del elemento y si no lo encuentra nos lanzara una excepción.

También podemos borrar un elemento por su índice (**del**).

>>> del Lista[1]  
>>> Lista  
[1, 'a']

otra forma para borrar uno o varios elementos es la siguiente. (eliminamos los dos primeros elementos)

>>> la = [1, 2, 11, 4, 5]  
>>> la   
[1, 2, 11, 4, 5]  
>>> la[0:2]=[]  
>>> la  
[11, 4, 5]

Y por último tenemos **pop**, que nos permite borrar el último elemento de la Lista como si de una cola se tratara. Esta función nos devuelve el último elemento de la lista y lo borra de ella.

>>> Lista  
[1, 'a']  
>>> Lista.pop()  
'a'  
>>> Lista  
[1]

Si a la función **pop** se le añade un índice, nos mostrará y eliminará de la lista el elemento apuntado por el índice.

>>> lst = range(10)  
>>> lst  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
>>> lst.pop(3)  
3  
>>> lst  
[0, 1, 2, 4, 5, 6, 7, 8, 9]

Hasta ahora nos hemos referido a los elementos a partir de su índice, las listas nos permiten hacer la operación inversa, la función **index** a la que pasado un elemento, nos devuelve su índice.

>>> Lista.index(1)  
0  
>>> Lista.index(['juan', 'pepe'])  
2  
>>> Lista.index('a')  
1  
>>> Lista.index('aa')  
Traceback (most recent call last):  
File "", line 1, in   
ValueError: list.index(x): x not in list

Como podéis ver, nos devuelve el índice del primer elemento que coincida, el elemento 'a' lo tenemos en 1 y en 3. Si hacemos una búsqueda de un elemento que no existe, como es en el último caso, el programa nos devolverá una excepción.

Podemos evitar el error anterior si antes de hacer la búsqueda nos aseguramos que el elemento exista, esto lo hacemos mediante **in** que nos devolverá **True** o **False** dependiendo si el elemento existe o no.

>>> 'aa' in Lista  
False  
>>> 1 in Lista  
True

Los valores booleanos True y False, se incorporaron a python a partir de la versión 2.2.1, hasta entonces se establecían una serie de reglas que determinaban cuando algo era cierto o verdadero.

* Cero es falso; el resto de los números son verdaderos.
* Una cadena vacía ("") es falso, cualquier otra cadena es verdadera.
* Una lista vacía ([]) es falso; el resto de las listas son verdaderas.
* Una tupla vacía (()) es falso; el resto de las tuplas son verdaderas.
* Un diccionario vacío ({}) es falso; todos los otros diccionarios son verdaderos.

Esto se sigue manteniendo la diferencia es que ahora disponemos de True (1) y False (0).

También nos puede ayudar la función **count** que nos dirá el número de veces que aparece un elemento en una lista.

>>> lst = [1, 2, 3, 1, 3, 5, 3, 6, 3]  
>>> lst.count(1)  
2  
>>> lst.count(3)  
4  
>>> lst.count(6)  
1  
>>> lst.count(11)  
0

**Operadores**  
Podemos concatenar dos listas. **+**

>>> l1 = [1, 2, 3, 4]  
>>> l2 = ['a', 'b', 'c', 'd']  
>>> l1 + l2  
[1, 2, 3, 4, 'a', 'b', 'c', 'd']  
>>> l2 + l1  
['a', 'b', 'c', 'd', 1, 2, 3, 4]  
>>> l1 + l1  
[1, 2, 3, 4, 1, 2, 3, 4]  
>>> l1 += l2  
>>> l1  
[1, 2, 3, 4, 'a', 'b', 'c', 'd']  
>>> l2  
['a', 'b', 'c', 'd']

El comportamiento de la suma es muy parecido al de **extend**, la diferencia esta en que la suma devuelve una nueva lista.

Con un poco de imaginación podemos modificar este comportamiento.

>>> la = range(1,5)  
>>> la  
[1, 2, 3, 4]  
>>> la += [5]  
>>> la  
[1, 2, 3, 4, 5]

Podemos disponer de un multiplicador **\***

>>> l1=[1, 2]  
>>> l1  
[1, 2]  
>>> l1 \* 3  
[1, 2, 1, 2, 1, 2]  
>>> l1  
[1, 2]  
>>> l1 = l1 \* 2  
>>> l1  
[1, 2, 1, 2]  
>>> l1 \*= 2  
>>> l1  
[1, 2, 1, 2, 1, 2, 1, 2]

El operador **\*** hace de repetidor, concatenando la lista tantas veces como se indique.

La función **sort**, nos permite ordenar los elementos de una lista.

l1 = [3, 5, 1, 3, 4]  
>>> l1.sort()  
>>> l1  
[1, 3, 3, 4, 5]

La función **reverse**, invierte los elementos de la lista

>>> l1 = [1, 2, 3, 4, 5]  
>>> l1.reverse()  
>>> l1  
[5, 4, 3, 2, 1]

Otra función que puede resultar útil es **list**, que permite convertir un string en una lista.

>>> [1, 2] + list("34")  
[1, 2, '3', '4']

Algo parecido podemos obtener con **split.** Que nos devuelve una lista con todas las palabras de una cadena de texto.

>>> nom = "Pedro Ruiz Mayoral"  
>>> nom.split()  
['Pedro', 'Ruiz', 'Mayoral']

Y como no la funcion inversa **join**, que os convierte una lista en una cadena de texto.

>>> l = ['Pedro', 'Ruiz', 'Mayoral']  
>>> l  
['Pedro', 'Ruiz', 'Mayoral']  
>>> ' '.join(l)  
'Pedro Ruiz Mayoral'  
>>> '<->'.join(l)  
'Pedro<->Ruiz<->Mayoral'

Podemos **comparar** dos listas.

>>> [1, 2, 3] == [1, 2, 3]  
True  
>>> [1, 2, 3] == [1, 3, 2]  
False  
>>> a = [1, 2, 3]  
>>> b = [1, 2, 3]  
>>> c = a  
>>> a == b  
True  
>>> c == b  
True

>>> [1, 2, 3] < [1, 2, 3]  
False  
>>> [1, 2, 3] <= [1, 2, 3]  
True  
>>> [1, 2, 3] > [1, 2, 3]  
False  
>>> [1, 2, 3] > [1, 1, 2]  
True  
>>> [1, 2, 3] < [1, 1, 2]  
False  
>>> [1, 2, 3] < [1, 3, 2]  
True

y por ultimo una interacción por la lista con **for**

>>> li = range(9)  
>>> for x in li:  
... print x\*2  
...  
0  
2  
4  
6  
8  
10  
12  
14  
16

un ejemplo más útil, podría ser este.

>>> pagos = [['gasolina', 30], ['cena', 20], ['libros', 12]]  
>>> total = 0  
>>> for pago in pagos:  
... total += pago[1]  
...  
>>> total  
62

**Operaciones con cadenas.**

**Extracción de subcadenas:** las cadenas en Python son almacenadas en un array, así por ejemplo s[0]=”M”. Los índices del array se pueden especificar como un par separado por dos puntos, esto retornará una subcadena desde el primer índice al último. También pueden omitirse uno de los índices, si se omite el primero indica el inicio de la cadena y la omisión del último indica el fin de la cadena. También pueden especificarse índices negativos, donde -1 es el último caracter.

>>> print s[2]  
l  
>>> print s[5:10]  
Mundo  
>>> print s[:4]  
Hola  
>>> print s[8:]  
do  
>>> print s[-5:-2]  
Mun

**int len(string):** retorna la longitud de una cadena.

**int count(sub [,start[,end]]):** retorna el número de ocurrencias de una cadena dentro de otra, por ejemplo:

>>> print s.count(“Hola”)  
1  
>>> print s.count(“o”)  
2

**boolean endswidth(sub [,start[,end]]):** retorna verdadero o falso si una cadena termina con la cadena especificada.

>>> print s.endswith(“do”)  
True  
>>> print s.endswith(“d”)  
False

**boolean startswidth(sub [,start[,end]]):** similar a endswidth.

>>> print s.startswith(“Hola”)  
True  
>>> print s.startswith(“Mundo”)  
False

**int find(sub [,start[,end]]):** retorna la posición numérica de la primera ocurrencia de una cadena dentro de otra.

>>> print s.find(“o”)  
1

**int rfind(sub [,start[,end]]):** similar a find, pero en vez de retornar la posición de la primera ocurrencia, lo hace de la última.

>>> print s.rfind(“o”)  
9

**string lower():** retorna la cadena original en letras minúsculas.

>> print s.lower()  
hola mundo

**string upper():** retorna la cadena original en letras mayúsculas.

>>> print s.upper()  
HOLA MUNDO

**string replace(old, new [,count]):** reemplaza todas o un número dado de ocurrencias de una cadena dentro de otra.

>>> print s.replace(“o”,”O”)  
HOla MundO  
>>> print s.replace(“o”,”O”,1)  
HOla Mundo

**string strip([chars]):** retorna una cadena eliminando al inicio y el fin de la misma un caracter dado. Si no se indica caracter utiliza como predeterminado el espacio en blanco.

>>> print s.strip()  
Hola Mundo  
>>> print s.strip(“H”)  
ola Mundo  
>>> print s.strip(“o”)  
Hola Mund

**array split([sep [,maxsplit]]):** divide una cadena en un array de cadenas dado un separador. Si no se indica separador utiliza como predeterminado el espacio en blanco. Opcionalmente, se puede especificar el número máximo de divisiones.

>>> a = ['abc','def','ghi']  
>>> t = “|”  
>>> print t.join(a)  
abc|def|ghi

**string join(array):** combina los elementos de un array de cadenas con otra cadena.

>>> s.split()  
['Hola', 'Mundo']  
>>> s.split(“o”)  
['H', 'la Mund', '']