list

- list is an array of elements
- in square brackets[ ]

- How to display
- type
- len
- max
- min
- sum
- reverse
- sorted
- in
- not in
- index
- for loop
- mutable
- immutable
- slice
- concatenation
- methods

In [1]:
```
list1=[1,2,3,4]
list1
```
Out[1]:
```
[1, 2, 3, 4]
```

In [2]:
```
type(list1)
```
Out[2]:
```
list
```

In [7]:
```
list2=['apple', 'ball', 'cat']
list2
```
Out[7]:
```
['apple', 'ball', 'cat']
```

In [8]:
```
list3=[1,2,3,'apple','ball','cat']
list3
```
Out[8]:
```
[1, 2, 3, 'apple', 'ball', 'cat']
```
In [3]:
```
list4=[1,'apple',10.5,True]
list4
```
Out[3]:
```
[1, 'apple', 10.5, True]
```
In [5]:
```
list5=[100,100,100]
list5
```
Out[5]:
```
[100, 100, 100]
```
In [6]:
```
list6=['Apple','Ball',[1,2,3]]
list6
```
Out[6]:
```
['Apple', 'Ball', [1, 2, 3]]
```

- List is a hetrogenous in nature it contains all type of values in it
- Which means list items can be all datatypes
- List items can be duplicates
- Which means list have same values multiple items
- list in list (list have another list inside)

In [ ]:
```
list1=[1,2,3,4]
list2=['apple', 'ball', 'cat']
list3=[1,2,3,'apple','ball','cat']
list4=[1,'apple',10.5,True]
list5=[100,100,100]
list6=['Apple','Ball',[1,2,3]]
```

len

In [9]:
```
list1=[1,2,3,4]
len(list1)
```
Out[9]:
```
4
```
In [29]:
```
max(list1)
```
Out[29]:
```
4
```
In [30]:
```
min(list1)
```
Out[30]:
```
1
```
In [34]:
```
for i in reversed(list1):
    print(i,end='')
```
```
4321
```
In [35]:
```
sorted(list1)
```

```
Out[35]:  [1, 2, 3, 4]
```

```
In [39]:  1  in list1    #True
          2 in list1    #True
          3 in list1    #True
          4 in list1    #True
          5 in list1    #not True(False)

          for i in list1:
              print(i)

          1
          2
          3
          4
```

```
In [40]:  5 not in list1

Out[40]:  True
```

```
In [44]:  list1[0],list1[1],list1[2],list1[3]

Out[44]:  (1, 2, 3, 4)
```

```
In [28]:  list1=[1,2,3,4]
          type(list1)

Out[28]:  list
```

max

```
In [8]:   list1=[[[1,2,3,['Apple',['Fruites',['Mango',['Cherry']]]]]]]
          list1[0][0][3][1][1][0]

Out[8]:   'Mango'
```

```
In [20]:  list1=[[[[[[[['Fruites',[[['Banana']]]]]]]]]]]
          list1[0][0][0][0][0][0][0][1][0][0][0]

Out[20]:  'Banana'
```

```
In [25]:  list1=[1,2,3,['Apple',['Fruites',['Cherry']]]]
          list1[3][1][1][0]

Out[25]:  'Cherry'
```

```
In [130…  list1=[[[[[[[[['Orange']]]]]]]]]
          list1[0][0][0][0][0][0][0][0][0]

Out[130…  'Orange'
```

```
In [146…  list1=[[[[[[[['Fruites',[[['Banana']]]]]]]]]]]
          list1[0][0][0][0][0][0][0][1][0][0][0]

Out[146…  'Banana'
```

slice

```
In [143…: list1=[10,20,30,40,'apple']
          list1[0:5:2] # start,stop,step values are given in slice
```

```
Out[143…: [10, 30, 'apple']
```

mutable and immutable

- mutuable concept based on index
- list are mutable we can change the values uisng index

```
In [2]: l=[1,2,3,4]
        l[0]=100
        l
```

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

concatenation

```
In [3]: l=[1,2,3,4]
        l1=[2,3,5,6]
        l+l1
```

```
Out[3]: [1, 2, 3, 4, 2, 3, 5, 6]
```

```
In [8]: l=[1,2,3,4,True]
        l1=['sruthi','raju']
        l+l1
```

```
Out[8]: [1, 2, 3, 4, True, 'sruthi', 'raju']
```

```
In [ ]: l-l1 #fail
        l*l1 #fail
        l/l1 #fail
```

```
In [9]: l*2  #repeats 2 times
```

```
Out[9]: [1, 2, 3, 4, True, 1, 2, 3, 4, True]
```

```
In [ ]: #l1=[1,2,3]
        #l2=[10,20,30]
        #o=[11,22,33]


        #l1=[1,2,3]
        #l2=[10,20,30,40]
        #o=[11,22,33,40]

        #l1=[1,2,3,4]
        #l2=[10,20,30]
        #0=[11,22,33,4]
```

methods

```
In [18]: 3*1**3

Out[18]: 3

In [19]: 3**3

Out[19]: 27

In [10]: dir(list)

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

append

- append means adding numbers to the end of the list
- append is a basic method we will use many times in our code
- it used to store the result

In [12]:
```
number=['one','Two','Three','Four']
number['Four']
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[12], line 2
      1 number=['one','Two','Three','Four']
----> 2 number['Four']

TypeError: list indices must be integers or slices, not str
```

In [13]:
```
number[3]
```

Out[13]:
```
'Four'
```

In [17]:
```
l=[1,2,3]
l.append(100)      # append is used to save the results
l
```

Out[17]:
```
[1, 2, 3, 100]
```

In [21]:
```
l=[]
l.append(100)
l.append('sruthi')
l
```

Out[21]:
```
[100, 'sruthi']
```

In [22]:
```
l=['apple']
l.append(100)
l.append('sruthi')
l
```

Out[22]:
```
['apple', 100, 'sruthi']
```

In [26]:
```
#q1) write a program ask the user print 1 to 5 number square
# save the result in a list
```

```
for i in range(1,6):
    print(f"{i} square is {i*i}")



l=[]
for i in range(1,6):
    l.append(i*i)
```

```
1 square is 1
2 square is 4
3 square is 9
4 square is 16
5 square is 25
```

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

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

In [51]:
```
#q2) l=['Hyd','blr','Chennai','pune']
#output=['Hyd','Chennai']
l=['Hyd','blr','Chennai','pune']
l1=[]
for i in l:
    if i.istitle():
        l1.append(i)
l1
```

Out[51]:
```
['Hyd', 'Chennai']
```

In [50]:
```
#q3) l=['HYD','bal','PUNE',chennai']

l=['HYD','bal','PUNE','chennai']
l1=[]
for i in l:
    if i.isupper():
        l1.append(i)
l1
```

Out[50]:
```
['HYD', 'PUNE']
```

In [56]:
```
#q4)l=['HYD','bal#','PUNE',che#nnai']
# out:bal#,che#nnai

l=['HYD','bal#','PUNE','che#nnai']
l1=[]
for i in l:
    if '#' in i:
        l1.append(i)
l1
```

Out[56]:
```
['bal#', 'che#nnai']
```

In [62]:
```
#Q5)l=['HYD','bal','PUNE',chennai']

l=['HYD','bal','PUNE','chennai']
l1=[]
for i in l:
    if len(i)>4:
        l1.append(i)
l1
```

Out[62]:
```
['chennai']
```

In [70]:
```
 #Q6) can canner can not a can but canner can you make a can
# find out the each word how many times repeated

s='can canner can not a can but canner can you make a can'
l=[]
for i in l:
    if str(i)=='can canner can not a can but canner can you make a can'
    l.append(i)
```

Out[70]:
```
'can canner can not a can but canner can you make a can'
```

In [141…
```
str1='can canner can not a can but canner can you make a can'
l=[]
m=str1.split()
i=0
for i in range(len(str1)):
    if i in m and i not in l:
        l.append(i)
        i=1+1

print(l)
```

```
[]
```

In [102…
```
#Q7) wap ask the get the even odd numbers
#   you want 5 numbers(random)
#   evenlist and odd list
#   append the even number in  even list
#   append the odd number in odd list

import random
evenlist=[]
oddlist=[]
for i in range(5):
    num=random.randint(1,6)
    if num%2==0:
        evenlist.append(num)
    else:
        oddlist.append(num)
print(evenlist)
print(oddlist)
```

```
[6, 2]
[5, 5, 1]
```

In [ ]:
```
l=['apple' ,'cat', 'dog', 'ball']
#sort it without using sorting

l=[10,15,2,25,89]
#maximum value

l=[1,2,3,4,]
l2=[10,20,30,40]
#output:[11,22,33,44]

l1=[1,2,3,4,5]
l2=[10,20,30,40]
#output[11,22,33,44,5]

s='hello how are you im good at python in naresh it'
find the maximum length word
minimum length word
repeated words
```

In [140…
```
l1=[1,2,3,4,5]
l2=[10,20,30,40]
l3=[]
for i in l1:
    i=0
    for j in l2:
        if i==j:
            i=i+1
            a=l1[i]+l2[i]
        print(a,end='')
```

```
11111111111111111111
```

In [115…
```
ord('a'),chr(97)
```

Out[115…
```
(97, 'a')
```

In [129…
```
l=['apple' ,'cat', 'dog', 'ball']

#sort it without using sorting
l.sort()
l
```

Out[129…
```
['apple', 'ball', 'cat', 'dog']
```

In [110…
```
s='hello how are you im good at python in naresh it'
l=s.split()
a=len(s)
b=len(l)
for i in l:
    for j in s:
        if a[i]>b[j]:
          print(i)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[110], line 3
      1 s='hello how are you im good at python in naresh it'
      2 l=s.split()
----> 3 a=len(s)
      4 b=len(l)
      5 for i in l:

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

list comprehensions

In [102…

```
l=[]
for i in range(1,6):
    l.append(i*i)
```

In [ ]:

```
**pattern 1**

**only for loop**
#syntax
l=[<output> <for loop>]
```

In [4]:

```
l=[i*i for i in range(1,6)]   #squares of 5 numbers
l
```

Out[4]:

```
[1, 4, 9, 16, 25]
```

In [10]:

```
# wap ask the user say 5 times hello save hello in list use list comprehension

l=['Hello' for i in range(5)]
l
```

Out[10]:

```
['Hello', 'Hello', 'Hello', 'Hello', 'Hello']
```

In [ ]:

```
**For loop with if condition**

[<output>  <forloop>  <if condition>]
```

In [14]:

```
#q2) l=['Hyd','blr','Chennai','pune']
#output=['Hyd','Chennai']


out=[i for i in l if i.istitle()]
out
```

Out[14]:

```
['Hyd', 'Chennai']
```

In [19]:

```
#q3) l=['HYD','bal','PUNE',chennai']

l=['HYD','bal','PUNE','chennai']

output=[i for i in l if i.isupper()]
output
```

Out[19]:

```
['HYD', 'PUNE']
```

```
In [21]: #Q5)l=['HYD','bal','PUNE',chennai']

         l=['HYD','bal','PUNE','chennai']


         l=[i for i in l if len(i)>4]
         l

Out[21]: ['chennai']
```

```
In [23]: #q4)l=['HYD','bal#','PUNE',che#nnai']
         # out:bal#,che#nnai

         l=['HYD','bal#','PUNE','che#nnai']


         output=[i for i in l if '#' in i]
         output

Out[23]: ['bal#', 'che#nnai']
```

```
In [ ]: **for-if-else**

        l=[<if_output>  <if condition>   <esle > <else_output> <for loop>]
```

```
In [25]: output=[f"even{i}" if i%2==0   else f"odd{i}"  for i in range(1,11)]
         output

Out[25]: ['odd1',
          'even2',
          'odd3',
          'even4',
          'odd5',
          'even6',
          'odd7',
          'even8',
          'odd9',
          'even10']
```

uniuqe vowel wrapper

```
In [40]: str1='can canner can not a can but canner can you make a can'
         l=str1.split()
         l1=[]
         for i in l:
             if i in l and i not in l1:
                 l1.append(i)

         l1

Out[40]: ['can', 'canner', 'not', 'a', 'but', 'you', 'make']
```

```
In [47]: #by using count method
         str1='can canner can not a can but canner can you make a can'
         l=str1.split()
         l1=[]
         for i in l:
             if  i not in l1:
```

```
        print(i,l.count(i))
        l1.append(i)
```

```
can 5
canner 2
not 1
a 2
but 1
you 1
make 1
```

extend

In [51]:
```
l1=[1,2,3]
l2=['a','b','c']
l2.extend(l1)  #[1,2,3,a,b,c]
l2.append(l1)  #[1,2,3,[a,b,c]]
l1+l2          #[1,2,3,a,b,c]
```

difference between extend and concatenation

- extend will overide the output
- concatenation will same as it is

In [59]:
```
l=[i for i in range(10,20,3)]
a=l.index(13)
l.insert(a+1,'apple')
l
```

Out[59]:
```
[10, 13, 'apple', 16, 19]
```

In [ ]:
```
**pop**
```

In [70]:
```
l=[1,2,3,4,5]
a=l.index(2)
l.pop(a)
l
```

Out[70]:
```
[1, 3, 4, 5]
```

In [60]:
```
l=[1,2,3,4]
l.pop()
```

Out[60]:
```
4
```

In [ ]:
```
**remove**
```

In [73]:
```
l=[1,2,3,'apple']
l.remove(3)
l
```

Out[73]:
```
[1, 2, 'apple']
```

In [ ]:
```
- append
- extend

- insert

- clear/copy

- pop/remove

- sorted/reverse

- index/count
```

In [1]:
```
dir([])
```

```
Out[1]: ['__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 [6]: l=[1,2,3,4]
        l.reverse()
        l
```

```
Out[6]: [4, 3, 2, 1]
```

```
In [7]: l=[1,22,55,74,5,6]
        l.sort()
        l
```

```
Out[7]:  [1, 5, 6, 22, 55, 74]
```

```
In [12]: val=[1,2,3,4,1,5,1]
         val.sort()
         val
```

```
Out[12]: [1, 1, 1, 2, 3, 4, 5]
```

- keywords are generic
- sorted keywords and reverse can use for list and string
- every data type has its own methods
- whenever you work on which datatype you need to use those methods only

del

```
In [16]: l=[1,2,3,4]
         del(l[0])
         l
```

```
Out[16]: [2, 3, 4]
```

```
In [17]: l=[1,5,2,6]
         del(l[1])
         l
```

```
Out[17]: [1, 2, 6]
```

```
In [ ]:  del(l)   # all the items are deleted
```

zip

- zip is the iteratior
- use for loop
- we need two variables(i,j)
- we can add two variables

```
In [20]: l1=[1,2,3]
         l2=[10,20,30]
         out=[]
         # out=[11,22,33]
         for i in range(len(l1)):
             out.append(l1[i]+l2[i])
         out
```

```
Out[20]: [11, 22, 33]
```

```
In [26]: l1=[1,2,3]
         l2=[10,20,30]
         for i,j in zip(l1,l2):
             print(i,j)
```

```
1 10
2 20
3 30
```

In [27]:
```python
out=[i+j for i,j in zip(l1,l2)]  #list comprehension
out
```

Out[27]:
```
[11, 22, 33]
```

In [25]:
```python
l1=[1,2,3]
l2=[10,20,30]
for i,j in zip(l1,l2):
    print(i+j)
```

```
11
22
33
```

In [24]:
```python
l1=['apple','ball','cat']
l2=[1,2,3]
for i,j in zip(l1,l2):
    print(i,j)
```

```
apple 1
ball 2
cat 3
```

In [28]:
```python
dir(tuple)
```

```
Out[28]: ['__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 [ ]:
```