General Builtin Functions of List()

In [ ]:
```
#Builtin Functions related to List
len()-->return the number of elements inside the list
```

In [2]:
```
#Example:
x=[10,50.45,"The python concept" ,20,30,40,50,60]
len(x)
```
Out[2]:
```
8
```

In [4]:
```
#Traversal means visit each and every element of the list
x=[10,20,40,50,60]
for i in range(len(x)):
    print(x[i],end=" ")
```
```
10 20 40 50 60
```

In [10]:
```
#Traversal means visit each and every element of the list
x = [10,20,30,40,50,60]
for j in x:
    print(j,end=" ")
```
```
10 20 30 40 50 60
```

In [8]:
```
#count()--> return the number of occurances of a specified element of a list
x=[10,20,30,10,20,30]
x.count(10)
```
Out[8]:
```
2
```

In [9]:
```
#count()--> return the number of occurances of a specified element of a list
x=[10,20,30,10,20,30]
x.count(100)
```
Out[9]:
```
0
```

In [11]:
```
#index -->return the index of first occurance
x=[10,20,30,40,50]
x.index(30)
```
Out[11]:
```
2
```

In [15]:
```
#index -->return the index of first occurance
x=[10,20,20,50,30,40,50,50,50,50,30,40,50]
x.index(50)
```
Out[15]:
```
3
```

In [16]:
```
#index -->return the index of first occurance
x=[10,20,20,50,30,40,50,50,50,50,30,40,50]
x.index(500)
#Note--> If element is not present inside the given list then you will get an error
```
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_12252/697023210.py in <module>
      1 #index -->return the index of first occurance
      2 x=[10,20,20,50,30,40,50,50,50,50,30,40,50]
----> 3 x.index(500)
      4 #Note--> If element is not present inside the given list then you will get an error

ValueError: 500 is not in list
```

Insertion Functions of List()

In [22]:
```
#Insertion of elements inside th list.
#Append() --> Add the element at the last.
x=[10,20,30,40]
x.append("The python coding")
x.append(500.5)
x
```
Out[22]:
```
[10, 20, 30, 40, 'The python coding', 500.5]
```

In [53]:
```
#Another way to add any element in list --> it will replace the element
x=[10,20,30,40,50]
x[3]=200
x
```
Out[53]:
```
[10, 20, 30, 200, 50]
```

In [54]:
```
#Another way to add any element in list --> it will replace the element
x=[10,20,30,40,50]
x[500]=200
x
```
```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_12252/186560405.py in <module>
      1 #Another way to add any element in list --> it will replace the element
      2 x=[10,20,30,40,50]
----> 3 x[500]=200
      4 x

IndexError: list assignment index out of range
```

In [24]:
```
#Insertion of elements inside th list.
#insert() --> Add the element at the given index position
#Syantax of insert function:  list_name.insert(index_value,element_value)
x=[10,20,30,40]
x.insert(2,"The python coding")
x
```
Out[24]:
```
[10, 20, 'The python coding', 30, 40]
```

In [25]:
```
x=[10,20,30,40]
x.insert(10,700)   #if the index value is not present inside the list then PVM will add the element
#at last
x
```
Out[25]:
```
[10, 20, 30, 40, 700]
```

In [28]:
```
x=[10,20,30,40]
x.insert(-10,200)   #if the index value is not present inside the list then PVM will add the element
#at last
x
```
Out[28]:
```
[200, 10, 20, 30, 40]
```

In [ ]:
```
#Append vs Insert
append function will add elements at the last position
insert function will add the element at the given index or posisiton
Insert function will take argument like index value and element.
append function will take one argument
```

In [33]:
```
#Extend Function --> it is used to add one list inside another list
#Syntax--> first_list.extend(second_list)
#all items or elements that are present inside the second list will be added into the first list
x=[10,20,30,40,50]

x.extend([70,80,90])
x
```
Out[33]:
```
[10, 20, 30, 40, 50, 70, 80, 90]
```

Deletion functions inside the list

In [36]:
```
#remove() --> it will simply remove the element from the list
x=[10,20,20,30,40,50]
x.remove(20)
x
```
Out[36]:
```
[10, 20, 30, 40, 50]
```

In [37]:
```
#remove() --> it will simply remove the givem element from the list
x=[10,20,30,40,50]
x.remove(1000)
x
#Error
```
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_12252/213226116.py in <module>
      1 #remove() --> it will simply remove the element from the list
      2 x=[10,20,30,40,50]
----> 3 x.remove(1000)
      4 x
      5 #Error

ValueError: list.remove(x): x not in list
```

In [39]:
```
#pop()--> simply delete the element from the last position. and also return the deleted value
x=[10,20,30,40,50]
x.pop()  #50
x # 10,20,30,40
```
Out[39]:
```
[10, 20, 30, 40]
```

In [40]:
```
x=[]
x.pop()
x
```
```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_12252/1129598182.py in <module>
      1 x=[]
----> 2 x.pop()
      3 x

IndexError: pop from empty list
```

In [41]:
```
#Variation of pop function --> if we want to deleted any element at the given index in that case
#we can also use pop function
#Syntax: list_name.pop(index_value)

x=[10,20,30,40,50]
print(x.pop(2)) #30
x
```
```
30
```
Out[41]:
```
[10, 20, 40, 50]
```

In [42]:
```
x=[10,20,30,40,50]
print(x.pop(100)) #30
x
```
```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_12252/1559382746.py in <module>
      1 x=[10,20,30,40,50]
----> 2 print(x.pop(100)) #30
      3 x

IndexError: pop index out of range
```

General Builtin Functions

In [43]:
```
#reverse() --> it will simply reverse the list
x=[10,20,30,40,50]
x.reverse()
x
```
Out[43]:
```
[50, 40, 30, 20, 10]
```

In [45]:
```
#sorted --> it will simply sort the string in ascending order
x=[100,20,30,40,50]
y=sorted(x)
y
```
Out[45]:
```
[20, 30, 40, 50, 100]
```

Airthemtic Operators that we will use in list

In [49]:
```
# + operator -->Concatenation operator (adding two list)
x=[10,20,30]
y=[50,60,70,70,80]
z=[100,200,400]
x+y+z
#Note: When we are performing concatenation operator then we should take care that both
#the operands are of list type only if you are using any other type then it will give you an error
```
Out[49]:
```
[10, 20, 30, 50, 60, 70, 70, 80, 100, 200, 400]
```

In [51]:
```
#  * operator  --> Repetition Opertaor (Repeat the list numebr of times)
x=[10,20,30,50]
y=x*4
y
#Note: One oprand should be of list type and second should of int type
```
Out[51]:
```
[10, 20, 30, 50, 10, 20, 30, 50, 10, 20, 30, 50, 10, 20, 30, 50]
```

In [52]:
```
#Membership --> in and not in
x =[10,20,30,40,50]
print(10 in x)
print(200 in x)
```
```
True
False
```

list vs mutability

In [55]:
```
#Once we create a list object then we can modify the content of it. list are mutable(changeable)
x=[10,20,30,50,60]
print(id(x))
x[3]=3000
print(id(x))
```
```
1935627772032
1935627772032
```

Tuple

In [ ]:
```
-->Tuple is exactly same as list but there is only one difference betwen list and tuple and that
difference is list are mutable (changeable) and tuples are immutable.
--> Element of the tuple are fixed that means you cannot modify any element in tuple.
--> readonly version of list
---> Indexing is important in case of tuple(indexing and slicing)
--> Duplicates are allow in case of typle
--> disimilar element you can use incase of tuple
--> Tuples are define inside the parenthesis
```

In [56]:
```
#Creation of Tuple Object
x=(10,20,40,50,70)
print(type(x))
```
```
<class 'tuple'>
```

In [58]:
```
#Creation of Tuple Object
x=10,20,40,50,70  #--> tuple packing
print(type(x))
```
```
<class 'tuple'>
```

In [57]:
```
t=()
print(type(t))
```
```
<class 'tuple'>
```

In [59]:
```
#We cannot perform insertion , deletion and modification incase of tuple that means it is immutable
```

In [61]:
```
#Acces the element of the tuple
#1.Indexing --> return element at the given index
#2.Slicing --> return a piece of list
x=(10,20,30,40,50)
print(x[1])  #20
#print(x[5])    #error
#print(x[80])       #error
print(x[::])  #all
print(x[2:9])   #30,40,50,
print(x[::-1])  #revser
print(x[::-2])     #50,30,10
print(x[10::9])
```
```
20
(10, 20, 30, 40, 50)
(30, 40, 50)
(50, 40, 30, 20, 10)
(50, 30, 10)
()
```

Tuple vs Immutability

In [65]:
```
#Once tuple is created then you can not do any modification on that object(immutable)
y=(10,20,30,40,50)

y[1]=5000
y
```
Out[65]:
```
[10, 5000, 30, 40, 50]
```

Airthemtic Operators that we will use in Tuple()

In [67]:
```
# + -->concatenation operator (add two tuples)
x=(10,20,30)
y=(50,60,60)
t1=x+y
t1
```
Out[67]:
```
(10, 20, 30, 50, 60, 60)
```

In [68]:
```
# * --> repetition operator
x=(10,20,40)
y=x*3
y
```
Out[68]:
```
(10, 20, 40, 10, 20, 40, 10, 20, 40)
```

In [69]:
```
#membership operator in and not in
x=(10,20,30)
10 in x
```
Out[69]:
```
True
```

Builtin Functions related to List

len()-->return the number of elements inside the list

In [70]:
```
x=(10,20,40,50)
len(x)
```
Out[70]:
```
4
```

In [71]:
```
#count()--> return the number of occurances of a specified element of a tuple
x=(10,20,30,10,20,30)
x.count(10)
```
Out[71]:
```
2
```

In [72]:
```
#index -->return the index of first occurance
x=(10,20,30,40,50)
x.index(30)
```
Out[72]:
```
2
```

In [81]:
```
#sorted --> it will simply sort the string in ascending order
x=(100,20,30,40,50)
y = sorted(x) #-->
y
```
Out[81]:
```
[20, 30, 40, 50, 100]
```

In [82]:
```
#min or max
#min --> return minimum value  of the list/tuple
#max--> return the maximum value of tuple/list
t=(10,20,40,50,60)
print(min(t))
print(max(t))
```
```
10
60
```

In [ ]: