List:

The most versatile data structure in Python, a list can contain a sequence of data which may be heterogeneous or homogeneous. To access an element in the list, one needs to use square brackets and specify which index they want to access. Index numbers start from 0 and span till n-1, where n is the size of the list. Negative indexes signify an index from the end, i.e. an index -2 signifies the second last element in the list. Python does not have built-in support for Arrays, but Python Lists can be used instead.

```
my_list=[1,"hello",3,4,5]
```

We can print elements in the list in many ways

```
for item in my_list:
      print(item,end=" ") #1 hello 3 4 5
for i in range(0,len(my_list)):
      print(my_list[i],end=" ") #1 hello 3 4 5
```

You can use square brackets to access individual elements in list.

```
my_list[0]=-1
print(my_list) #[-1,"hello",3,4,5]
```

You can use the append() or insert() or extend() function to add elements to the list.

```
A=[1,2,3]
A.insert(0,0)
A.append(100)
B=[4,5,6]
A.extend(B)
```

Also you can use remove or pop keyword to delete a elements of list.

```
A=[1,2,3]
A.remove(1)
A.pop()
A.pop(0)
```

The clear function is used to clear the functions of all elements.

```
A=[1,2,3,4]
A.clear()
B=A.copy()
```

The copy function can be used to copy a list.

Indexing and slicing in the list is the same as that of string in python.

```
print(my_list[0:2]) #["hello",3]
print(my_list[-2]) #4
print(my_list[-1]) #5
```

Below are few functions taking Iterable objects like list,tupes,sets,dictionaries as

```
print(len(my_list));
print(max(my_list));#error you cannot compare string and int data
type
my_list[1]=1.1
print(max(my_list));#5 can also use min()
```

parameter.

Below are few functions on list object.

```
my_list.append(10)#["hello",3,4,5,10]
print(my_list.count("hello"))#1
my_list.extend([1,2,3])#['hello', 3, 4, 5, 10, 1, 2, 3]
my_list.insert(0,100)#[100, 'hello', 3, 4, 5, 10, 1, 2, 3]
my_list.pop(0)#['hello', 3, 4, 5, 10, 1, 2, 3]
my_list.remove("hello")#[3, 4, 5, 10, 1, 2, 3]
my_list.clear()#[ ]
```

Note the difference between del my_list and my_list.clear() ,after 1st one you will not be able to access my_list variable , print(my_list) will throw error , but second just empties list.

We can use + to concatenate list and * to repeat

```
print(my_list+[100,200])#[100, 200]
print([100,200]*3)#[100, 200, 100, 200, 100, 200]
```

]Also, there is a useful way to create a list using for loops called list

```
new_list=[x*x for x in range(10)]#[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

comprehension