List

- Collection of homogenous/heterogenous data structure
- Ordered collection of elements, Mutable, any depth ??
- Include duplicate elements
- Elements are accessed
  - using index (start with 0)
  - Slicing operation (forward & backward)
- List creation -> [ ] , list( )
- Concatenation -> + operator

Copy and Concatenation

- Reference copy(shallow copy) ??
  - Using = operator
  - Refer same list object (id value same)
  - Changes in list affect another list
  - Eg:

```
A=[10,20,30]
B=A
A[1]=100
B[2]='@@@'
```

- Deep copy
  - Import copy module
  - copy.copy(oldlist)
  - copy.deepcopy(oldlist)
  - Id value different (changes not reflected)

```
A=[10,20,30]
B=copy.copy(A)
A[1]=100
B[2]='@@@'
C=copy.deepcopy(A)
A[1]='333'
C[2]=99
```

Modifying List values

- List is mutable

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

- Using index position
  - A[2]=30 = > A => [1, 2, 30, 4, 5, 6, 7, 8, 9, 10]
  - A[1]=20 =>A=> [1, 20, 30, 4, 5, 6, 7, 8, 9, 10]
  - A[-4]=70 => A=> [1, 20, 30, 4, 5, 6, 70, 8, 9, 10]
- Slicing
  - A[1:4]=[11,22,33] =>A => [1, 11, 22, 33,4,5,6,70,8,9,10]
  - A[-1:-3]=> [ ]
  - A[-1:-3:-1]=[90,100]= > A=> [1, 11, 22, 33,4,5,6,70,8,100,90]

Insert item in list

- Increase the list length
- Using insert( ) method

  list_name.insert(index,item)

  - Index maybe positive and negative
  - A.insert(2,'****')
- Using append( ) method

  list_name.append(item)

  - A.append('@@@')
- Using extend( ) method

  list_name.extend(iterable_obj)

  - A.extend(B)

Remove an element from list

- Reduce the list length
- Using del keyword
  - del listname[index]
  - del A[2], del A[-1], del A
- Using pop( ) method
  - listname.pop(index)
  - Parameter is the index and return the deleted element
  - A.pop(3),A.pop(-2),A.pop( )
- Using remove ( ) method
  - listname.remove(list_value)
  - Parameter is the list values and return nothing
  - A.remove(30),A.remove('sss'),A.remove(index)

Other methods

Method	Description
len( )	Returns number of items in a list
index(item)	Returns the index position of an item(parameter) in list otherwise return error (not found)
max(list)	Returns an maximum item from the list
min(list)	Returns an minimum item from the list
type(list)	Returns the type of the list
sort( )	Sort the list items in ascending order
clear ()	Remove all elements from the list
reverse ()	Reverse the existing content of the list

Removal of duplicate elements

```
my_list = [1, 2, 3, 1, 2, 4, 5, 4 ,6, 8]
print("List Before ", my_list)


for i in range(0,len(my_list)-1):
    print('i=',i)
    for j in range(i+1,len(my_list)-1):
        print('j=',j)
        if(my_list[i]==my_list[j]):
            del my_list[j]


print("List After removing duplicates ", my_list)
```

Passing List as an argument to function

- Mutable object
- Pass by reference
- Changes in formal arg reflected in actual arg

Eg 1:

```
def modify(L):
    for i in range(len(L)):
        L[i]=L[i]+5
    print(L)
mylist=[10,20,30,40,50]
modify(mylist)
print(mylist)
```

output

```
[15, 25, 35, 45, 55]
[15, 25, 35, 45, 55]
```

Eg 2:

```
def no_modify(L):
    L=L*2
    print(L)
mylist=[10,20,30,40,50]
no_modify(mylist)
print(mylist)
```

output

```
[10, 20, 30, 40, 50, 10, 20, 30, 40, 50]
[10,20,30,40,50]
```

Eg 3:

```
def no_modify(L):
    L*=2
    print(L)
mylist=[10,20,30,40,50]
no_modify(mylist)
print(mylist)
```

output:

```
[10, 20, 30, 40, 50, 10, 20, 30, 40, 50]
[10, 20, 30, 40, 50, 10, 20, 30, 40, 50]
```

List of List

- List items are another list
- Nested list elements
- Accessed using row and column position
  - Two dimensional list
  - Extended to any depth (nD)

- A=[[10,20,30],[‘aa’,’bb’,’cc’],[2.3,12,’@@’]]
  A[0] =>[10, 20, 30]
  A[2] => [2.3, 12, '@@']
  A[2][1] => 12
- B=[[1, [2, 4], 5], [2.3, 4.3, [22, 33, 44]]
  B[0] => [1, [2, 4], 5]
  B[1] => [2.3, 4.3, [22, 33, 44]]
  B[0][1][1] => 4
  B[1][2][-1] => 44

- Nested list creation using list( )

```
b=list([list([1,2,3]),list([44,55]),list(['aa','bb'])])
b=>
[[1, 2, 3], [44, 55], ['aa', 'bb']]
```

Eg: 1

```
n = 3
m = 3
val = [0] * n
li_li=[0]*n
for x in range (n):
    val[x] = x* m
    li_li[x]=[x]*m
print(val)
print(li_li)
```

Matrix Addition

```
X = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9]]

Y = [[5,8,1],
    [6,7,3],
    [4,5,9]]

result = [[0,0,0],
          [0,0,0],
          [0,0,0]]

print('matrix addition using loop')
for i in range(len(X)):
   # iterate through columns
   for j in range(len(X[0])):
       result[i][j] = X[i][j] + Y[i][j]

for r in result:
   print(r)
```