Builtin Functions in List

len() Function

In [ ]:
```
len function ---> len function is used to return the length of  a given sequence.(list,set,dictionary,
                                                                  tuple,string)
```

Examples of len Functions

In [11]:
```
list = [10,20,30,40,50]
len(list)
```
Out[11]:
```
5
```

In [12]:
```
list = []
len(list)
```
Out[12]:
```
0
```

Count() Function

In [ ]:
```
count function  --> Count function is basically used to return the number of occurances of an element.

Syntax:
    list_name.count(element)
```

Examples of Count Function

In [13]:
```
x=[10,20,30,40,50,10,20,30,10,20]
x.count(10)
```
Out[13]:
```
3
```

In [3]:
```
x=[10,20,30,40,50,10,20,30,10,20]
x.count(100)
Note --> if the element is not present in the list then count function will give you 0.
```
Out[3]:
```
0
```

Index Function

In [ ]:
```
index function --> Index function is used to get the index value of the first occurance of an element
                   if element is not present then index function will give you an error.

Syntax:
    list_name.index(element_value)
```

Examples of Index Function

In [5]:
```
x=[10,20,30,40,50]
x.index(50)
```
Out[5]:
```
4
```

In [7]:
```
x=[10,20,30,40,50,10,20,30,40,50,10,20,30,40,50]
x.index(50)
```
Out[7]:
```
4
```

Insertion Functions of List

In [ ]:
```
In Python if you want to insert any element in the list then we have 3 Types of insertion function:
    1.append function
    2.Insert Function
    3.extend function
```

Append Function

In [ ]:
```
append () Function --> append function will add any element at the end of the list.

Syntax:
    list_name.append(value/data)

Note: Append function will add a single element  at a time.
```

Examples of Append Function

In [15]:
```
list=[10,20]
print(list)
list.append("Hello")
print(list)
list.append("World")
print(list)
```
```
[10, 20]
[10, 20, 'Hello']
[10, 20, 'Hello', 'World']
```

Insert Function

In [ ]:
```
insert () Function --> insert function is to add a given element on the given index

Syntax:
    list_name.insert(index_value ,value/data)

Note: Insert function will take two arguments( indedx_value,data)
```

Examples of Insert Function

In [2]:
```
x=[10,20,30,40,50]
x.insert(2,"python")
x
```
Out[2]:
```
[10, 20, 'python', 30, 40, 50]
```

In [ ]:
```
x=[10,20,30,40,50]
x.insert(200,"python")
x

Note --> If the index value is not present in the given list then index function first check weather
         the index value is positive or negative. If the index value is positive then index function
         will add the element at last position of the list. and If the index value is negative then
         given element will add at the first position.
```

In [3]:
```
x=[10,20,30,40,50]
x.insert(200,"python")
x
```
Out[3]:
```
[10, 20, 30, 40, 50, 'python']
```

In [4]:
```
x=[10,20,30,40,50]
x.insert(-200,"python")
x
```
Out[4]:
```
['python', 10, 20, 30, 40, 50]
```

In [23]:
```
x=[10,20,30,40,50]
x.insert(-1,"python")    # 1-n--> -1 --> -1-1 ==-2
x

Note --> if you are using insert function and you want to add element at the last position with teh
help of insert function then it is not possible The element will be added at the second last position.
```
Out[23]:
```
[10, 20, 30, 40, 'python', 50]
```

In [24]:
```
x=[10,20,30,40,50]
x.insert(-2,"python")
x
```
Out[24]:
```
[10, 20, 30, 'python', 40, 50]
```

In [25]:
```
x=[10,20,30,40,50]
x.insert(-5,"python")
x
```
Out[25]:
```
['python', 10, 20, 30, 40, 50]
```

Extend Function

In [ ]:
```
Extend function --> Extend function is used to add a given sequence into a list.
Syntax:
    first_list.extend(second_sequence)
```

Examples of Extend Functions

In [5]:
```
x=[10,20,30,40,50]
y=[90,80,70]
x.extend(y)
x
```
Out[5]:
```
[10, 20, 30, 40, 50, 90, 80, 70]
```

In [6]:
```
x=[10,20,30,40,50]
y=[20,30,40]
y.extend(x)
y
```
Out[6]:
```
[20, 30, 40, 10, 20, 30, 40, 50]
```

In [7]:
```
x=[10,20,30,40,50]
y=[20,30,40]
y.extend(x)
y
```
Out[7]:
```
[20, 30, 40, 10, 20, 30, 40, 50]
```

In [39]:
```
x=[10,20,30,40,50]
y=(10,)
x.extend(y)
x
```
Out[39]:
```
[10, 20, 30, 40, 50, 10]
```

In [8]:
```
x=[10,20,30,40,50]
x.extend([20,10,30,40])
x
```
Out[8]:
```
[10, 20, 30, 40, 50, 20, 10, 30, 40]
```

In [9]:
```
x=[10,20,30,40,50]
x.extend([10])
x
```
Out[9]:
```
[10, 20, 30, 40, 50, 10]
```

In [10]:
```
x=[10,20,30,40,50]
x.extend("string")
x
```
Out[10]:
```
[10, 20, 30, 40, 50, 's', 't', 'r', 'i', 'n', 'g']
```

In [ ]:
```
Note --> In extend function you need to give a sequence (list,tuple,set) for adding that sequence
         into the list. if you are giveing something else then you will get an error
```

Python Program to Insert Element at the Middle of the list

In [32]:
```
x=[10,20,30,40,50]
y=len(x)//2
print(y)
x.insert(y,"Java")
x
```
```
2
```
Out[32]:
```
[10, 20, 'Java', 30, 40, 50]
```

Append vs Extend Vs Insert

In [ ]:
```
Append  --> Will take a single element at a time and it will add that element at the last
Extend --> Will take a sequence and add all the elements of the given sequence in the first list.
Insert --> add element at the given index
Join  --> convert a list into string.
```

Deletion Functions of list

In [ ]:
```
There are two function that we use for delete an element of the list:
    1.pop Function
    2.remove Function
```

Pop Function Without Argument

In [ ]:
```
pop() function --> pop function without an argument will delete the last element of the given list.
                   and it will return the deleted element.
Syntax:
    list_name.pop()
```

Example of Pop Function Without Argument

In [15]:
```
x=[10,20,30,40,50]
x.pop()
x
```
Out[15]:
```
[10, 20, 30, 40]
```

In [16]:
```
x=[10,20,"Hello",10]
x.pop()
x
```
Out[16]:
```
[10, 20, 'Hello']
```

Pop Function With Argument(Index)

In [ ]:
```
pop function(index): pop function with an argument will delete the element based on the given index.
                     and return the deleted element.

Syntax:
    list_name.pop(index)
```

In [21]:
```
x=[10,20,30,40,50]
x.pop(2)
x
```
Out[21]:
```
[10, 20, 40, 50]
```

In [ ]:
```
x=[10,20,30,40,50]
x.pop(2000)
x
Note --> if you are using pop function and the index value that you are giving is not present
         in the given list then you will get index error.
```

In [19]:
```
x=[]
x.pop()
x

Note --> if list is empty then we cannot use pop function it will give index error.
```
```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [19], in <cell line: 2>()
      1 x=[]
----> 2 x.pop()
      3 x

IndexError: pop from empty list
```

Python Program to get the Last String from a list

In [23]:
```
x = [10, 20, 30, 40, 50, 's', 't', 'r', 'i', 'n', 'g']
y = x.index("s")
z = x[5:]
print("".join(z))
```
```
string
```

Remove Function

In [50]:
```
Remove Function --> Remove function is used to delete the element from a given list.

Syntax:
    list_name.remove(Element)
```

Example of Remove Function

In [24]:
```
x=[10,20,30,40,50,60]
x.remove(20)
x
```
Out[24]:
```
[10, 30, 40, 50, 60]
```

In [52]:
```
x=[10,20,30,40,50,60]
x.remove(2000)
x
Note --> If the element is not present then you will get an error
```
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [52], in <cell line: 2>()
      1 x=[10,20,30,40,50,60]
----> 2 x.remove(2000)
      3 x

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

Reverse Function

In [ ]:
```
reverse --> reverse Function is used to reverse a given list. It will return the reversed list.

Syntax:
    list_name.reverse()
```

Example of Reverse Function

In [25]:
```
x=[10,20,30,40,50]
x.reverse()
x
```
Out[25]:
```
[50, 40, 30, 20, 10]
```

In [26]:
```
x=[10,20,30,40,50,90,23,123,2345]
x.reverse()
x
```
Out[26]:
```
[2345, 123, 23, 90, 50, 40, 30, 20, 10]
```

Sorted Function

In [ ]:
```
Sorted Function -->  is used to sort the given element of the list either in Ascending Order or
                     descending Order.
Syntax:
    Sorted(list_name,reverse=true)   #for Descending
    Sorted(list_name)                #for Ascending
```

Example of Sorted Function

In [29]:
```
x=[90,98,786,345]
y=sorted(x)
y
```
Out[29]:
```
[90, 98, 345, 786]
```

In [30]:
```
x=[90,98,786,345]
y=sorted(x,reverse=True)
y
```
Out[30]:
```
[786, 345, 98, 90]
```

Arithmetic operators in List

In [ ]:
```
Below are the operators that we can use in List:
    + --> concatenation  --> Join of two list
    * -->repition operator -> repeat the list n number of time
    membership operator  --> in and not in
    identity operator --> is or is not
    logical operator --> and or not
    Comparsion Operator --> < >  == ....
```

Example of Each Operator in case of List

Concatenation Operator (+)

In [65]:
```
x=[10,20,30,40]
y=[60,70,80]
x+y

Note: if you are using + operator in list then  it is forsure that both the operands
    are of list type only
```
Out[65]:
```
[10, 20, 30, 40, 60, 70, 80]
```

Repetition Operator (*)

In [66]:
```
x=[10,20,30,40]
x*4
Note: if you are using * operator in list then  it is forsure that one operand is of list type and
      second operand is of int type.
```
Out[66]:
```
[10, 20, 30, 40, 10, 20, 30, 40, 10, 20, 30, 40, 10, 20, 30, 40]
```

Membership Operator(in and not in)

In [31]:
```
x=[10,20,30,40]
40 in x
```
Out[31]:
```
True
```

In [32]:
```
x=[10,20,30,40]
50 in x
```
Out[32]:
```
False
```

Identity Operator(is and is not)

In [33]:
```
x=[10,20,30,40]
y=[90,80,70]
x is y
```
Out[33]:
```
False
```

logical operator(and or and not)

In [34]:
```
[] and [60,70,80]
```
Out[34]:
```
[]
```

In [37]:
```
[10,20] and [90,80]
```
Out[37]:
```
[90, 80]
```

In [35]:
```
not []
```
Out[35]:
```
True
```

In [36]:
```
[] or [60,70,80]
```
Out[36]:
```
[60, 70, 80]
```

In [38]:
```
[10,20] or [90,80]
```
Out[38]:
```
[10, 20]
```

Comparison Operator(< , > ,== ,!= ....)

In [39]:
```
x=[10,20,30]
y=[10,20,30]
x==y
```
Out[39]:
```
True
```

In [40]:
```
x=[10,20,30]
y=[10,80,90]
x<y
```
Out[40]:
```
True
```

List vs Mutability

In [ ]:
```
List are mutable if you are going to do any change or updation in the given list object
then that updation will be done on same object. because list is a mutable object and we can perform
any change in mutable objects.
```

In [69]:
```
x=[10,20,30]
print(id(x))
x.append(40)
print(id(x))
```
```
2082209650752
2082209650752
```

In [74]:
```
x=[10,20,30,40,50]
print(id(x))
x.extend([90.80])
print(id(x))
print(x)
```
```
2082212537344
2082212537344
[10, 20, 30, 40, 50, 90.8]
```

Traversal over list.

In [ ]:
```
Traversal means visiting each and every element of a list(iteratating over the list).
```

Example:

In [41]:
```
x = [10,20,30,40,50]
for i in x:
    print(i)
```
```
10
20
30
40
50
```

In [42]:
```
x=[10,20,30,40]
for i in range(len(x)):
    print(x[i])
```
```
10
20
30
40
```

In [43]:
```
x=[10,20,30,40,50]
i=0
while i<len(x):
    print(x[i])
    i=i+1
```
```
10
20
30
40
50
```