Purpose	Syntax	Examples

Access to an element of the list. Negative indices can be used to index the list from the end.
When a wrong index is used, python returns an error.
	list[index]	>>>L= [“abc”, 12345, True]
		>>> L[2]
		True
		>>> L[1:]
		[12345, True]
		>>> L[6]
		Traceback (most recent last):File"<stdin>", line 1, in ?
		IndexError: list index out of range
		>>> L[-1]
		True
		>>> L [-2]
		12345

Assign a value to an element of the list. The contents of a list can be changed by simple assignment using the target index.	**list[index]=value**	>>>L= [“abc”, 12345, True] >>>L[2] = 1 >>> L [“abc”, 12345, 1]

Sum all the element of a list	**sum(l)**	>>> print(sum([1, 2, 3]))
		>>> 6
		>>> print(sum(["aa", "bb", "cc"]))
		"aabbcc"

Minimum of a list	**min(l)**	>>>print(min([1,-2,4,3])) -2

Maximum of a list	**max(l)**	>>>print(max([1,-2,4,3])) 4
Length : number of elements in the list	**len(list)**	>>>L=[4,67,5] >>>print(len(l)) 3

Slicing:
Negative indices can be used in slicing..
An optional third index can be used to specify the increment, which by default is 1.
	list[i :j:k]
	list[i:j]
	list[i:]
	list[:j] list[:]
		>>> x[0:-1]
		[1, 2, 3]
		>>> x = range(10)
		>>> x
		[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
		>>> x[0:6:2]
		[0, 2, 4]

Repetition: duplicate a list a given number of times.	**list*num**	>>>l = [“#”, 2, 0.6] ***3*** >>>[“#”, 2, 0.6, “#”, 2, 0.6, “#”, 2, 0.6] **>>>** m= [[1, 2, 3]] * 3 [[1, 2, 3], [1, 2, 3], [1, 2, 3]] **>>>** m[1][1] = 4 [[1, 4, 3], [1, 4, 3], [1, 4, 3]]
A for statement executes the specified block of code for every element in a list.	**for <x> in <list>**	>>> for e in [5,6,8]: ……. print(e,”***”) 5***6***10*** >>>for x in [1, 2, 3, 4]: print x*2 2,4,6,8

Create a list from a string using a delimiter if no delimiter use space	**str.split(“,”)**	>>>c1=“the kids are playing” >>>print(c1.split()) >>> [“the,”kids”,”are”,”playing”] >>>c2=“the kids, are playing,outside” >>>print(c2.split(‘ ,’)) [“the kids”, “are playing”, “outside”]

List concatenation	**l1+l2**	**>>>** pop = [0, 1] + [1, 0] **>>>** pop [0, 1, 1, 0] **>>>** pop += [2, 3] **>>>** pop [0, 1, 1, 0, 2, 3]

Return the index of the first occurrence of an element in a list. If the element is not present, the method raises the ValueError exception	**l.index(element)**	>>> L=[1,11,111]
		>>>L.index(111)
		2
		>>> L.index(33)
		Traceback (most recent call last):
		File "<stdin>", line 1, in ?
		ValueError: list.index(x): x not in list

Returns True if the element is present in the list and False otherwise.	**<element>in <list>**	>>> H=[1,2,3]
		>>> 34 in H
		False

Remove the first occurrence of an element from a list. If the item provided as a parameter does not exist in the list, the ValueError exception is thrown.	**l.remove(element)**	>>> h=[1,2,3,2,4,5]
		>>>print(h.remove("new"))
		>>>[1,3,2,4,5]]
		>>> h.remove(34)
		Traceback (most recent call last):
		File "<stdin>", line 1, in ?
		ValueError: list.remove(x): x not in list

Delete a part of a list, from index	**del list[i :j]**	**>>>** h=["asd", 12345, 1, 67, 89] **>>>del** hop[1:3] **>>>** hop

i included to *j* excluded		["asd", 67, 89]

The method sort(), sort the elements of the list in the ascending order. The built-in function sorted() returns a new sorted list without modifying the source list .	**l.sort() sorted(l)**	>>>numbers = [17, 38, 10, 25, 72]
		>>>print{numbers.sort())
		[10,17,25,38,72]
		>>> a = ["hello", 1, "world", 45, 2]
		>>> print(a.sort())
		[1, 2, 45, “hello”, “world”]
		>>> a = [[2, 3], [1, 6]]
		>>>print(a.sort())
		[[1, 6], [2, 3]]
		>>> a = [4, 3, 5, 9, 2]
		>>> print(sorted(a))
		[2, 3, 4, 5, 9]
		>>> print(a)
		[4, 3, 5, 9, 2]

Replace the sub-list of L1 from index i to j by the list L2.	**L1[i,j]=L2**	>>>L1=[10,20,30,40,50,60] >>>L1[0:4]=[-1,-2] >>>print(L1) [-1,-2,50,60]
Reverse a list	**l[::-1]**	>>> L=[0,1,2,3] >>>print(L[::-1]) [3, 2, 1, 0]
number of occurrences of a value in the list	**list.count(value)**	>>>l= [3,2,4,2,3] >>>print(l.count(2)) 2

Create a list of pairs from two lists. It is useful when we want to iterate over two lists together.	**zip(l1,l2)**	>>> print(zip(["a", "b", "c"], [1, 2, 3])) [(“a”, 1), (“b”, 2), (“c”, 3)]