- Representations
- type
- len
- max
- min
- sorted
- reversed
- in
- for loop in
- index
- for loop index
- mutable
- concatenation
- slicing
- methods

Representations

- string represntation with quotes
- list represnt with square brackets []

In [1]:
```
l1=[1,2,3,4]
l1
```

Out[1]:
```
[1, 2, 3, 4]
```

In [2]:
```
type(l1)
```

Out[2]:
```
list
```

In [3]:
```
l2=['A','B','C','D']
l2
```

Out[3]:
```
['A', 'B', 'C', 'D']
```

In [4]:
```
l3=[1,2,3,4,'A','B','C','D']
```

```
l3
```

Out[4]: [1, 2, 3, 4, 'A', 'B', 'C', 'D']

In [5]:
```
l4=[1,2,3,'Apple','Ball','Cat',True,False]
l4
```

Out[5]: [1, 2, 3, 'Apple', 'Ball', 'Cat', True, False]

In [6]:
```
name='naresh it'
l5=[name]
l5
```

Out[6]: ['naresh it']

In [7]:
```
l6=[100,100,100]
l6
```

Out[7]: [100, 100, 100]

In [8]:
```
l7=[100,'Apple',['A',2]]
l7
```

Out[8]: [100, 'Apple', ['A', 2]]

In [ ]:
```
l1=[1,2,3,4]
l2=['A','B','C','D']
l3=[1,2,3,4,'A','B','C','D']
l4=[1,2,3,'Apple','Ball','Cat',True,False]
name='naresh it'
l5=[name]
l6=[100,100,100]
l7=[100,'Apple',['A',2]]
```

- list are array of elements
- list elements are heterogeneous all the data types we can represent in a list
- list elements are allowed duplicates
- list in list is possible
- variable assignment also possible in the list

In [ ]:
```
- len

- max

- min

- sorted

- reversed
```

In [ ]:
```
l1=[1,2,3,4]
l2=['A','B','C','D']
```

```
l3=[1,2,3,4,'A','B','C','D']
l4=[1,2,3,'Apple','Ball','Cat',True,False]
name='naresh it'
l5=[name]
l6=[100,100,100]
l7=[100,'Apple',['A',2]]
len()
max()
min()
sorted()
reversed()
```

In [9]:
```
l4=[1,2,3,'Apple','Ball','Cat',True,False]
reversed(l4)
```

Out[9]: <list_reverseiterator at 0x20be8d4aa40>

In [10]:
```
for i in reversed(l4):
    print(i)
```

```
False
True
Cat
Ball
Apple
3
2
1
```

In [11]:
```
l4=[1,2,3,'Apple','Ball','Cat',True,False]
list(reversed(l4))
```

Out[11]: [False, True, 'Cat', 'Ball', 'Apple', 3, 2, 1]

In [14]:
```
name=['adithya','aadithya','banu']
max(name)
# First compare all first letters
# If all first letters ascii values same
# then compare with second letters
# Continue the process
```

Out[14]: 'banu'

In [15]:
```
list1=['Apple',
       1,
       ['Cherry','Banana']
      ]
len(list1)
```

Out[15]: 3

In [ ]:
```
list1=[65,'A']
max(list1) # error

# same data type only comparable
```

index

In [16]:
```
l4=[1,2,3,'Apple','Ball','Cat',True,False]

#-8  -7  -6    -5     -4    -3    -2    -1
#1    2   3   Apple  Ball  Cat   True   False
#0    1   2    3      4     5      6     7
```

In [18]:
```
l4[3],l4[-5]
```

Out[18]: ('Apple', 'Apple')

In [19]:
```
l1=[1,[2]]
l1[0]
```

Out[19]: 1

In [23]:
```
new=l1[1]
new[0]
```

Out[23]: 2

In [ ]:
```
l1=[1,[2]]
new=l1[1]  #  [2]   will come again it is a list, one element
new[0]

l1[1][0]
```

In [25]:
```
l1=[1,[2,'A']]
l1[1][1]

# 'A' is presented at index  1
# so we given l1[1] : [2,'A'] it is also a list
# The list has two elements
# 'A' at 1st index
# l1[1][1]
```

Out[25]: 'A'

In [30]:
```
# How to get 'A'
#  1    [2,'A']
#  0      1
l1=[1,[2,'A']]
# Q1) How many elements are there in a list: 2
# Q2) 'A' is at which index :
# In python index start at  0
# 0 and 1
l1[1]    #  [2,'A']
# Q3) l1[1]  output is which type list
# Q4) 'A'  1
l1[1][1]
```

Out[30]: 'A'

In [39]:
```
l1=[1,  # 0
    [2,['A','B']]  # 1
   ]
# I want 'B'
# How many elements
```

```
len(l1)
l1[1][1]
```

Out[39]: ['A', 'B']

In [48]:
```
l1=[1,2,[3,[4,[5,['Apple']]]]]
# 2110
# 21110
len(l1)  #  index=2
l1[2][1][1][1][0]
```

Out[48]: 'Apple'

In [51]:
```
list1=['Apple',
       1,
       'Solapur',
       ['cherry','papaya',70,[123,'Banana'],
        'tomato'],
       'python'] #get banana  331
list1[3][3][1]
```

Out[51]: 'Banana'

In [58]:
```
list1=[[[[[[[1,'Cherry']]]]]]]
list1[0][0][0][0][0][0][1]
```

Out[58]: 'Cherry'

In [62]:
```
list1=['MH',
       ['Nagpur',['Orange',['likes',['king'],['shivaji',['Son',['shmbaji']]]]]
       ]]
len(list1)
```

Out[62]: 2

In [69]:
```
list1[1][1][1][2][1][1][0]
```

Out[69]: 'shmbaji'

In [ ]:
```
# In operator : For loop
# Index : For loop
# Mutability check
# Slicing
#
```

In [1]:
```
l4=[1,2,3,'A','B','C',True,False]

#-8  -7   -6   -5    -4    -3    -2    -1
#1   2    3    'A'   'B'   'C'   True  False
#0   1    2     3     4     5     6     7
for i in range(len(l4)):
    print(l4[i])
```

```
1
2
3
A
B
C
True
False
```

In [3]:
```python
l4=[1,2,3,'A','B','C',True,False]
for i in range(len(l4)):
    print(f"The postive index of {l4[i]} is {i}")

s='welcome'
for i in range(len(s)):
    print(f"The postive index of {s[i]} is {i}")
```

```
The postive index of 1 is 0
The postive index of 2 is 1
The postive index of 3 is 2
The postive index of A is 3
The postive index of B is 4
The postive index of C is 5
The postive index of True is 6
The postive index of False is 7
```

In [ ]:
```python
l4=[1,2,3,'A','B','C',True,False]
for i in range(len(l4)):
    print(f"The postive index of {l4[i]} is {i}")
```

In [5]:
```python
# We want to change 1  with 100
l=[1,2,3]
l[0]=100
l

#Hi Sir - how the string is immutable  please explain
s='Naresh'
# We want to change 'h'  'I'
s[5]='I'

# TypeError: 'str' object does not support item assignment
# Not able
```

Out[5]:
```
[100, 2, 3]
```

In [ ]:
```python
l=[1,2,3,4,5,6,'A','B','C','D','E','F','G']
-13  -12  -11  -10  -9   -8  -7  -6  -5  -4  -3  -2  -1
1,    2,   3,   4,   5,   6, 'A','B','C','D','E','F','G'
0     1    2    3    4    5   6   7   8   9   10  11  12

l[3:9:2] # P
l[3:9:-2] # Np
l[3:-9:2] # -----
l[3:-9:-2] # Np
l[-3:9:2]  # Np
l[-3:-9:-2] # W
l[-3:9:-2]
```

In [11]:
```python
l=[1,2,3,4,5,6,'A','B','C','D','E','F','G']
l[3:-9:2]
```

```
# start=3
# dire =+ve
# last= stop-1 = -9-1=-10

l[-3:9:-2]
```

Out[11]: ['E']

In [ ]:
```
# Tomorrow you have a exam
# MCQ in Python

# Sunday you dont have class

# Friday 50-50 (26)
# Sat (27)
# Sun (28)

# Aug 4 class
# Aug 11 class
```

List Methods

In [12]:
```
# strings dir('')
dir([])
```

Out[12]:
```
['__add__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

In [ ]:
```
'append',
'clear',
'copy',
'count',
'extend',
'index',
'insert',
'pop',
'remove',
```

```
'reverse',
'sort'
```

append

- adding elements in a list at last index
- append method is very important
- most of the time we will use append in the real time
- When you add the value the original list will be overwrite

In [13]:
```
l=['Apple','Ball','Cat']
l
```

Out[13]:
```
['Apple', 'Ball', 'Cat']
```

In [ ]:
```
# You want to add Doll in a given list

# Keywords  vs methods
len()
max()
min()
reversed()
sorted()

# Methods
<package>.<method>()
# we already discussed
# Data types has own method
# <data type>.<method>
# <pacckage>.<methods>
```

In [18]:
```
l.append('Doll')
```

In [19]:
```
l
```

Out[19]:
```
['Apple', 'Ball', 'Cat', 'Doll', 'Doll']
```

In [22]:
```
l1=[1,2,3,4]
l1.append(100)
l1
```

Out[22]:
```
[1, 2, 3, 4, 100]
```

In [24]:
```
l2=[]
l2.append(10)
l2.append(20)
l2.append('apple')
l2.append(['A',1])
l2.append(True)
l2
```

Out[24]:
```
[10, 20, 'apple', ['A', 1], True]
```

In [ ]:
```
list.append() takes exactly one argument how can we add more in one statement si
l1.append(100,200,300)  # Error
```

Append method used to save the output in a list

In [25]:
```
for i in range(1,6):
    print(i*i)
```

```
1
4
9
16
25
```

In [29]:
```
square_list=[]
for i in range(1,6):
    square_list.append(i*i)
# print ==== square_list.append
square_list
```

Out[29]:
```
[1, 4, 9, 16, 25]
```

In [ ]:
```
s=''
for i in 'python':
    s=s+i

# str='hai hai hai'
# what are  index of  'a'
# save that index numbers in a list
```

In [34]:
```
str1='hai hai hai'
id_list=[]
for i in range(len(str1)):
    if str1[i]=='a':
        id_list.append(i)

id_list,sum(id_list)
```

Out[34]:
```
([1, 5, 9], 15)
```

In [33]:
```
summ=0
str1='hai hai hai'
for i in range(len(str1)):
    if str1[i]=='a':
        summ=summ+i

summ
```

Out[33]:
```
15
```

In [ ]:
```
# Q2)l1=['Hyd','Mumbai','Chennai','blr']
#  ans=['Mumbai','Chennai']
#  we want lements which are len of element >4

# Q3)l1=['Hyd','Mum#bai','Chen#nai','blr']
#  ans=['Mum#bai'',Chen#nai']
#  we want lements which are having '#'
```

```
# Q4)l1=['hyd','mumbai','chennai','blr']
#  ans= ['Hyd','Mumbai','Chennai','Blr']
#  we want lements which are len of element >4

# Q5)l1=['Hyd','Mumbai','chennai','blr']
#  ans= ['Hyd','Mumbai']
#  we want lements which are having first letter capital

# Q6)l1=['Hyd','Mum#bai','Chen#nai','blr']
#  ans_#=['Mum#bai'',Chen#nai']
#  ans_without_#=['Hyd'','blr']
#  we want lements which are having '#'

# Q7) ask the user get 5 numbers randomly
#     even_list and odd_list
#     even numbers should append at even_list
#     odd number should append at odd_list

# Q8) str='hello hai how are you'
# Maximum len of word using split and max method
# sum of all the indexes of the maximum len of word using append

# Q9) str1='virat.kohli@rcb.com, Rohit.sharma@mi.co, KL.Rahul@lucknow.com'
# Firstname=[]  second name=[]  cname =[]
# append first name should be in first name list
#        second name shoul be in second name list
#        thirs name will be in thirs name list

# 10 )
#  You have two lists
# qns=['What is capital of India',
       'Who is PM of india',
        'Who is ICT ODI captian']
# ans = ['Delhi','Modi','Rohit']

# For i in qns:
#     print(i)
#     ans= delhi
#      index should match
#      delhi  modi
#      marks= marks+1
# print the total marks
```

In [4]:
```
# Q2)l1=['Hyd','Mumbai','Chennai','blr']
#  ans=['Mumbai','Chennai']
#  we want lements which are len of element >4
l1=['Hyd','Mumbai','Chennai','blr']
ans=[]
for i in l1:
    if len(i)>4:
        ans.append(i)
ans
```

Out[4]:
```
['Mumbai', 'Chennai']
```

In [5]:
```
# Q3)l1=['Hyd','Mum#bai','Chen#nai','blr']
#  ans=['Mum#bai'',Chen#nai']
#  we want lements which are having '#'
l1=['Hyd','Mumb#ai','Chen#nai','blr']
```

```
ans=[]
for i in l1:
    if '#' in i:
        ans.append(i)
ans
```

Out[5]: ['Mumb#ai', 'Chen#nai']

In [ ]:
```
l1=['Hyd','Mumb#ai','Chen#nai','blr']
for i in l1:
    if i in '#'
```

In [9]:
```
'nadeem' in '#'  # 'nadeem' is avialable in '#'
'#' in 'nadeem'  # is '#'  available in 'nadeem'
'nad##m' in '#'  # is 'nad##m'  avaialabe in '#'
'#' in 'nad##m'  # is '#'  avaialabe in 'nad##m'
```

Out[9]: True

In [12]:
```
# Q5)l1=['Hyd','Mumbai','chennai','blr']
#  ans= ['Hyd','Mumbai']
#  we want lements which are having first letter capital
l1=['Hyd','Mumbai','Chennai','blr']
ans=[]
for i in l1:
    ans.append(i.capitalize())
ans

# what is the difference betweeb capitalize and title
```

Out[12]: ['Hyd', 'Mumbai', 'Chennai', 'Blr']

In [13]:
```
str1='hello im learning python'
# Make everyword capitalize
str1.title()
```

Out[13]: 'Hello Im Learning Python'

In [ ]:
```
# Do the same using capitalize method
# step-1: split the string
# step-2: take empty list
# step-3: iterate the each letter apply capitlize and append it
# step-4: join the list
```

split-join

- split will apply for strings , to divide the words or characters
- when we split the elements will be stored in a list format
- Join is used to combine the elements of a list in a string format
- **split-strings**
- **join-list**

In [20]:
```
str1='hello im learning python'
words=str1.split()
new_words=[]
for i in words:
    new_words.append(i.capitalize())
' '.join(new_words)
```

Out[20]: 'Hello Im Learning Python'

In [21]:
```
# Calculate the distance between two points
# a=[1,2]
# b=[4,5]
# a=[x1,x2]
# b=[y1,y2]
# step-0: Import math
# step-1:v1= x2-x1=a[2]-a[1]   === index
# step-2:v2= y2-y1=b[2]-b[1]   === index
# step-3: v11=v1**2
# step-4: v22=v2**2
# step-5: ans=math.sqrt(v11+v22)
import math
a= [1,2]
b = [1,5]
v1 = a[1]-a[0]
v2 = b[1]-b[0]
v11 = v1**2
v22 = v2**2
print(math.sqrt(v11+v22))
```

```
4.123105625617661
```

In [25]:
```
import math
a= [1,2]
b = [1,5]
print(math.sqrt(math.pow((a[1]-a[0]),2)+math.pow((b[1]-b[0]),2)))
```

```
4.123105625617661
```

In [ ]:
```
[1,2,3,4,5,4,3,2,1]
[100,500,600,700,400,300,50]
```

In [26]:
```
pints=[[1,2],[2,3],[3,4]]
val=[2,3]

# calculate the distance between val [2,3] to each and every point
# and find the maximum distance
import math
a= [1,2]  # [x1,y1]
b = [1,5] # [x2,y2]
# for loop
v1 = b[0]-a[0] # x2-x1
v2 = b[1]-a[1] # y2-y1
v11 = v1**2
v22 = v2**2
print(math.sqrt(v11+v22))
```

```
3.0
```

In [28]:
```
points=[[1,2],[2,3],[3,4]]
val=[2,3]
```

```
# [1,2]  vs   [2,3]
# [2,3]  vs   [2,3]
# [3,4]  vs  [2,3]
for i in points:
    print(i[0],i[1])  # x1,y1
    print(val[0],val[1]) # x2,y2
    print('====================')
```

```
1 2
2 3
====================
2 3
2 3
====================
3 4
2 3
====================
```

In [42]:
```
points=[[1,2],[2,3],[5,6]]
val=[2,3]
ans=[]
for i in points:
    print(i[0],i[1])
    print(val[0],val[1])
    o1=val[0]-i[0]
    o2=val[1]-i[1]
    print(f"{val[1]}-{val[0]}={o1},{val[0]}-{i[0]}={o2}")
    v11 = o1**2
    v22 = o2**2
    print(math.sqrt(v11+v22))
    ans.append(math.sqrt(v11+v22))
    print('=======================')
```

```
1 2
2 3
3-2=1,2-1=1
1.4142135623730951
=======================
2 3
2 3
3-2=0,2-2=0
0.0
=======================
5 6
2 3
3-2=-3,2-5=-3
4.242640687119285
=======================
```

In [44]:
```
max(ans)
```

Out[44]: 4.242640687119285

In [48]:
```
# Q8) str='hello hai how are you'
# Maximum len of word using split and max method
# sum of all the indexes of the maximum len of word using append
str1='hello hai how are you'
words=str1.split()
str2=[]
for i in words:
```

```
    x=len(i)
    str2.append(x)
str2
```

Out[48]: [5, 3, 3, 3, 3]

In [53]:
```
str1='hello hai how are you'
words=str1.split()
len_Words=[]
for i in words:
    x=len(i)
    len_words.append(x)
max_len=max(len_words)

for i in words:
    if len(i)==max_len:
        print(i)

# First we are splitting
# the we are calculating each and every len of words and append in a list
# we  are taking the max value from the list

# again we are itertating words list
# calculating len of each word
# if that word length equal to max value
# then that is our answer

hello  == 0
01234
```

```
hello
```

In [56]:
```
str1='hello hai how are you'
words=str1.split()
max(words)
len_words=[5,3,3,3,3]
max(len_words)
```

Out[56]: 5

In [3]:
```
for i in range(10):
```

```
  Cell In[3], line 1
    for i in range(10):
                       ^
SyntaxError: incomplete input
```

In [ ]:
```
# 10 )
#  You have two lists
# qns=['What is capital of India',
       'Who is PM of india',
        'Who is ICT ODI captian']
# ans = ['Delhi','Modi','Rohit']

# For i in qns:
#     print(i)
#     ans= delhi
#      index should match
#      delhi  modi
```

```
#      marks= marks+1
# print the total marks
```

In [9]:
```
qns_list=['What is capital of India',
       'Who is PM of india',
        'Who is ICT ODI captian']
ans_list = ['Delhi','Modi','Rohit']
marks=0
for i in range(len(qns_list)):
    ans=input(qns_list[i])
    if ans.lower()==ans_list[i].lower():
        marks=marks+1  # marks=+1 (best practice)


print("The total marks are:",marks)


# Step-1: We are iterating throuh qns list
# step-2: We are taking each qns and we are providing the answer
# Step-3: That answer == anslist
```

```
The total marks are: 3
```

In [6]:
```
ans
```

Out[6]:
```
'rohit'
```

In [10]:
```
dir([])
```

Out[10]:
```
['__add__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

In [ ]:
```
'clear',
'copy',
'count',
'extend',
'index',
'insert',
'pop',
'remove',
'reverse',
'sort'
```

```
# Append vs extend vs concatenation vs insert
# pop vs remove
# reverse vs reversed
# sort vs sorted
# index
# count
# clear and copy

# Methods  vs keywords
# keywords : inbuilt functions
# print()  max()   min()  len(<) sorted(<>)  reversed(<l>)
# l.copy()
# l.clear()
```

Copy

In [11]:
```
l1=[1,2,3,100,200,80,70,60]
l2=l1.copy()
l2
```

Out[11]: [1, 2, 3, 100, 200, 80, 70, 60]

In [13]:
```
l1,l2
```

Out[13]: ([1, 2, 3, 100, 200, 80, 70, 60], [1, 2, 3, 100, 200, 80, 70, 60])

clear

In [14]:
```
l1.clear()
l1
```

Out[14]: []

In [15]:
```
l2
```

Out[15]: [1, 2, 3, 100, 200, 80, 70, 60]

In [16]:
```
print(type([]))
```

```
<class 'list'>
```

append-extend-concatenation-insert

In [19]:
```
l1=[1,2,3,4]
l2=['A','B','C','D']
l1.append(l2)
print(l1)
print(l2)

# l1 list is overwrire with l2 values
```

```
[1, 2, 3, 4, ['A', 'B', 'C', 'D']]
['A', 'B', 'C', 'D']
```

In [20]:
```
l1=[1,2,3,4]
l2=['A','B','C','D']
l1.extend(l2)
```

```
print(l1)
print(l2)
```

```
[1, 2, 3, 4, 'A', 'B', 'C', 'D']
['A', 'B', 'C', 'D']
```

In [24]:
```
l1=[1,2,3,4]
l2=['A','B','C','D']
l2.extend(l1)
print(l1)
print(l2)
```

```
[1, 2, 3, 4]
['A', 'B', 'C', 'D', 1, 2, 3, 4]
```

In [26]:
```
l1=[1,2,3,4]
l2=['A','B','C','D']
print(l1+l2)
print(l2+l1)
print(l1)
print(l2)
```

```
[1, 2, 3, 4, 'A', 'B', 'C', 'D']
['A', 'B', 'C', 'D', 1, 2, 3, 4]
[1, 2, 3, 4]
['A', 'B', 'C', 'D']
```

In [28]:
```
l1=[1,2,3,4]
l1.extend('apple')
l1
#[1,2,3,4,'a','p','p','l','e']
```

Out[28]:
```
[1, 2, 3, 4, 'a', 'p', 'p', 'l', 'e']
```

In [30]:
```
l1=[1,2,3,4]
l1.extend((10,20,30,40))
l1
```

Out[30]:
```
[1, 2, 3, 4, 10, 20, 30, 40]
```

- append just append the value with base data type
- extend is concatenate the value, and the list will be overwrite with new values
- extend can take any iterable format type i.e. string,list,tuple
- extend and concatenation both are same
- the list will not overwrite in concatenation

index

- same like string only

In [34]:
```
l1=[1, 2, 3,'Apple', 10, 20, 'Apple', 40]
l1.index(2)
i1=l1.index('Apple')
```

```
i2=l1.index('Apple',i1+1)
i1,i2
```

Out[34]: (3, 6)

In [35]:
```
l1.index('Apple',l1.index('Apple')+1)
```

Out[35]: 6

In [39]:
```
l1=[1, 2, 3,'Apple', 10, 20, 'Apple', 40]
#   0  1  2   3      4   5    6       7
i1=l1.index('Apple')  # Case-1
i2=l1.index('Apple',i1+1) # Case-2
i3=l1.index('Apple',2,5) # Case-3
print(i3)
# If i want second 'Apple'
# from which index my eyes will start
```

3

- case-1: Directly get the index
- case-2: Next value index, by providing start
- case-3: Between the values

- Copy/clear
- Append/Extend/Concatenation
- Index
- There is no find in list
- count is exactly same as strings

insert

- will insert the element based on index
- append will add the element at last only

In [4]:
```
l1=[1, 2, 3,'Apple', 10, 20, 'Apple', 40]
#   0  1  2   3      4   5    6       7
i1=l1.index('Apple')  # Apple index = 3
l1.insert(i1,'Banana') # insert banana before index i1(3)
l1
```

Out[4]: [1, 2, 3, 'Banana', 'Apple', 10, 20, 'Apple', 40]

In [5]:
```
l1=[1, 2, 3,'Apple', 10, 20, 'Apple', 40]
# Insert 'Cherry'  before 40
# You need to find index of the 40
i1=l1.index(40)
```

```
l1.insert(i1,'Cherry')
l1
```

Out[5]: `[1, 2, 3, 'Apple', 10, 20, 'Apple', 'Cherry', 40]`

In [9]:
```
#If we want to add cherry before  2nd apple then how ?
l1=[1, 2, 3,'Apple', 10, 20, 'Apple', 40]
# step-1: we want to find the index of 2nd 'APPLE'
# Step-2: How to find the second 'Apple'  index
i1=l1.index('Apple')
i2=l1.index('Apple',i1+1)
l1.insert(6,'Cherry')
l1
```

Out[9]: `[1, 2, 3, 'Apple', 10, 20, 'Cherry', 'Apple', 40]`

pop-remove-del

- pop will remove the item
- pop also display the item which is removing
- pop will take one argument:index
- pop will remove the item based on index
- if you dont provide any index, by default it will remove last value
- because the default index value is -1

In [12]:
```
l1=[1, 2, 3,'Apple', 10, 20, 'Apple', 40]
l1.pop()
l1
```

Out[12]: `[1, 2, 3, 'Apple', 10, 20, 'Apple']`

In [13]:
```
# We are concentarting only on first Apple
l1=[1, 2, 3,'Apple', 10, 20, 'Apple', 40]
i1=l1.index('Apple')
l1.pop(i1)
l1
```

Out[13]: `[1, 2, 3, 10, 20, 'Apple', 40]`

In [14]:
```
# We are concentarting only on Second apple
l1=[1, 2, 3,'Apple', 10, 20, 'Apple', 40]
i1=l1.index('Apple')
i2=l1.index('Apple',i1+1)
l1.pop(i2)
l1
```

Out[14]: `[1, 2, 3, 'Apple', 10, 20, 40]`

In [17]:
```
l1=[1, 2, 3,'Apple', 10, 20, 'Apple', 40]
i1=l1.index('Apple') #3
l1.pop(i1)  #  l1=[1, 2, 3,10, 20, 'Apple', 40]
#####################################################################
```

```
i1=l1.index('Apple')
l1.pop(i1)
l1
```

Out[17]: [1, 2, 3, 10, 20, 40]

In [18]:

Out[18]: [1, 2, 3, 10, 20, 40]

In [23]:
```
str='hai apple how are you apple im good apple thank you apple'
# o/p='hai how are you im goof thank you'
# Idea:
# step-1:  split
# itearte it
# pop it
# join it
list1=str.split()
count=list1.count('apple')
for i in range(count):
    i1=list1.index('apple')
    list1.pop(i1)
' '.join(list1)
```

Out[23]: 'hai how are you im good thank you'

- pop will take the index
- remove will take the value
- remove will delete the first occurence of element

In [25]:
```
l1=[1, 2, 3,'Apple', 10, 20, 'Apple', 40]
l1.remove('Apple')
l1
```

Out[25]: [1, 2, 3, 10, 20, 'Apple', 40]

In [26]:
```
str='hai apple how are you apple im good apple thank you apple'
list1=str.split()
count=list1.count('apple')
for i in range(count):
    list1.remove('apple')
' '.join(list1)
```

Out[26]: 'hai how are you im good thank you'

del

In [28]:
```
l1=[1, 2, 3,'Apple', 10, 20, 'Apple', 40]
del(l1[3])
l1
```

Out[28]: [1, 2, 3, 10, 20, 'Apple', 40]

- append / extend/ insert/concatenation
- pop/remove/del
- sort/sorted
- reverse/reveresd
- count and index

In [ ]:
```
- representation
- type
- len
- max
- min
- sum
- sorted
- reversed
- in
- for loop with in
- index
- for loop with index
- mutable and immutable
- slicing
- concatenation
- Methods
```

In [29]:
```
dir(())
```

Out[29]:
```
['__add__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'count',
 'index']
```

In [ ]: