Assignment List

List : List is a mutable data structure

NAME : Lovenish Gaur

Python List Methods

Methods that are available with list object

Are accessed as list.method().

'''

1. append() - Add an element to the end of the list
2. extend() - Add all elements of a list to the another list
3. insert()- I nsert an item at the defined index
4. remove() - Removes an item from the list
5. pop() - Removes and returns an element at the given index
6. clear() - Removes all items from the list
7. index() - Returns the index of the first matched item
8. count() - Returns the count of number of items passed as an argument
9. sort() - Sort items in a list in ascending order
10. reverse() - Reverse the order of items in the list
11. copy() - Returns a shallow copy of the list '''

'' Built-in Functions with List

1. all() Return True if all elements of the list are true (or if the list is empty).
2. any() Return True if any element of the list is true. If the list is empty, return False.
3. enumerate() Return an enumerate object. It contains the index and value of all the items of list as a tuple.
4. len() Return the length (the number of items) in the list.
5. list() Convert an iterable (tuple, string, set, dictionary) to a list.
6. max() Return the largest item in the list.
7. min() Return the smallest item in the list
8. sorted() Return a new sorted list (does not sort the list itself).
9. sum() Return the sum of all elements in the list.

In [1]:

```
ID = [1,2,3,4,5,6,7,8,9,10] # integer
ID
```

Out[1]:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [2]:

```
Furniture = ["Table", "Chair","Mirror","Sofa"] # Strings
Furniture
```

Out[2]:

```
['Table', 'Chair', 'Mirror', 'Sofa']
```

In [3]:

```
Tax = [2.3,3.3,4.5,4.3,3.4] # Floats
Tax
```

Out[3]:

```
[2.3, 3.3, 4.5, 4.3, 3.4]
```

In [4]:

```
Mixed = [1, "Mohit", 2 , "Gilberto", 3.3, "Pena", 4.4 ,"Pablo"]
Mixed
```

Out[4]:

```
[1, 'Mohit', 2, 'Gilberto', 3.3, 'Pena', 4.4, 'Pablo']
```

In [5]:

```
#Append : Add an element to the end of the list

Furniture.append("Stand") # adding single value
Furniture
```

Out[5]:

```
['Table', 'Chair', 'Mirror', 'Sofa', 'Stand']
```

In [6]:

```
#Append : Add an element to the end of the list

Tax.append(3.5)
Tax
```

Out[6]:

```
[2.3, 3.3, 4.5, 4.3, 3.4, 3.5]
```

In [7]:

```
#Extend : Add all elements of a list to the another list
Furniture.extend(["Stool","Bed","Lamp","Mat"])
Furniture
```

Out[7]:

```
['Table', 'Chair', 'Mirror', 'Sofa', 'Stand', 'Stool', 'Bed', 'Lamp',
'Mat']
```

In [8]:

```
#Extend : Add all elements of a list to the another list
Tax.extend([4.5,5.6,6.7,7.8])
Tax
```

Out[8]:

```
[2.3, 3.3, 4.5, 4.3, 3.4, 3.5, 4.5, 5.6, 6.7, 7.8]
```

In [9]:

```
#insert()- I nsert an item at the defined index
Furniture.insert(3,"Speaker")
Furniture
```

Out[9]:

```
['Table',
 'Chair',
 'Mirror',
 'Speaker',
 'Sofa',
 'Stand',
 'Stool',
 'Bed',
 'Lamp',
 'Mat']
```

In [10]:

```
#insert()- I nsert an item at the defined index
Tax.insert(5,9.0)
```

In [11]:

```
Tax
```

Out[11]:

```
[2.3, 3.3, 4.5, 4.3, 3.4, 9.0, 3.5, 4.5, 5.6, 6.7, 7.8]
```

In [12]:

```
#remove() - Removes an item from the list
Furniture.remove("Speaker")
Furniture
```

Out[12]:

```
['Table', 'Chair', 'Mirror', 'Sofa', 'Stand', 'Stool', 'Bed', 'Lamp',
'Mat']
```

In [13]:

```
#remove() - Removes an item from the list
Tax.remove(9.0)
Tax
```

Out[13]:

```
[2.3, 3.3, 4.5, 4.3, 3.4, 3.5, 4.5, 5.6, 6.7, 7.8]
```

In [14]:

```
len(Furniture)
```

Out[14]:

```
9
```

In [15]:

```
len(Tax)
```

Out[15]:

```
10
```

In [16]:

```
#pop() - Removes and returns an element at the given index
Furniture.pop(8)
Furniture
```

Out[16]:

```
['Table', 'Chair', 'Mirror', 'Sofa', 'Stand', 'Stool', 'Bed', 'Lamp']
```

In [17]:

```
Tax
```

Out[17]:

```
[2.3, 3.3, 4.5, 4.3, 3.4, 3.5, 4.5, 5.6, 6.7, 7.8]
```

In [18]:

```
#pop() - Removes and returns an element at the given index
Tax.pop(7)
Tax
```

Out[18]:

```
[2.3, 3.3, 4.5, 4.3, 3.4, 3.5, 4.5, 6.7, 7.8]
```

In [19]:

```
#index() - Returns the index of the first matched item
Furniture.index("Bed")
```

Out[19]:

```
6
```

In [20]:

```
#index() - Returns the index of the first matched item
Tax.index(3.4)
```

Out[20]:

```
4
```

In [21]:

```
#count() - Returns the count of number of items passed as an argument
Furniture.count("Bed")
```

Out[21]:

```
1
```

In [22]:

```
#count() - Returns the count of number of items passed as an argument
Tax.count(3.4)
```

Out[22]:

```
1
```

In [23]:

```
#sort() - Sort items in a list in ascending order
Furniture.sort()
Furniture
```

Out[23]:

```
['Bed', 'Chair', 'Lamp', 'Mirror', 'Sofa', 'Stand', 'Stool', 'Table']
```

In [24]:

```
#sort() - Sort items in a list in ascending order
Tax.sort()
Tax
```

Out[24]:

```
[2.3, 3.3, 3.4, 3.5, 4.3, 4.5, 4.5, 6.7, 7.8]
```

In [25]:

```
#reverse() - Reverse the order of items in the list
Furniture.reverse()
Furniture
```

Out[25]:

```
['Table', 'Stool', 'Stand', 'Sofa', 'Mirror', 'Lamp', 'Chair', 'Bed']
```

In [26]:

```
#reverse() - Reverse the order of items in the list
Tax.reverse()
Tax
```

Out[26]:

```
[7.8, 6.7, 4.5, 4.5, 4.3, 3.5, 3.4, 3.3, 2.3]
```

In [27]:

```
#copy() - Returns a shallow copy of the list
Furniture.copy()
```

Out[27]:

```
['Table', 'Stool', 'Stand', 'Sofa', 'Mirror', 'Lamp', 'Chair', 'Bed']
```

In [28]:

```
#copy() - Returns a shallow copy of the list '''
Tax.copy()
```

Out[28]:

```
[7.8, 6.7, 4.5, 4.5, 4.3, 3.5, 3.4, 3.3, 2.3]
```

In [29]:

```
#clear() - Removes all items from the list
Furniture.clear()
Furniture
```

Out[29]:

```
[]
```

In [30]:

```
#clear() - Removes all items from the list
Tax.clear()
Tax
```

Out[30]:

```
[]
```

'' Built-in Functions with List

1. all() Return True if all elements of the list are true (or if the list is empty).
2. any() Return True if any element of the list is true. If the list is empty, return False.
3. enumerate() Return an enumerate object. It contains the index and value of all the items of list as a tuple.
4. len() Return the length (the number of items) in the list.
5. list() Convert an iterable (tuple, string, set, dictionary) to a list.
6. max() Return the largest item in the list.
7. min() Return the smallest item in the list
8. sorted() Return a new sorted list (does not sort the list itself).
9. sum() Return the sum of all elements in the list.

In [31]:

```
#all() Return True if all elements of the list are true (or if the list is empty).
```

In [32]:

```
List = ["",1,"Moon",3.4,""]
Furniture = ["Table", "Chair","Mirror","Sofa"]
Tax = [2.3,3.3,4.5,4.3,3.4]
ID = [1,2,3,4,5,6,7,8,9,10]
List
```

Out[32]:

```
['', 1, 'Moon', 3.4, '']
```

In [33]:

```
all(List)
```

Out[33]:

```
False
```

In [34]:

```
#any() Return True if any element of the list is true. If the list is empty, return
any(List)
```

Out[34]:

```
True
```

In [35]:

```
#enumerate() Return an enumerate object. It contains the index and value of all the
list(enumerate(List))
```

Out[35]:

```
[(0, ''), (1, 1), (2, 'Moon'), (3, 3.4), (4, '')]
```

In [36]:

```
#len() Return the length (the number of items) in the list.
len(List)
```

Out[36]:

```
5
```

In [37]:

```
#list() Convert an iterable (tuple, string, set, dictionary) to a list.
list(List)
```

Out[37]:

```
['', 1, 'Moon', 3.4, '']
```

In [38]:

```
#max() Return the largest item in the list.
max(ID)
```

Out[38]:

```
10
```

In [39]:

```
max(Furniture)
```

Out[39]:

```
'Table'
```

In [40]:

```
max(Tax)
```

Out[40]:

```
4.5
```

In [41]:

```
#min() Return the smallest item in the list
min(ID)
```

Out[41]:

```
1
```

In [42]:

```
min(Furniture)
```

Out[42]:

```
'Chair'
```

In [43]:

```
min(Tax)
```

Out[43]:

```
2.3
```

In [44]:

```
#sorted() Return a new sorted list (does not sort the list itself).
sorted(ID)
```

Out[44]:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [45]:

```
sorted(Furniture)
```

Out[45]:

```
['Chair', 'Mirror', 'Sofa', 'Table']
```

In [46]:

```
sorted(Tax)
```

Out[46]:

```
[2.3, 3.3, 3.4, 4.3, 4.5]
```

In [47]:

```
#sum() Return the sum of all elements in the list.
sum(ID)
```

Out[47]:

```
55
```

In [48]:

```
sum(Tax)
```

Out[48]:

```
17.799999999999997
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```