26-Nov

- intilaization
- inbuilt functions
  - min
  - max
  - len
  - sorted
  - reversed
  - print
  - type
  - sum
- index operations (**for loop range vs in**)
- concatenation
- mutable vs immutable
- slicing
- list Methods

- lists denoted with square brackets
- list means array of elements
- list means array of elements
- liist can access any data types together

In [149…
```
l1 = [1,2,3,4]
l1
```
Out[149…
```
[1, 2, 3, 4]
```
In [150…
```
type(l1)
```
Out[150…
```
list
```
In [151…
```
l2 = ['A',"B","C"]
```

```
l2
```

Out[151…

```
['A', 'B', 'C']
```

In [152…

```
l3 = [1,2,3,4,'A',"B","C"]
l3
```

Out[152…

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

In [154…

```
l4 = [10,20,30,'Apple','Banana','Cherry',True,False,10.5,20.5,20+30j]
l4
```

Out[154…

```
[10, 20, 30, 'Apple', 'Banana', 'Cherry', True, False, 10.5, 20.5, (20+30j)]
```

List can represent with any data types

In [155…

```
l5 = [10,10,10]
l5
```

Out[155…

```
[10, 10, 10]
```

Duplicates are allowed

In [156…

```
l6 = [10,20,30,['A','B','C']]
l6
```

Out[156…

```
[10, 20, 30, ['A', 'B', 'C']]
```

List inside a List are allowed

In [157…

```
l7=[]
l7
```

Out[157…

```
[]
```

In [10]:

```
l8 = [_]
l8
```

Out[10]:

```
[[]]
```

In [14]:

```
l8 = [_]
l8
```

Out[14]:

```
[[[0]]]
```

_ underscore means a variable

In [24]:

```
[python]  # Error

[_]       # Answer
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[24], line 1
----> 1 [python]  # Error
      3 [_]       # Answer

NameError: name 'python' is not defined
```

- List can access with square brackets
- List can have any data type of elements
  - Heterogeneous
- List can have any duplicates
- List in List is Possible
- Empty List can also Possible
- The values inside the list is called as elements

- **for all 7 min max len sorted reversed**

```
In [26]: len(l1),len(l2),len(l3),len(l4),len(l5),len(l6),len(l7)
Out[26]: (4, 3, 7, 11, 3, 4, 0)
```

```
In [27]: l1
Out[27]: [1, 2, 3, 4]
```

```
In [28]: min(l1),max(l2)
Out[28]: (1, 'C')
```

```
In [29]: l2
Out[29]: ['A', 'B', 'C']
```

```
In [31]: min(l2),max(l2)
Out[31]: ('A', 'C')
```

```
In [32]: l3
Out[32]: [1, 2, 3, 4, 'A', 'B', 'C']
```

```
In [33]: min(l3),max(l3)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[33], line 1
----> 1 min(l3),max(l3)

TypeError: '<' not supported between instances of 'str' and 'int'
```

Very Very Important

- Same data types shoubld be compare
- 'A' is characters ==== some level
- 1 is numerical type ==== some different level

```
In [35]: l4

         # Min Max Fails

Out[35]: [10, 20, 30, 'Apple', 'Banana', 'Cherry', True, False, 10.5, 20.5, (20+30j)]

In [36]: l5

Out[36]: [10, 10, 10]

In [37]: min(l5),max(l5)

Out[37]: (10, 10)

In [38]: l6

Out[38]: [10, 20, 30, ['A', 'B', 'C']]

In [39]: chr(10)

Out[39]: '\n'

In [41]: chr([])

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[41], line 1
----> 1 chr([])

TypeError: 'list' object cannot be interpreted as an integer

In [42]: min(l6)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[42], line 1
----> 1 min(l6)

TypeError: '<' not supported between instances of 'list' and 'int'

In [43]: l7
```

```
Out[43]: []
```

```
In [44]: min(l7)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[44], line 1
----> 1 min(l7)

ValueError: min() iterable argument is empty
```

```
In [45]: max(l7)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[45], line 1
----> 1 max(l7)

ValueError: max() iterable argument is empty
```

```
In [25]: len([]),len([100]),len([''])

         # is list has elements
```

```
Out[25]: (0, 1, 1)
```

```
In [19]: min([''])
```

```
Out[19]: ''
```

```
In [28]: len('100')
```

```
Out[28]: 3
```

```
In [ ]: [100] #1
        [1,0,0] #3
        ['100'] #1 Here list so, output will be 1
        len('100') #3
```

Note

- Strings and Number can not compare
- 10 with 'A'
- 10 Level 0 ====== No Other level in python
  - 'A' ===> 65 level1
  - 'B' ===> 66 level1
  - level1 compair with level1 only

Sorted

```
In [164…: sorted(l1),sorted(l2),sorted(l5),sorted(l7)
```

```
Out[164…  ([1, 2, 3, 4], ['A', 'B', 'C'], [10, 10, 10], [])
```

- By default sorted is in accendening order

```
In [165…  l1 # W
          l2 # W
          l3 # Hetro NW
          l4 # NW
          l5 # W
          l6 #NW
          l7 #W

Out[165…  []
```

```
In [39]:  sorted([])

          #empty list means
          # does not has any elements

Out[39]:  []
```

- empty list means
- does not has any elements

```
In [1]:  l1=['Nest','Mango','Zebra','Elephant','Apple']
         sorted(l1)

Out[1]:  ['Apple', 'Elephant', 'Mango', 'Nest', 'Zebra']
```

```
In [2]:  l1=['Nest','Mango','Zebra','Elephant','Apple']
         sorted(l1,key=len)

Out[2]:  ['Nest', 'Mango', 'Zebra', 'Apple', 'Elephant']
```

```
In [3]:  l1=['Nest','Mango','Zebra','Elephant','Apple']
         sorted(l1,reverse=True)

Out[3]:  ['Zebra', 'Nest', 'Mango', 'Elephant', 'Apple']
```

```
In [4]:  l1=['Nest','Mango','Zebra','Elephant','Apple']
         sorted(l1,key=len,reverse=True)

Out[4]:  ['Elephant', 'Mango', 'Zebra', 'Apple', 'Nest']
```

```
In [40]:  min([])

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[40], line 1
----> 1 min([])

ValueError: min() iterable argument is empty
```

- both are compared elements min and sorted

- min will show error
- but sorted will not
    - Answer => Min will required iterable argument
    - but sorted will not required

reversed

```
In [44]: reversed(l1),reversed(l2),reversed(l3),reversed(l4),reversed(l5),reversed(l6),re
```

```
Out[44]: (<list_reverseiterator at 0x16bab5f0b50>,
          <list_reverseiterator at 0x16bab5f0bb0>,
          <list_reverseiterator at 0x16bab5f2230>,
          <list_reverseiterator at 0x16bab5f0c10>,
          <list_reverseiterator at 0x16bab5f2440>,
          <list_reverseiterator at 0x16bab5f02b0>,
          <list_reverseiterator at 0x16bab5f2350>)
```

```
In [58]: list(reversed(l1))
```

```
Out[58]: [4, 3, 2, 1]
```

```
In [46]: for i in reversed(l1):
             print()
```

```
4
3
2
1
```

```
In [47]: for i in reversed(l2):
             print(i)
```

```
C
B
A
```

```
In [48]: for i in reversed(l3):
             print(i)
```

```
C
B
A
4
3
2
1
```

```
In [49]: for i in reversed(l4):
             print(i)
```

```
(20+30j)
20.5
10.5
False
True
Cherry
Banana
Apple
30
20
10
```

In [50]:
```
for i in reversed(l5):
    print(i)
```

```
10
10
10
```

In [51]:
```
for i in reversed(l6):
    print(i)
```

```
['A', 'B', 'C']
30
20
10
```

In [53]:
```
for i in reversed(l7):
    print(i)
```

In [59]:
```
list(reversed(l2))
```

Out[59]:
```
['C', 'B', 'A']
```

- Two ways to see the answer of reversed
  - using for loop
    - for i in reversed(l1): print(i)
  - using list
    - list(reversed(l1))

- sorted will compare the elements
- the same rules applicable for min and max
- sorted never return any error eventhough we have an empty list
- min and max will give error if it has empty list
- reversed only do reverses elements

In [ ]:
```
min/max/sorted/reversed/len

- completed
```

sum

```
In [60]: sum([100,200,300])

         # by default start is 0
```

```
Out[60]: 600
```

```
In [61]: sum(['A','B','C'])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[61], line 1
----> 1 sum(['A','B','C'])

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

- when we press shift + tab
    - two elemnets will be the
        - iterabel
        - start = 0
- **iterable means**:- any thing can be iterate through loop
    - thats is might be string or list

```
In [62]: sum('23','34')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[62], line 1
----> 1 sum('23','34')

TypeError: sum() can't sum strings [use ''.join(seq) instead]
```

```
In [2]: sum([100,200,300],start=100)

        # start=100 means sum = 100 after all will added inside sum
```

```
Out[2]: 700
```

```
In [3]: sum=0
        i=120
        sum=sum+i
        sum
```

```
Out[3]: 120
```

```
In [67]: sum([])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[67], line 1
----> 1 sum([])

TypeError: 'int' object is not callable
```

indexing

In [70]:
```
l1 = [1,2,3,4,'A','B','C']
```

In [71]:
```
# -7  -6 -5 -4  -3   -2   -1
# [1, 2, 3, 4, 'A', 'B', 'C']
# 0   1   2   3  4    5   6
```

In [72]:
```
l1[0],l1[-7]
```

Out[72]:
```
(1, 1)
```

Que:-

In [75]:
```
for i in range(len(l1)):
    print(f'the positive index of {l1[i]} is {i}')
```

```
the positive index of 1 is 0
the positive index of 2 is 1
the positive index of 3 is 2
the positive index of 4 is 3
the positive index of A is 4
the positive index of B is 5
the positive index of C is 6
```

In [79]:
```
for i in range(len(l1)):
    print(f'The negative index of {l1[i]} is {i-len(l1)}')
```

```
The negative index of 1 is -7
The negative index of 2 is -6
The negative index of 3 is -5
The negative index of 4 is -4
The negative index of A is -3
The negative index of B is -2
The negative index of C is -1
```

In [81]:
```
for i in range(len(l1)):
    print(f'The positive index of {l1[i]} is {i} and The negative index of {i-le
```

```
The positive index of 1 is 0 and The negative index of -7
The positive index of 2 is 1 and The negative index of -6
The positive index of 3 is 2 and The negative index of -5
The positive index of 4 is 3 and The negative index of -4
The positive index of A is 4 and The negative index of -3
The positive index of B is 5 and The negative index of -2
The positive index of C is 6 and The negative index of -1
```

In [82]:
```
l = [10,20,30]
l[0]
```

Out[82]:
```
10
```

In [83]:
```
l=[[10,20,30]]
# How to access the 10
len(l)
```
Out[83]:
```
1
```

In [86]:
```
l=[[10,20,30]]

# Question yourself:- How many elements are there in a list?
# ans:- 1 element is there
# how to prove simple len(L)

len(l) #1
```
Out[86]:
```
1
```

- Always look inside the entire list how many list is present
- [
- [10,20,30]
-]
- In above ex this is [10,20,30] consider as 1 element

In [93]:
```
print(l[0]) # [10,20,30]
# how many elements will available? ans:- 3
print(len(l[0])) # 3
```
```
[10, 20, 30]
3
```

How to access - list inside list

In [94]:
```
print(l[0][0],l[0][1],l[0][2])
```
```
10 20 30
```

In [100…
```
l1 = [10,20,['A','B']]
# we want 'A' as output
len(l1)
# Que:- how many elements are there ? => 3
# step1- How can we access=> 0,1,2
# L1[2] => ['A','B']
# step2:- how many elements are there ? => 2
# How can we access=> 0,1
# L1[2][0] =>'A'

l1[2][0] # 'A'
```
Out[100…
```
'A'
```

In [116…
```
l1 = [1,2,3,4,[5,6,["Apple"]]]
len(l1)
```

```
Out[116… 5

In [105… len(l1[4])

Out[105… 3

In [115… l1[4][2][0]

Out[115… 'Apple'

In [118… l1 =[1,2,3,[4,[5,['cherry']]]]
         len(l1)

Out[118… 4

In [119… l1[3]

Out[119… [4, [5, ['cherry']]]

In [120… len(l1[3])

Out[120… 2

In [121… l1[3][1]

Out[121… [5, ['cherry']]

In [122… len(l1[3][1])

Out[122… 2

In [123… l1[3][1][1]

Out[123… ['cherry']

In [125… len(l1[3][1][1])

Out[125… 1

In [126… l1[3][1][1][0]

Out[126… 'cherry'

In [127… l1=[[[[[[[['banana']]]]]]]]
         len(l1)

Out[127… 1

In [130… l1[0][0][0][0][0][0][0][0]

Out[130… 'banana'

In [131… l1 =['Apple',['Kishmir',['India',['Mumbai',['SRK',['Film',['DDLJ']]]]]]]

In [132… len(l1)
```

```
Out[132… 2
```

```
In [147… l1[1][1][1][1][1][1][0]
```

```
Out[147… 'DDLJ'
```

27-Nov

Mutable Vs Immutable

```
In [4]: str1 = 'welcome'
        str1[2]='L'
        str1
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[4], line 2
      1 str1 = 'welcome'
----> 2 str1[2]='L'
      3 str1

TypeError: 'str' object does not support item assignment
```

```
In [7]: l1 = [10,20,30]
        l1[2]=300
        l1

        # Here no error will come that's why List is Mutable
        # So we can change the list elements using index
```

```
Out[7]: [10, 20, 300]
```

We can change the list elements Using index

Concatenation

```
In [8]: l1 = ['Hi']
        l2 = ['Hello']
        l1+l2
```

```
Out[8]: ['Hi', 'Hello']
```

```
In [9]: l1-l2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[9], line 1
----> 1 l1-l2

TypeError: unsupported operand type(s) for -: 'list' and 'list'
```

```
In [10]: l1*l2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[10], line 1
----> 1 l1*l2

TypeError: can't multiply sequence by non-int of type 'list'
```

In [11]:
```
l1/l2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 1
----> 1 l1/l2

TypeError: unsupported operand type(s) for /: 'list' and 'list'
```

In [12]:
```
l1*2
```

Out[12]:
```
['Hi', 'Hi']
```

In [ ]:
```
l1+l2 # Works
l1-l2 # Error
l1*l2 # Error
l1/l2 # Error
l1*2 # Works
```

Slicing

In [14]:
```
l1 = [1,2,3,4,5,'A','B','C','D','10.5',True,10.5,100,200]
```

In [15]:
```
l1[:] # Same List
```

Out[15]:
```
[1, 2, 3, 4, 5, 'A', 'B', 'C', 'D', '10.5', True, 10.5, 100, 200]
```

In [16]:
```
l1[::] # Same List
```

Out[16]:
```
[1, 2, 3, 4, 5, 'A', 'B', 'C', 'D', '10.5', True, 10.5, 100, 200]
```

In [18]:
```
l1[::-1] #reversed List
```

Out[18]:
```
[200, 100, 10.5, True, '10.5', 'D', 'C', 'B', 'A', 5, 4, 3, 2, 1]
```

In [19]:
```
l1[2:14:2] # Works gap of 2
```

Out[19]:
```
[3, 5, 'B', 'D', True, 100]
```

In [23]:
```
l1[2:14:-2] #Not Works
```

Out[23]:
```
[]
```

In [22]:
```
l1[2:-14:-2] # Works
```

Out[22]:
```
[3]
```

In [25]:
```
l1[-2:14:2] #Works
```

```
Out[25]: [100]
```

```
In [27]: l1[-2:-14:-2] #Works
```

```
Out[27]: [100, True, 'D', 'B', 5, 3]
```

```
In [29]: l1[14:2:2] #NW
```

```
Out[29]: []
```

```
In [31]: l1[14:2:-2] #Works
```

```
Out[31]: [200, 10.5, '10.5', 'C', 'A', 4]
```

```
In [33]: l1[14:-2:2] #NW
```

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

```
In [35]: l1[-14:2:2] #Works
```

```
Out[35]: [1]
```

```
In [37]: l1[-14:-2:-2] #NW
```

```
Out[37]: []
```

Methods

```
In [39]: dir([])
```

- append
- clear
- copy
- count
- extend
- index
- insert
- pop
- remove
- reverse
- sort

```
In [40]: l1 =[10,20,30,40]
         l1
```

```
Out[40]: [10, 20, 30, 40]
```

```
In [ ]: #####################inbuilt function###########

        print()
        len()
        type()

        ################ Methods ##################

        l1.<method_name>()
        l1.append()
        method_name(l1) # Wrong or Error will come
```

Copy

```
In [51]: l1 = [10,20,30,40]
         l2 = l1.copy()
```

```
In [52]: l2
```

```
Out[52]: [10, 20, 30, 40]
```

clear

```
In [53]: print(l2) # here elements are available
         l2.clear()
         l2 # Here it will Clear
```

```
[10, 20, 30, 40]
Out[53]: []
```

reversed

- reversed is a inbulit function
- reversed can be applicable all *iterable*
- reversed applicable for **String, list, tuple, dict, etc...**
  - Ex:-
    - 1. reversed(string)
    - 2. reversed(list)
    - 3. reversed(tuple)
    - 4. reversed(dict)

reverse

- reverse is a method is belongs to only list
- reverse method can not applicable to strings,tuple, and dict
- list.reverse()

reverse

```
In [64]: l1 = [10,20,30,40]
         l1.reverse()
         l1

         # Here shift+tab then Inplace is available
```

```
Out[64]: [40, 30, 20, 10]
```

Here shift+tab then Inplace is available

- reverse the elements and save the output in same variable
- It is Indicates as **Inplace*
- Some Time Inplace = True available

```
In [58]: str1 = 'hello how are you'
         str1.reverse()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[58], line 2
      1 str1 = 'hello how are you'
----> 2 str1.reverse()

AttributeError: 'str' object has no attribute 'reverse'
```

```
In [59]: reversed(str1)
```

```
Out[59]: <reversed at 0x1f3af150310>
```

```
In [66]: print(list(reversed(l1)))
         l1.reverse()
         l1
```

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

```
Out[66]: [40, 30, 20, 10]
```

In List only

- above cell reversed and reverse ans should same
- when we use reversed
  - then answer check using for loop or list
- When we use reverse
  - then the answer will in original vairable or overwrite in given variables
  - like l1 as it is reverse

```
In [ ]: l1.reverse() # Answer
        str.reverse() # Error
```

sort vs sorted

sorted():

Returns a new list: It does not modify the original list. Instead, it returns a new list that is sorted.

Works on any iterable: sorted() can be used on any iterable (e.g., list, tuple, string, etc.).

sort():

Modifies the list in place: sort() sorts the list it is called on and does not return a new list. It changes the original list.

Works only on lists: sort() is a method specific to lists, so you cannot use it on other iterables like tuples or strings.

```
In [11]: l1 = [1122,22,343,454,45]

         sorted(l1)
```

```
Out[11]: [22, 45, 343, 454, 1122]
```

```
In [12]: l1 # no changes in l1 if we want output then stored in a new variable
```

```
Out[12]: [1122, 22, 343, 454, 45]
```

```
In [7]: l1 = [1122,22,343,454,45]
        l1.sort()
```

```
In [9]: l1 # The original list will modified
```

```
Out[9]: [22, 45, 343, 454, 1122]
```

Here shift+tab then Inplace is available

- It is Indicates as **Inplace**
- Some Time Inplae=True available

```
In [70]: sorted(l1)
```

```
Out[70]: [22, 45, 343, 454, 1122]
```

```
In [71]: print(sorted(l1))
         l1.sort()
         l1
```

```
[22, 45, 343, 454, 1122]
Out[71]: [22, 45, 343, 454, 1122]
```

In List only

- above cell sorteded and sort ans should same

- when we use sorted
    - then answer directly comes
- When we use sort()
    - then the answer will in original vairable/ the change happed in original variable
    - like l1 as it is sort

In [ ]:
```
l1.sort() #Works
string.sort() # Fail

sorted(l1) #Works
sorted(string) #Works
```

append

- append means add an elements at the end of the list
- append is a method very very important
- append is used to store the outputs in a list
- till last class we just printed all the output
- if we want to save the output we need to do append only
- append method we will use very very frequently
- **Append add elements to end of the list.**

In [75]:
```
# empty List

l1 = []
l1.append(10)
l1
```

Out[75]:
```
[10]
```

In [78]:
```
l1 = [1,2,3]
l1.append('Apple')
l1
```

Out[78]:
```
[1, 2, 3, 'Apple']
```

In [79]:
```
l1 = [1,2,3,4]
l1.append('Apple')
l1.append(['Banana'])
l1.append([True,False])

l1
```

Out[79]:
```
[1, 2, 3, 4, 'Apple', ['Banana'], [True, False]]
```

In [83]:
```
# Que:- Create a list of 10 number using for loop

for i in range(1,11):
    print(i, end=' ')
```

```
1 2 3 4 5 6 7 8 9 10
```

- step-1 Take an empty list
    - l1=[]
- step-2:- write as usuall for loop
    - for i in range(1,11):
        - print(i)
- step-3:- Instead of printing this time use append
    - l1.append(i)

In [85]:
```
l1=[]
for i in range(1,11):
    l1.append(i)

l1
```

Out[85]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

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

# what is common? => l1.append() consider as it is
# what is changing? => 1,2,3,4 consider it (i)
```

In [ ]:
```
# QUE2:- wap ask the user get 5 random numbers
# perform the square of the random numbers
# save in a list
```

In [88]:
```
import random
l2= []
for i in range(5):
    n1 = eval(input('Enter a numbers'))
    l2.append(n1*n1)

l2
```

Out[88]:
```
[5625, 7056, 25, 576, 256]
```

In [91]:
```
import random
l2= []
for i in range(5):
    n1 = random.randint(1,100)
    l2.append(f'{n1}:{n1*n1}')

l2
```

Out[91]:
```
['64:4096', '100:10000', '95:9025', '89:7921', '36:1296']
```

In [92]:
```
# Que:3- list1 = [100,27,20,37,38,47,52,87,94,239]
# using this we need to extract even and odd numbers
# even_list = []
# odd_list = []
```

In [94]:
```
list1 = [100,27,20,37,38,47,52,87,94,239]
even_list = []
odd_list = []

for i in list1:
    if i%2==0:
        even_list.append(i)
    else:
        odd_list.append(i)

print(even_list)
print(odd_list)
```

```
[100, 20, 38, 52, 94]
[27, 37, 47, 87, 239]
```

In [95]:
```
even_list.sort()
even_list
```

Out[95]:
```
[20, 38, 52, 94, 100]
```

In [96]:
```
odd_list.sort()
odd_list
```

Out[96]:
```
[27, 37, 47, 87, 239]
```

In [100…
```
# QUE:4- l1 = ['hyd','chennai','mumbai','pune']
# ans= ['Hyd', 'Chennai', 'Mumbai', 'Pune']
l1 = ['hyd','chennai','mumbai','pune']
ans=[]
for i in l1:
    ans.append(i.capitalize())

ans
```

Out[100…
```
['Hyd', 'Chennai', 'Mumbai', 'Pune']
```

In [102…
```
#Que5:- l1 = ['hyd','chennai','mumbai','pune']
# ans1 = ['HYD', 'CHENNAI', 'MUMBAI', 'PUNE']

l1 = ['hyd','chennai','mumbai','pune']
ans1=[]
for i in l1:
    ans1.append(i.upper())

ans1
```

Out[102…
```
['HYD', 'CHENNAI', 'MUMBAI', 'PUNE']
```

In [112…
```
#Que6:- ['hyd','chen#ai','mu#bai','pune']
# ans=['chen#ai', 'mu#bai']
```

```
l1 = ['hyd','chen#ai','mu#bai','pune']
ans=[]
for i in range(len(l1)):
    if '#' in l1[i]:
        ans.append(l1[i])

ans
```

Out[112…

```
['chen#ai', 'mu#bai']
```

Note

- what is the meaning of (i in '#')
  - ans:- it will check chen#ai in '#'
- what is the meaning of ('#' in i)
  - ans:- it will chec '#' in chen#ai

In [115…

```
#Que7:- ['hyd','chen#ai','mu#bai','pune']
# ans=['hyd', 'pune']

l1 = ['hyd','chen#ai','mu#bai','pune']
ans = []
for i in range(len(l1)):
    if '#' not in l1[i]:
        ans.append(l1[i])

ans
```

Out[115…

```
['hyd', 'pune']
```

In [14]:

```
# Que8:- str1 = 'hello hai how are you'
# ans = ['Hello','Hai','How','Are','You']

str1 = 'hello hai how are you'
s1 = str1.title()
s1.split()
```

Out[14]:

```
['Hello', 'Hai', 'How', 'Are', 'You']
```

Note

- split output comes in the list form

joining the list of elements

- we can convert string to list by using **string.split**

```
- str1.split()
```

- we can convert list to string by using **join(list)**

```
- ' '.join(l1)
```

In [169…
```
str1 = 'hai how are you'

str1.split() # by default space se seperate krta hai
```
Out[169…
```
['hai', 'how', 'are', 'you']
```
In [170…
```
str1 = 'hai how are you'

str1.split('h')
```
Out[170…
```
['', 'ai ', 'ow are you']
```
In [164…
```
l1 = ['hai', 'how', 'are', 'you']
str1=' '
str1.join(l1)
```
Out[164…
```
'hai how are you'
```
In [165…
```
l1 = ['hai', 'how', 'are', 'you']

' '.join(l1)
```
Out[165…
```
'hai how are you'
```
In [166…
```
l1 = ['hai', 'how', 'are', 'you']

'*'.join(l1)
```
Out[166…
```
'hai*how*are*you'
```

- **How you want to join (सामील होणे)**
  - it will give here
  - '*'.join(l1)
  - "hai* how* are*you"
- **How you want to split (विभाजन)**
  - it will give here
  - str1.split('h')
  - ['', 'ai ', 'ow are you']
- **Main thing how you want to join and how you want to split you will decide**

In [184…
```
# Que9:- str1 = 'virat.kohli@rcb.com; rohit.sharma@mi.com; ms.dhoni@csk.com'
# fname= ['virat','rohit','ms']
# sname= ['kohli','sharma','dhoni']
# cname= ['rcb','mi','csk']
str1 = 'virat.kohli@rcb.com; rohit.sharma@mi.com; ms.dhoni@csk.com'

s1 = str1.split(';')
first_dot = s1[0].index('.')
at_sym = s1[0].index('@')
second_dot = s1[0].index('.',first_dot+1)
f_name = s1[0][:first_dot]
s_name = s1[0][first_dot+1:at_sym]
```

```
c_name = s1[0][at_sym+1:second_dot]
print(f_name,s_name,c_name)
```

```
virat kohli rcb
```

In [185…

```
str1 = 'virat.kohli@rcb.com; rohit.sharma@mi.com; ms.dhoni@csk.com'

s1 = str1.split(';')
for i in s1:
    first_dot = i.index('.')
    at_sym = i.index('@')
    second_dot = i.index('.',first_dot+1)
    f_name = i[:first_dot]
    s_name = i[first_dot+1:at_sym]
    c_name = i[at_sym+1:second_dot]
    print(f_name,s_name,c_name)
```

```
virat kohli rcb
 rohit sharma mi
 ms dhoni csk
```

In [186…

```
s1[0] # virat
s1[1] # rohit
s1[2] # dhoni

instead of s1[0] we can take simply i and iterate it
```

Out[186…

```
('virat.kohli@rcb.com', ' rohit.sharma@mi.com', ' ms.dhoni@csk.com')
```

- instead of s1[0] we can use in above code i so we can iterate it easily
- step-1 we are check for one virat kohli
- step-2 apply using for loop to all then it will iterate

In [189…

```
str1 = 'virat.kohli@rcb.com; rohit.sharma@mi.com; ms.dhoni@csk.com'
s1 = str1.split(';')
f_name,s_name,c_name=[],[],[]
for i in s1:
    first_dot = i.index('.')
    at_sym = i.index('@')
    second_dot = i.index('.',first_dot+1)
    f_name.append(i[:first_dot])
    s_name.append(i[first_dot+1:at_sym])
    c_name.append(i[at_sym+1:second_dot])
print(f_name)
print(s_name)
print(c_name)
```

```
['virat', ' rohit', ' ms']
['kohli', 'sharma', 'dhoni']
['rcb', 'mi', 'csk']
```

Tip

- The above code beauty is
- we are checking for 1 element means s1[0]

- we understand the pattern and after that we can putting inside the loop or iterate on all elements

In [1]:
```
# Que 10:- get the 7 random numbers in a list between 1 to 100
# find the min and max value without using min and max function

import random
l1 = []

for i in range(7):
    n1 = random.randint(1,100)
    l1.append(n1)
print(l1)
print(max(l1))
print(min(l1))
```
```
[93, 93, 95, 60, 74, 65, 63]
95
60
```

In [ ]:
```
max_val = <position>

81 > max_val then max_val = 81
55 > max_val False
98 > max_val then max_val= 98
21 > max_val False
62 False
28 False
3 False

min_val = <position>

81 < min_val then min_val = 81
55 < min_val True then min_val = 55
98 < min_val False
21 < min_val True then min_val = 21
62 False
28 False
3 True min_val= 3
```

In [5]:
```
max_val = l1[0]
for i in l1[1:]:
    if i>max_val:
        max_val=i
print(max_val)
```
```
95
```

Very Very Important Quetion

- Assume that first value is a maximum value
- then iterate the loop from next value onwards
- apply the condition if any value greater than assumed value
- then replace max value with iterated value

In [12]:
```
max_val = l1[0]
for i in l1[1:]:
    if i>max_val:
        max_val=i
print('max_val',max_val)
```
```
max_val 95
```

In [9]:
```
min_val = l1[0]
for i in l1[1:]:
    if i<min_val:
        min_val=i

print('min_val',min_val)
```
```
min_val 60
```

In [160…
```
# Que11:- str1 = 'can canner can not you cannner can be can you can not'
# list = ['can-6','canner-2','not-2','you-2','be-1']

str1 = 'can canner can not you canner can be can you can not'
can = 0
canner=0
no=0
you=0
be=0

lst = []
for i in range(len(str1)):
    if str1[i:i+3] == 'can':
        can = can+1
    elif str1[i:i+6] == 'canner':
        canner = canner+1
    elif str1[i:i+3] == 'not':
        no = no+1
    elif str1[i:i+3] == 'you':
        you = you+1
    elif str1[i:i+2] == 'be':
        be = be+1
lst.append(f'can - {can}')
lst.append(f'canner - {canner}')
lst.append(f'not - {no}')
lst.append(f'you - {you}')
lst.append(f'be - {be}')
print(lst)
```
```
['can - 7', 'canner - 0', 'not - 2', 'you - 2', 'be - 1']
```

In [157…
```
str1 = 'can canner can not you canner can be can you can not'
canner=0
for i in range(len(str1)):
    if str1[i:i+6] == 'canner':
        canner = canner+1
print(canner)
```
```
2
```

In [16]:
```
# Que11:- str1 = 'can canner can not you cannner can be can you can not'
# list = ['can-6','canner-2','not-2','you-2','be-1']

str1 = 'can canner can not can you canner can be can you can not'
```

```
l = str1.split()
l1 = []
l2 = []
count=0
for i in l:
    if i not in l1:
        l1.append(i)
        l2.append(l.count(i))

l2
```

Out[16]:
```
[6, 2, 2, 2, 1]
```

In [ ]:
```
str1 = 'can canner can not can you canner can be can you can not'

l = str1.split()

# step-1: first 'can' will coming
# step-2: in list there is a method count
# step-3 l.count('can')
    # meaning is how many times can will be available
    # ans = 6
    # but there is a drawback
    # 1st time i == 'can' and ans = 6
    # 2nd time i == 'canner' and ans = 2
    # 3rd time i == 'can' it will come again so we will avoid the repetation
# step-4 so we can use the method called unique vowel to avoid the repetation
    # like a,e,i,o,u
    # we need empty list l2=[]
# step-5 we can add one condition
    # if i not in l2:
#            print(i,l.count(i))
            # l2.append(i)
```

drawback code below

In [28]:
```
for i in l:
    print(i,l.count(i))
```

```
can 6
canner 2
can 6
not 2
can 6
you 2
canner 2
can 6
be 1
can 6
you 2
can 6
not 2
```

- we take a empty list l1=[]
- add condition

- if i not in l1:
  - print(i,l.count(i))
  - l2.append(i)

In [124…

```
str1 = 'can canner can not can you canner can be can you can not'
l2=[]
l = str1.split() # here we can seperate
#print(l)
print()

for i in l:
    if i not in l2:
        print(i,l.count(i))
        l2.append(i)
```

```
can 6
canner 2
not 2
you 2
be 1
```

In [161…

```
# Que12:- que = ['Who is Pm of India', 'Who is ICT captain','What is the Capital
# ans = ['Modi','Rohit','Delhi']

# step-1 = iterate through each qn
# step-2 = user will enter the answer
    # check-1:- the user given 'Modi' correct answer
    # check-2:- qn index and ans index should be
# step-3: count_marks = 0 at the top
# step-4: for every correct answer 1 Marks
# step-5: How many correct answers and how many marks
```

In [37]:

```
que = ['Who is Pm of India', 'Who is ICT captain','What is the Capital of India'
ans = ['Modi','Rohit','Delhi']
count = 0
for i in range(len(que)):
    ANSWER = input(que[i])
    if ANSWER.lower() == ans[i].lower():
        count = count + 1
        print("Correct")
print(f"The total correct answer: {count}")
```

```
Correct
The total correct answer: 1
```

pop vs remove

In [38]:

```
l = [100,200,300,400,'A','B',"C"]

l.pop()
```

Out[38]: 'C'

pop

- pop will remove the element based on index
- If we don't give any index by default it will remove last value

- The default value is -1

In [39]:
```
l
```

Out[39]:
```
[100, 200, 300, 400, 'A', 'B']
```

In [40]:
```
l.pop(2)

# what will return as output
# ans => output comes removed value ans=> 300
```

Out[40]:
```
300
```

In [41]:
```
l.pop(200)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[41], line 1
----> 1 l.pop(200)

IndexError: pop index out of range
```

remove

In [43]:
```
l = [100, 200, 300, 100, 100, 400, 'A', 'B']
l.remove(100)
l
```

Out[43]:
```
[200, 300, 100, 100, 400, 'A', 'B']
```

difference

- pop except a **index value**
- remove except a **value** inside the list

In [ ]:
```
l = [100,200, 300, 100, 100, 400, 'A', 'B']

# Que:- I want to remove the second 100
# which one will use remove or pop
# ans=> pop is correct
l.pop(3)

# pop wants a index
# here only 8 elements are there, so we are able to count
# imagine that there 80k elements  so we are not able to count
# when you are counting and giving: hard coded
#python code should give the answer automaticallly
# for that we need to use index method
```

index

In [46]:
```
i1 = l.index(100)
i2 = l.index(100,i1+1)
i2 # now we know 2nd 100 index is 3
```

```
l.pop(i2)
```

Out[46]:
```
100
```

In [47]:
```
l
```

Out[47]:
```
[200, 300, 100, 400, 'A', 'B']
```

In [48]:
```
dir(())
```

Out[48]:
```
['__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']
```

29th Nov

extend

- difference between concatenation vs append vs extend

In [22]:
```
l1 = [1,2,3,4]
l2 = ['A','B','C','D']
```

In [18]:
```
l1.append(l2)
```

In [19]:
```
l1
```
Out[19]:
```
[1, 2, 3, 4, ['A', 'B', 'C', 'D']]
```

In [16]:
```
l1 = [1,2,3,4]
l2 = ['A','B','C','D']
l1+l2
# if you see the output stored in a new variable l3
```
Out[16]:
```
[1, 2, 3, 4, 'A', 'B', 'C', 'D']
```

In [31]:
```
# Concatenation

l1 = [1,2,3,4]
l2 = ['A','B','C','D']
l3 = l1+l2
l3
```
Out[31]:
```
[1, 2, 3, 4, 'A', 'B', 'C', 'D']
```

- In above concatenation we required new variable to stored the value
- but when we use extend no required to new variable
- output we overwrite automatically

In [29]:
```
l1 = [1,2,3,4]
l2 = ['A','B','C','D']
l1.extend(l2)
l1
```
Out[29]:
```
[1, 2, 3, 4, 'A', 'B', 'C', 'D']
```

In [35]:
```
l1 = ['You']
l2 = ['Yours']
l1+l2 # adding you and yours
l1  # but obly you coming

l1.extend(l2) # adding you and yours
l1 # both will come
```
Out[35]:
```
['You', 'Yours']
```

- extend and concatenation both has same behaviour
- means add the two lists
- but extend will overwrite the list

insert

- Difference between insert and append

In [44]:
```
l1 = [1,2,3,4]
l2 = ['A','B','C','D']
l1.extend(l2)
l1.insert(3,120)
# I want to insert 120 before=3 (at 2)
l1
```

Out[44]:
```
[1, 2, 3, 120, 4, 'A', 'B', 'C', 'D']
```

In [48]:
```
l1 = [1, 2, 3, 120, 4, 'A', 'B', 'C', 'D']
# I want insert 1000 before 'B'
i1 = l1.index('B')
l1.insert(i1,1000)
l1
```

Out[48]:
```
[1, 2, 3, 120, 4, 'A', 1000, 'B', 'C', 'D']
```

- append and insert both work are same
- when we use append value or element it will added in the end of list
- But when we use insert it required index number
- it will add the value before index we will provided

In [57]:
```
# Que: wap ask the user add the elements of the lists
# l1 = [100,200,300]
# l2 = [10,20,30]
# ans = [110,220,330]

l1 = [100,200,300]
l2 = [10,20,30]
ans = []
l1[0]+l2[0] #110
l1[1]+l2[1] #220
l1[2]+l2[2] #330

# common l1[]+l2[]
# changeing i
for i in range(len(l1)):
    ans.append(l1[i]+l2[i])

ans
```

Out[57]:
```
[110, 220, 330]
```

In [68]:
```
# Que
# l1 = [100,200,300,400]
# l2 = [10,20,30]
# ans = [110,220,330,400]

l1 = [100,200,300,400]
l2 = [10,20,30]

# step1- iterate the loop with max lenght iteratios
# if l2 values available then add those
# otherwise append l1 values into ans list

for i in range(len(l1)):
    if i < len(l2):
        print(l1[i]+l2[i])
    else:
        print(l1[i])
```

```
110
220
330
400
```

In [70]:

```
l1 = [100,200,300,400]
l2 = [10,20,30]
ans = []
# step1- iterate the loop with max lenght iteratios
# if l2 values available then add those
# otherwise append l1 values into ans list
len1 = len(l1)
len2 = len(l2)
max_val = max(len1,len2)
for i in range(max_val):
    if i < len(l2):
        ans.append(l1[i]+l2[i])
    else:
        ans.append(l1[i])

ans
```

Out[70]:

```
[110, 220, 330, 400]
```

In [76]:

```
l1 = [100,200,300,400]
l2 = [10,20,30]
ans = []

len1 = len(l1)
len2 = len(l2)
max_len = max(len1,len2)
min_len = min(len1,len2)
for i in range(max_len):
    if i < min_len:
        ans.append(l1[i]+l2[i])
    else:
        ans.append(l1[i])

ans
```

Out[76]:

```
[110, 220, 330, 400]
```

Distance Between Two points

In [82]:

```
# Que:
import math
#d = root([x2-x1]**2+[y2-y1]**2)

# given  l1=[2,5] l2=[4,9]
#           x1,y1     x2,y2
```

```
l1=[2,5]
l2=[4,9]

x1 = l1[0]
y1 = l1[1]
x2 = l2[0]
y2 = l2[1]

# step -2: d1 = x2-x1, d2 = y2-y1
# step-3: (x2-x1)**2  (y2-y1)**2
        # (d1)^2         (d2)^2
# step-4: d = (d1)^2 +(d2)^2
# step:5 math.sqrt(d)

d1 = (x2-x1)
d2 = (y2-y1)
d = d1**2 + d2**2
math.sqrt(d)
```

```
Out[82]: 4.47213595499958
```

```
In [102…
l1=[2,5]
l2=[4,9]
#####################################################
x1 = l1[0]
y1 = l1[1]
x2 = l2[0]
y2 = l2[1]
#####################################################
d1 = (x2-x1)
d2 = (y2-y1)
#####################################################
d = d1**2 + d2**2
###################################################3#
round(math.sqrt(d),2)
```

```
Out[102… 4.47
```

```
In [105…
round(math.sqrt((l2[0]-l1[0])**2 + (l2[1]-l1[1])**2),2)
```

```
Out[105… 4.47
```

Very IMP

- 1st we can use logic on above code and find ans
- after the founded logic we can use a for loop on below code
- and answer comes

- instead of l2 and l1 we can simply taking b

```
In [109…
# QUE
import math
# wap
# a = [(1,3),(7,8),(2,6),(9,3)]
# b = (5,9)
# find the max and min distance points

a = [(1,3),(7,8),(2,6),(9,3)]
b = (5,9)

x1 = a[0][0]
y1 = a[0][1]
x2 = b[0]
y2 = b[1]
a[0]
a[1]
a[2]
# wht is common? a[]
# wht is changing? i

for i in a:
    l1 = i
    print(round(math.sqrt((b[0]-l1[0])**2 + (b[1]-l1[1])**2),2))
```

```
7.21
2.24
4.24
7.21
```

In [108…

```
a = [(1,3),(7,8),(2,6),(9,3)]
b = (5,9)

for i in a:
    l1 = i
    print(i)
```

```
(1, 3)
(7, 8)
(2, 6)
(9, 3)
```

In [119…

```
a = [(1,3),(7,8),(2,6),(9,3)]
b = (5,9)

x1 = a[0][0]
y1 = a[0][1]
x2 = b[0]
y2 = b[1]
a[0]
a[1]
a[2]
# wht is common? a[]
# wht is changing? i
ans = []
for i in a:
    l1 = i
    ans.append(round(math.sqrt((b[0]-l1[0])**2 + (b[1]-l1[1])**2),2))
max(ans),min(ans)
```

Out[119…

```
(7.21, 2.24)
```

In [ ]:

```
# wap
# a = [(1,3),(7,8),(2,6),(9,3),(2,9),(10,12)]
# find all the distances among the points
# (1,3) with all other data points
# (7,8) with others data points
# and so on
```

In [122…

```
a = [(1,3),(7,8),(2,6),(9,3),(2,9),(10,12)]
ans = []

# we want 1,3 with 7,8
#         1,3 with 2,6
#         1,3 with 9,3
#         1,3 with 2,9

for i in a:
    for j in a[1:]:
        ans.append(round(math.sqrt((j[0]-i[0])**2 + (j[1]-i[1])**2),2))

print(max(ans))
print(min(ans))
print()
ans
```

```
12.73
0.0
```

Out[122…

```
[7.81,
 3.16,
 8.0,
 6.08,
 12.73,
 0.0,
 5.39,
 5.39,
 5.1,
 5.0,
 5.39,
 0.0,
 7.62,
 3.0,
 10.0,
 5.39,
 7.62,
 0.0,
 9.22,
 9.06,
 5.1,
 3.0,
 9.22,
 0.0,
 8.54,
 5.0,
 10.0,
 9.06,
 8.54,
 0.0]
```

above 3 -4 quetions are very very IMP

- Practice as much as you can
- all are ask in interview

- **List and string quetions they will ask 100% so Prctice**

End List and String

In [ ]:

```

```