List Operators

**Loop through List items**

**For loop**

Print all the items one by one

L1=[500,800,400,100,2,50]  
for i in L1:  
 print(i)   
#500  
#800  
#400  
#100  
#2  
#50

Print all the items by referring to their index. In order to refer the index, you need to use range and len()

L1=[500,800,400,100,2,50]  
for i in range(len(L1)):  
 print(L1[i])

#500  
#800  
#400  
#100  
#2  
#50

A shorthand for loop that will print all items in a list called list comprehensive

L1=[500,800,400,100,2,50]  
[print(i) for i in L1]

#500  
#800  
#400  
#100  
#2  
#50

**While Loop**

Use the len() function to determine the length of the list, then start at 0 and loop your way through the list items by referring to their indexes and increasing the index by 1 after each iteration.

L1=[500,800,400,100,2,50]  
print(len(L1))#6  
i=0  
while i < len(L1):  
 print(L1[i])  
 i+=1

#500  
#800  
#400  
#100  
#2  
#50

**Python List Comprehension** – offers a short syntax to create a new list based on the values of an existing list.

**"Without list comprehension"**  
fruits = ["Apple", "orange", "Grapes", "Cherry"]  
newList = []  
for i in fruits:  
 if "a" not in i:  
 newList.append(i)  
print(newList)  
  
“**with list comprehension"**  
fruits = ["Apple", "orange", "Grapes", "Cherry"]  
newList = [i for i in fruits if "a" not in i]  
print(newList)

The Syntax = [***expression*** for ***item*** in ***iterable*** if ***condition*** == True]

* **Expression** is the current item in the iteration, but it also the outcome., which allows to manipulate the expression before it ends up like a list item in the new list.

Ex-1: Manipulate the expression by converting to upper case.

fruits = ["Apple", "orange", "Grapes", "Cherry"]  
enlist = [i.**upper()** for i in fruits if "a" not in i]  
print(newList) # ['APPLE', 'CHERRY']

Ex-1: Manipulate the expression by setting all values in the new list as “hello”

fruits = ["Apple", "orange", "Grapes", "Cherry"]

newList=["**hello**" for x in fruits]  
print(newList) # ['hello', 'hello', 'hello', 'hello']

* + **Iterable** is any object like list, tuple, set, dictionary etc or range (#).

**Ex-1:** Add items values from 1 to 9

x=[i for i in range(10)]  
print(x) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

**Ex-2**: Accepts only numbers lower than 5

x=[i for i in range(10) if i <5]  
print(x)#[0, 1, 2, 3, 4]

* + **Condition** is like a filter that only accepts the items that valuate to True

**Ex-1:** Only accepts the items that are not apple

fruits = ["apple", "orange", "Grapes", "Cherry"]  
x=[i for i in fruits if i != "apple"]  
print(x) #['orange', 'Grapes', 'Cherry']

**Sort List —**

List objects have a sort() method that will sort the list alphanumerically, ascending, by default.

Ex: Sort the list alphabetically

l1 = ["orange", "mango", "kiwi", "pineapple", "banana"]  
l1.sort()  
print(l1)

**output:**

['banana', 'kiwi', 'mango', 'orange', 'pineapple']

Ex: Sort the list numerically

l1 =[10,5,8,2,4,7,1]  
l1.sort()  
print(l1)

**output:**

[1,2,4,5,7,8,10]

If need to make the list descending, “reverse = True” argument is required.

Ex: Sort the list descending for strings

l1 = ["orange", "mango", "kiwi", "pineapple", "banana"]  
l1.sort(reverse=True)  
print(l1)*#['pineapple', 'orange', 'mango', 'kiwi', 'banana']*

Ex: Sort the list descending for numeric

l1 = [10,5,8,2,4,7,1]  
l1.sort(reverse=True)  
print(l1)*#[10, 8, 7, 5, 4, 2, 1]*

if you want to reverse the order of a list, regardless of the alphabet? The reverse() method reverses the current sorting order of the elements

Ex: Reverse the order of the list items

l1 = [10,5,8,2,4,7,1]  
l1.reverse()  
print(l1)*#[1, 7, 4, 2, 8, 5, 10]*

If need to make the list customise sort, “key = function” argument is required. The function will return a number that will be used to sort the list (the lowest number first)

Ex: Sort the list based on how close the number is to 50:

def func(n):  
 return abs(n + 50)  
  
x = [100, 50, 65, 82, 23]  
x.sort(key = func)  
print(x)*#23,50,65,82,100  
  
#100+50=150 ->5  
#50+50=100 ->2  
#65+50=115 ->3  
#82+50=132 ->4  
#50-23=73 ->1*

By default, the sort() method is case sensitive, resulting in all capital letters being sorted before lower case letters. Therefore, “key= str.lower” is required to overcome that situation.

Ex: Perform a case-insensitive sort of the list:

list = ["banana", "Orange", "Kiwi", "cherry"]  
list.sort(key=str.lower)  
print(list)*#['banana', 'cherry', 'Kiwi', 'Orange']*

**Copy List—**

You cannot copy a list simply by typing **list2 = list1**, because: **list2** will only be a reference to **list1**, and changes made in **list1** will automatically also be made in **list2**.

There are ways to make a copy, one way is to use the built-in List method **copy()** and **list()**

Ex: Make a copy of a list with the copy() method

l1=[1,5,6]  
l2=l1.copy()  
print(l2) # [1, 5, 6]

Ex: Make a copy of a list with the list() method

l1=[1,5,6]  
l2=list(l1)  
print(l2) # [1, 5, 6]

**Join List**

There are few ways to join or concatenate two lists to one. Such as using + operator, append(),and extend().

Ex: using + operator

L1=[20,15]  
L2=[35,55]  
L3=L1+L2  
print(L3) # [20, 15, 35, 55]

Ex: using append() method

L1=[20,15]  
L2=[35,55]  
for x in L2:  
 L1.append(x)  
print(L1) # [20, 15, 35, 55]

Ex: using extend() method

L1=[20,15]  
L2=[35,55]  
L1.extend(L2)  
print(L1) # [20, 15, 35, 55]

**List Method**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list | my\_list=["one","two"] my\_list.append("three") print(my\_list) *#['one', 'two', 'three']* |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list | my\_list=["one","two"] my\_list.clear() print(my\_list)*#[]* |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list | my\_list=["one","two"] copy\_list=my\_list.copy() print(copy\_list) *#['one', 'two']* |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of times the element appear in the list | my\_list=[1,6,5,6] x=my\_list.count(6) print(x) *#2* |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Add the elements of a list (or any iterable), to the end of the current list | my\_list=[1,6,5,6] other\_list=[100,50] my\_list.extend(other\_list) print(my\_list) *#[1, 6, 5, 6, 100, 50]* |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | returns the position at the first occurrence of the specified value | my\_list=[1,6,5,6] print(my\_list.index(6)) *#1* |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position | my\_list=[1,6,5,6] my\_list.insert(2,1) print(my\_list) *#[1, 6, 1, 5, 6]* |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position | my\_list=[1,6,5,6] my\_list.pop(1) print(my\_list) *#[1, 5, 6]* |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Removes the item with the specified value | my\_list=[1,6,5,6] my\_list.remove(5) print(my\_list) *#[1, 6, 6]* |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list | my\_list=[1,6,5,6] my\_list.reverse() print(my\_list) *#[6, 5, 6, 1]* |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list | my\_list=[1,6,5,6] my\_list.sort() print(my\_list) *#[1, 5, 6, 6]* |
| Min() | Smallest element in the list | my\_list=[1,6,5,6] print(min(my\_list)) *#1* |
| Max() | Largest element in the list | my\_list=[1,6,5,6] print(max(my\_list)) *#6* |
| Sum() | Sum of all numbers of the list | my\_list=[1,6,5,6] print(sum(my\_list)) *#18* |
| \* | Replicate the elements in the list specified number of times | my\_list=[1,6,5,6] x=my\_list\*2 print(x) *#[1, 6, 5, 6, 1, 6, 5, 6]* |