Python List:

A **list** is a data structure in **Python** that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a **list** is called an item. Just as strings are defined as characters between quotes, **lists** are defined by having values between square brackets [ ]

Look at a simple list that stores several names of dogs' breeds:

```
dog_breeds = ['corgi', 'labrador', 'poodle', 'jack russel']
```

```
print(dog_breeds)  # ['corgi', 'labrador', 'poodle', 'jack russel']
```

In the first line, we use square brackets to create a list that contains four elements and then assign it to the `dog_breeds` variable. In the second line, the list is printed through the variable's name. All the elements are printed in the same order as they were stored in the list because lists are **ordered**.

Here is another list that contains five integers:

```
numbers = [1, 2, 3, 4, 5]
```

```
print(numbers)  # [1, 2, 3, 4, 5]
```

Lists can store **duplicate values** as many times as needed.

```
on_off_list = ['on', 'off', 'on', 'off', 'on']
```

```
print(on_off_list)  # ['on', 'off', 'on', 'off', 'on']
```

we note that lists are:

- **ordered**, i.e. each element has a fixed position in a list;
- **iterable**, i.e. you can get their elements one by one;
- able to store **duplicate values;**
- able to store **different types of elements**.

Accessing Values in List

1. Indexing : get the single element (Get the item at position)

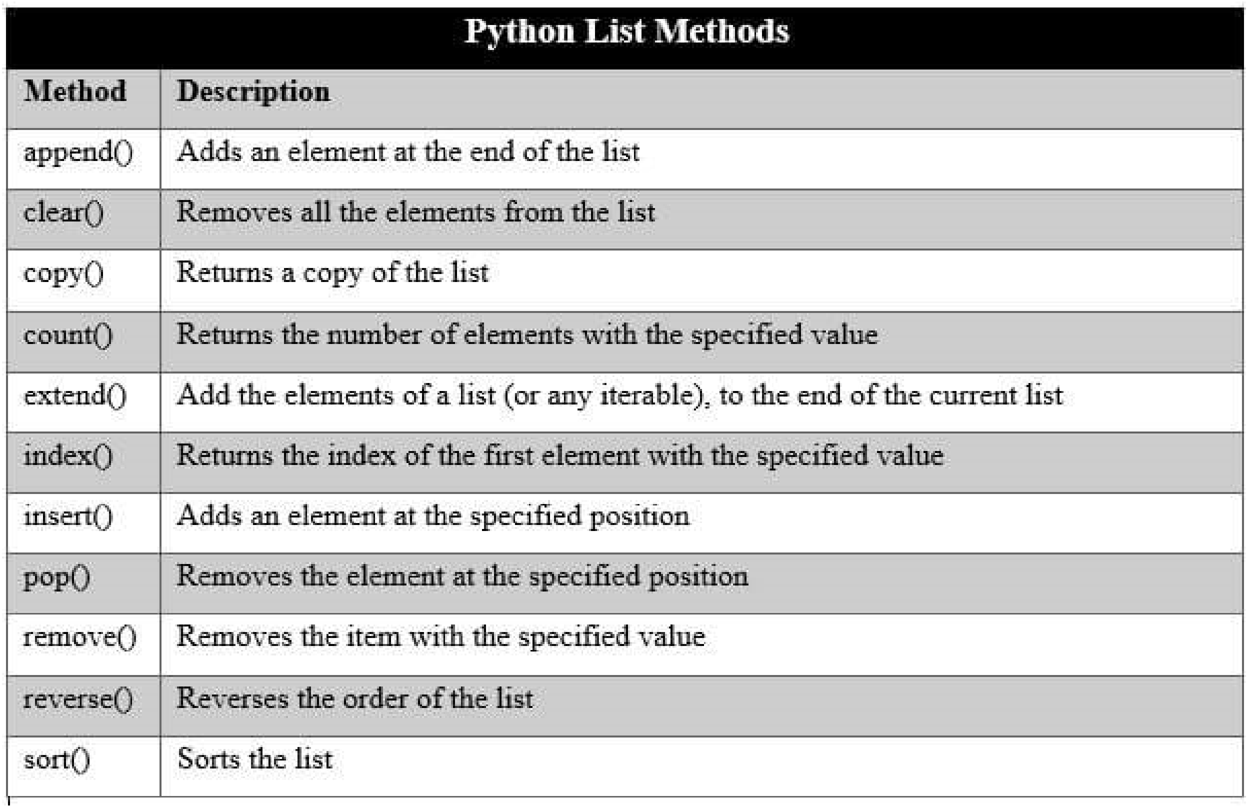
2. Slicing : get the sublist (subsequence from the sequential quantity) You can return a range of items by using the slice syntax.

```
st = [1, 3, 43, 4]

d = st[0]

print(d)  # output is 1
```

```
st = [1, 4, 5, 32, 5]

d = st[0:3]

print(d)  # output is [1, 4, 5]
```

```
# initialize the list (empty list)
# The square brackets and the list function can also be used to create
# empty lists that do not have elements at all.

ls = []
ls = list()
# output []

# initialize a list with single integer value
ls = [10]
#  output [10]

st = 'hello'
ls = list(st)
print(st)
print(ls)
# output hello
# output ['h', 'e', 'l', 'l', 'o']

# indexing
# get the element at given position
ls = [1, 3, 65, 76, 'hello']
d = ls[0]
print(d)
# output 1
# slicing
# get the subsequence(sub-list) at given range
ls = [1, 3, 65, 76, 'hello']
d = ls[0:4]
print(d)
# output [1, 3, 65, 76]
# why list is mutable

K = [2, 43, 1, 78]

K[-1] = 100  # item assignment allowed (changeable at run time ) update item
print(K) # output is  [2, 43, 1, 100]

k.append(29)   # add or append items at run time
print(K)   # output is  [2, 43, 1, 100, 29]

del k[0]    # delete items at run time without changing object id
print(K)   # output is  [43, 1, 100, 29]

k.remove(1) # remove item from starting
print(K)   # output is  [43, 100, 29]

k.pop()   # last element removed
print(K)   # output is  [43, 100]
```

list Methods



Python List Methods	
Method	**Description**
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

List Comprehensions

List comprehensions provide a concise way to create lists. List comprehension is a complete substitute for the lambda function as well as the functions map(), filter() and reduce()

Syntax

The list comprehension starts with a '[' and ']', to help you remember that the result is going to be a list**.**

The basic syntax is

[expression for item in list if conditional]

This is equivalent to:

```
for item in list:
    if conditional:
        expression
```

```
L = [2, 43, 5, 453,  5]

Out = [i+10 for i in L]   # list comprehension add 10 to each elements

print(Out)  # [12, 53, 15, 463, 15 ]
```

List Questions

1. Create a program that will keep track of items for a shopping list. The program should keep asking for new items until nothing is entered (no input followed by enter/return key). The program should then display the full shopping list.
2. Write a program that will store the schedule for a given day for a particular TV station. The program should ask you for the name of the station and the day of the week before asking you for the name of each show and the start and stop times. Once the schedule is complete it should be displayed as a table
3. WAPP to multiply two matrix supplied by user in row-major representation
4. Convert a string into characters
5. Sort the list by the length of string elements
6. Sort the list in reverse order
7. Given a non-empty array, return true if there is a place to split the array so that the sum of the numbers on one side is equal to the sum of the numbers on the other side.

canBalance([1, 1, 1, 2, 1]) → true
canBalance([2, 1, 1, 2, 1]) → false
canBalance([10, 10]) → true

8. Given two arrays of ints sorted in increasing order, outer and inner, return true if all of the numbers in inner appear in outer. The best solution makes only a single "linear" pass of both arrays, taking advantage of the fact that both arrays are already in sorted order.
   linearIn([1, 2, 4, 6], [2, 4]) → true
   linearIn([1, 2, 4, 6], [2, 3, 4]) → false
   linearIn([1, 2, 4, 4, 6], [2, 4]) → true

9. Write a Python function that takes a list and returns a new list with unique elements of the first list.

   Sample List: [1,2,3,3,3,3,4,5]
   Unique List: [1, 2, 3, 4, 5]

10. Write a Python program to print the even numbers from a given list.

    Sample List:[1, 2, 3, 4, 5, 6, 7, 8, 9]
    Expected Result: [2, 4, 6, 8]