List manipulations

- Predefined class for list is <class 'list'>
- Heterogeneous Objects are allowed [dis-similar values]
- Insertion order is maintained
- List is Mutable
- Duplication Objects are allowed
- None type Objects are allowed
- All Object in the list must be taken in between [ ]
- Every Object must be separated by using ,

Creating List Collection:

```
#App-1
lst=[10,"A",None,10,3.14]
print("Type is : ",type(lst))
print("Data is : ",lst)

#App-2
lst2=[ ]
print("type is : ",type(lst2))
print("Data is : ",lst2)

#list( ) or [ ]
lst3=list()
print("type is : ",type(lst3))
print("Data is : ",lst3)
```

```
#list(iterable) -> list object
#iterables [list | set | tuple | str | dict ...]

s="welcome"
lst4=list(s)
print("Type is : ",type(lst4))
print("Data is :",lst4)

t=(12,13,14)
lst5=list(t)
print("Type is : ",type(lst5))
print("Data is : ",lst5)

#S.split([chars]) -> list
st="have a nice day"
lst6=st.split()
print("Type is : ",type(lst6))
print("Data is : ",lst6)
```

Reading The Data From List Collection:

```
lst=[10,20,"A","B",None,3.14]

#Reading Data From List Using Index
print("First : ",lst[0])
print("First : ",lst[-6])
print("Last : ",lst[-1])
```

#Reading Data From List using slicing [start:end:step]

```
print("First 3 Objects " ,lst[0:3:1])
print("Last 3 Objects ",lst[3:6])
```

#Reading Data From Entire List

```
import time
lst=[10,20,"A","B",None,3.14]

index=0
while index<len(lst):    #len(iterable) -> int
   time.sleep(.2)
   print(lst[index])
   index=index+1

#App-2
print("-"*30)
for i in lst:
   time.sleep(.2)
   print(i)
```

Reading Data From List Using Unpacking :

- Unpacking is the process of reading from list collection to ordinary variables
- During unpacking, variables count must be same with no.of.objects are existed in list collection

```
lst=[10,"ramesh"]
'''
no=lst[0]
name=lst[1] '''
no,name=lst  #unpacking
print(no,name,sep='....')

lst2=[101,"sudha","kmm"]
eno,ename,ecity=lst2
print(eno,ename,ecity,sep='...')

#a,b=lst2 ValueError: too many values to unpack (expected 2)

eno=lst2  # valid but it is a Ref.copy
print(lst2)
print(eno)
```

List Methods :

For Adding Object

append(item):

It is used to append or add new Object to an existed List Collection [at the end of the collection]

- **Syn: L.append(item)**

Example :

```
lst=[]
print("Data is : ",lst)
```

```
item=input("Enter any Object " )
lst.append(item)
print("Result is : ",lst)
```

Example :

```
lst=[]
for i in range(1,6):
   item=int(input("enter an item : "))
   lst.append(item)
print("Result is : ",lst)
```

insert():

- Used to insert an object at the specified valid index position

Syn: L.insert(pos,item)

```
lst=[10,20,30,40]
print("Before insert : ",lst)
lst.insert(2,222)
print("After insert : ",lst)
```

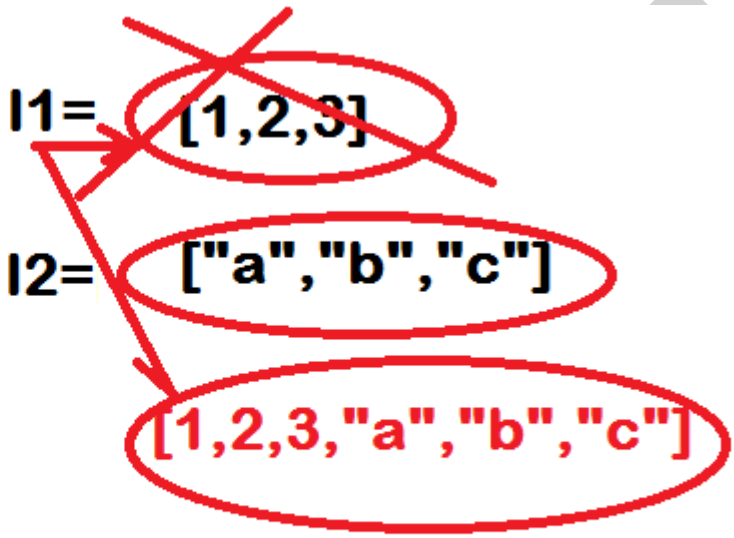
extend():

Syn: L.extend(iterable)

- it used to extend an existed list collection with given iterable Object

```
l1=[1,2,3]
l2=["a","b","c"]
l1=l1+l2
print(l1)

L.extend(iterable)
#l1=l1+l2 or
l1.extend(l2)
```



Deleting Objects:

Pop():

Syn: L.pop([index]) -> item

- Used to remove the object existed at the specified valid index it will return the deleted item , but default index value is -1

Example :

```
lst=[10,20,30,40,50]
print("Before POP ",lst)
lst.pop()
print("After POP  ",lst)
```

Example 2:

```
lst=[10,20,30,40,50]
print("Before POP ",lst)
lst.pop(1)  #del lst[1]
print("After POP  ",lst)
```

remove()

Syn: L.remove(item) -> None

- Is used to remove the specified item from list collection i.e.first item with given value

```
lst=[10,20,30,40,50,10]
print("Before remove ",lst)
lst.remove(10)
print("After Remove : ",lst)
```

clear( ):

L.clear() -> None

- It will clear [Erase all object from list collection but list object will be the memory]

Example:

```
lst=[10,20,30,40,50,10]
print("Before clear ",lst)
lst.clear()
print("After clear : ",lst)
```

Reference Copy VS Shallow Copy:

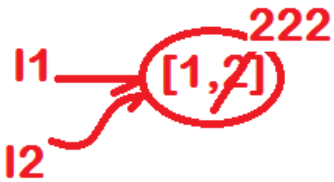
- Process of creating duplicate reference for an existed object called reference copy [i.e One Object with multiple references]

```
l1=[1,2]
l2=l1 #ref.copy

print(l1) #[1,2]
print(l2) #[1,2]

l1[1]=222

print("After Modify")
print(l1)  #[1,222]
print(l2)  #[1,222]
```



- **Shallow Copy**

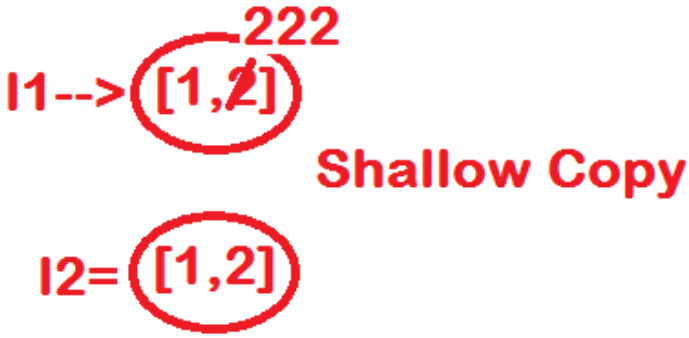
  It is the process of creating a duplicate Object for list collection

```
l1=[1,2]
l2=l1.copy()

print(l1) #[1,2]
print(l2) #[1,2]

l1[1]=222
print(l1)  #[1,222]
print(l2)  #[1,2]
```

#L.copy() -> List



#L.index(item[,start[,end]]) -> int

It will return the index position of the specified Object [first occurrence]

```
lst=[10,20,30,"A",3.14,20]
print("Data is : ",lst)
pos=lst.index(20)
print("Index of Object : ",pos)

pos=lst.index(20,2,len(lst))
print("Found @ : ",pos)
```

Count()

L.count(item) -> int

It Will return frequency of the specified Object

Example:

```
#L.count(item) -> int
lst=[10,20,30,"A",None,10]
print("List : ",lst)

noc=lst.count(10)
print("Found For : ",noc)
```

#L.reverse() -> None

It will reverse all the Object existed in the list

```
lst=[10,20,30,"A",None]
print("List : ",lst)
lst.reverse()
print("Reverse : ",lst)
```