List & Tuple

List

* Data type = int,float,complex,bool,str -> datatype : user can pass only 1 element which may be(int,float,complex,bool,str * Data Structures : collection of data types ->In Data Structures User passes one or more than one datatype (datatype : int,float,complex,bool,str) * Matrix is collection of data structers with rows and columns

In [1]:
```
i = []
i
```

Out[1]: []

In [3]:
```
type(i)
```

Out[3]: list

In [5]:
```
print(type(i))
```

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

In [7]:
```
l = [] #empty list
l
```

Out[7]: []

In [9]:
```
l1 = [1,2,3] # integer list
l1
```

Out[9]: [1, 2, 3]

In [11]:
```
len(l1)
```

Out[11]: 3

In [13]:
```
l2 = [1.1,2.1,3.1,4.1,5.1] #float list
l2
```

Out[13]: [1.1, 2.1, 3.1, 4.1, 5.1]

In [15]:
```
len(l2)
```

Out[15]: 5

In [17]:
```
l3 = ['a','b','c','d'] #string list
l3
```

Out[17]: ['a', 'b', 'c', 'd']

In [19]:
```
for i in l3:
    print(i)
```

```
a
b
c
d
```

In [21]:
```
for i in enumerate(l3): #enumerate built-in function
    print(i)
```

```
(0, 'a')
(1, 'b')
(2, 'c')
(3, 'd')
```

In [23]:
```
l4 = [True,False,None] #boolean list
l4
```

Out[23]: [True, False, None]

In [25]:
```
l5 = [1+2j, 3+4j, 5+6j] #complex list
l5
```

Out[25]: [(1+2j), (3+4j), (5+6j)]

In [27]:
```
l6 = [1,2.1,True,'Yes',1+2j] #Mixed Datatype list
l6
```

Out[27]: [1, 2.1, True, 'Yes', (1+2j)]

In [29]:
```
#for functions use l.(Tab button) you can see all the functions
```

In [31]:
```
#Append function
```

In [33]:
```
l.append(10)
l
```

Out[33]: [10]

In [35]:
```
l.append(20,30) #list.append() takes exactly one argument
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 1
----> 1 l.append(20,30)

TypeError: list.append() takes exactly one argument (2 given)
```

In [39]:
```
l.append(2.3)
l.append('nit')
l.append(1+2j)
l.append(True)
```

In [41]:
```
l
```

Out[41]: [10, 2.3, 'nit', (1+2j), True, 2.3, 'nit', (1+2j), True]

In [43]:
```
l.append(10)
l
```

Out[43]: [10, 2.3, 'nit', (1+2j), True, 2.3, 'nit', (1+2j), True, 10]

In [45]:
```
print(l)
print(l1)
print(l2)
print(l3)
print(l4)
print(l5)
print(l6)
```

```
[10, 2.3, 'nit', (1+2j), True, 2.3, 'nit', (1+2j), True, 10]
[1, 2, 3]
[1.1, 2.1, 3.1, 4.1, 5.1]
['a', 'b', 'c', 'd']
[True, False, None]
[(1+2j), (3+4j), (5+6j)]
[1, 2.1, True, 'Yes', (1+2j)]
```

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

Out[47]: [10, 2.3, 'nit', (1+2j), True, 2.3, 'nit', (1+2j), True, 10]

In [49]:
```
len(l)
```

Out[49]: 10

In [51]:
```
l6.clear() #Remove all items from list.
```

In [53]:
```
l6
```

Out[53]: []

In [55]:
```
l.count(10)
```

Out[55]: 2

In [57]:
```
del l6
l6
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[57], line 2
      1 del l6
----> 2 l6

NameError: name 'l6' is not defined
```

In [59]:
```
l
```

Out[59]: [10, 2.3, 'nit', (1+2j), True, 2.3, 'nit', (1+2j), True, 10]

In [61]:
```
#slicing
```

In [63]:
```
l[:]
```

Out[63]: [10, 2.3, 'nit', (1+2j), True, 2.3, 'nit', (1+2j), True, 10]

In [65]:
```
l[0]
```

Out[65]: 10

In [ ]:
```
#indexing
->is divided by two types
1.Forward Indexing : Left to right
                     start with zero
2.Backward Indexing
```

In [69]:
```
l[-1]
```

Out[69]: 10

In [71]:
```
l[2]
```

Out[71]: 'nit'

In [73]:
```
l[2][0]
```

Out[73]: 'n'

In [75]:
```
print(l[2][0])
print(l[2][1])
print(l[2][2])
```

```
n
i
t
```

In [77]:
```
l
```

Out[77]: [10, 2.3, 'nit', (1+2j), True, 2.3, 'nit', (1+2j), True, 10]

In [79]:
```
l[0:5]
```

Out[79]: [10, 2.3, 'nit', (1+2j), True]

In [81]:
```
l[0:4]
```

Out[81]: [10, 2.3, 'nit', (1+2j)]

In [83]:
```
l[1:5]
```

Out[83]: [2.3, 'nit', (1+2j), True]

In [85]:
```
l[0:10]
```

Out[85]: [10, 2.3, 'nit', (1+2j), True, 2.3, 'nit', (1+2j), True, 10]

In [87]:
```
l[0:6:3]
```

Out[87]: [10, (1+2j)]

In [89]:
```
l[0:-2]
```

Out[89]: [10, 2.3, 'nit', (1+2j), True, 2.3, 'nit', (1+2j)]

In [91]:
```
l
```

Out[91]: [10, 2.3, 'nit', (1+2j), True, 2.3, 'nit', (1+2j), True, 10]

In [93]:
```
for j in l:
    print(j)
```

```
10
2.3
nit
(1+2j)
True
2.3
nit
(1+2j)
True
10
```

List Functions

In [96]:
```
l = []
```

In [98]:
```
type(l)
```

Out[98]: list

In [100]:
```
l_ = list()
```

In [102]:
```
type(l_)
```

Out[102]: list

In [104]:
```
l
```

Out[104]: []

append function

In [107]:
```
l.append(10) #append function is used to add the elements into the list
l
```

Out[107]: [10]

In [109]:
```
l
```

Out[109]: [10]

In [111]:
```
l.append(20)
l.append(2.2)
l.append(True)
```

```
l.append('nit')
l.append(1+2j)
```

In [113]:
```
l
```

Out[113]:
```
[10, 20, 2.2, True, 'nit', (1+2j)]
```

In [115]:
```
l.append(10)
```

In [117]:
```
l
```

Out[117]:
```
[10, 20, 2.2, True, 'nit', (1+2j), 10]
```

count function

In [122]:
```
l.count(10) #count is used to know how many times the number repeated
```

Out[122]:
```
2
```

In [124]:
```
l.count(20)
```

Out[124]:
```
1
```

In [126]:
```
l
```

Out[126]:
```
[10, 20, 2.2, True, 'nit', (1+2j), 10]
```

In [128]:
```
len(l)
```

Out[128]:
```
7
```

copy function

In [131]:
```
l1 = l.copy() #copy function is used to copy from one list to another
```

In [133]:
```
l1 #Here we coppied the datatypes from the list 'l' to the list 'l1'
```

Out[133]:
```
[10, 20, 2.2, True, 'nit', (1+2j), 10]
```

In [135]:
```
l == l1 # we are checking if the both lists are equal
```

Out[135]:
```
True
```

In [137]:
```
l
```

Out[137]:
```
[10, 20, 2.2, True, 'nit', (1+2j), 10]
```

remove function

In [140]:
```
l.remove(1+2j) # we use remove to remove specific datatypes from the list
               #it removes from the starting datatype from left to right
```

In [142]:
```
l
```

Out[142]: [10, 20, 2.2, True, 'nit', 10]

In [144]:
```
l1.remove(10)
l1
```

Out[144]: [20, 2.2, True, 'nit', (1+2j), 10]

In [146]:
```
l
```

Out[146]: [10, 20, 2.2, True, 'nit', 10]

In [148]:
```
l1
```

Out[148]: [20, 2.2, True, 'nit', (1+2j), 10]

pop function

In [153]:
```
l.pop() #pop is used to remove the ekements from the end i.e from the right side t
```

Out[153]: 10

In [ ]:
```
l
```

In [155]:
```
l1 #here's the last element is 10 and that will be removed by pop function
```

Out[155]: [20, 2.2, True, 'nit', (1+2j), 10]

In [157]:
```
l1.pop()
```

Out[157]: 10

In [159]:
```
l1
```

Out[159]: [20, 2.2, True, 'nit', (1+2j)]

In [161]:
```
l1.pop(1) # to pop a specific element we specify index number which is starting fr
          #here in list l1 0 = 20,1 = 2.2,...
```

Out[161]: 2.2

In [163]:
```
l
```

Out[163]: [10, 20, 2.2, True, 'nit']

slicing

In [166]:
```
l[:] #slicing
```

Out[166]: [10, 20, 2.2, True, 'nit']

In [168]:
```
l[1:] #prints all the elements from the first index
```

Out[168]: [20, 2.2, True, 'nit']

In [170]:
```
l[:5]  # print the elemnt till 5th indext ( 5th index is at last ) n =n-1
       #here in list'l' theres no index 5 so it prints the list
```

Out[170]: [10, 20, 2.2, True, 'nit']

In [172]:
```
l
```

Out[172]: [10, 20, 2.2, True, 'nit']

In [174]:
```
l[1:5] # it prints from the 1st index  to 5th index
```

Out[174]: [20, 2.2, True, 'nit']

In [176]:
```
l
```

Out[176]: [10, 20, 2.2, True, 'nit']

In [178]:
```
l[1:-1] #it prints from the index 1 to index -1
        #-1 index is backward indexing where the indexing is from right to left fr
        #here index 1 is 20 and i
```

Out[178]: [20, 2.2, True]

In [180]:
```
l
```

Out[180]: [10, 20, 2.2, True, 'nit']

In [182]:
```
l1
```

Out[182]: [20, True, 'nit', (1+2j)]

In [184]:
```
l1[1]
```

Out[184]: True

In [186]:
```
l1[1] = 1000 #we are updating the current datatype with index1  to 1000
```

In [188]:
```
l1
```

Out[188]: [20, 1000, 'nit', (1+2j)]

In [190]:
```
l
```

Out[190]: [10, 20, 2.2, True, 'nit']

In [192]:
```
l[4] = 'mit'
l
```

Out[192]: [10, 20, 2.2, True, 'mit']

In [194]:
```
l[-1] = 'nit'
l
```

Out[194]: [10, 20, 2.2, True, 'nit']

index function

In [197]:
```
l.index(10)
```

Out[197]: 0

In [199]:
```
l.index(True)
```

Out[199]: 3

In [201]:
```
l1
```

Out[201]: [20, 1000, 'nit', (1+2j)]

In [203]:
```
l
```

Out[203]: [10, 20, 2.2, True, 'nit']

extend function

In [208]:
```
l.extend(l1) #combines l and l1 into a single list
```

In [210]:
```
l
```

Out[210]: [10, 20, 2.2, True, 'nit', 20, 1000, 'nit', (1+2j)]

In [212]:
```
len(l)
```

Out[212]: 9

In [214]:
```
l1
```

Out[214]: [20, 1000, 'nit', (1+2j)]

insert function

In [217]:
```
l1.insert(1,'nit') #insert is used to add elements to the specific index
l1
```

Out[217]: [20, 'nit', 1000, 'nit', (1+2j)]

In [219]:
```
l1.insert(5,'mit')
l1
```

Out[219]: [20, 'nit', 1000, 'nit', (1+2j), 'mit']

In [221]:
```
l3 = [100, 4, 76, 3, 9, 200]
```

reverse function

In [224]:
```
l3.reverse() #reverse function is used to print the values in the reverse order
l3
```

Out[224]: [200, 9, 3, 76, 4, 100]

sort function

In [229]:
```
l3.sort() #sort function is used to arrange the datatypes in ascending order
l3
```

Out[229]: [3, 4, 9, 76, 100, 200]

In [231]:
```
l3.sort(reverse=True) #using sort function to print datatypes in descending order
l3
```

Out[231]: [200, 100, 76, 9, 4, 3]

In [233]:
```
l4 = [1,2,'nit','z',3.3]
l4
```

Out[233]: [1, 2, 'nit', 'z', 3.3]

In [235]:
```
l4.sort() #we cannot sort integer and string datatypes at the same time
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[235], line 1
----> 1 l4.sort()

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

In [237]:
```
l5 = ['z', 'm', 'a', 'o']
l5.sort()
l5
```

Out[237]: ['a', 'm', 'o', 'z']

In [239]:
```
l = [10, 20, 2.2, True, 'nit']
```

In [241]:
```
l3 = [200, 100, 76, 9, 4, 3]
```

In [243]:
```
l4 = [1, 2, 'nit', 'z', 3.3]
```

In [245]:
```
l5 = ['a', 'm', 'o', 'z']
```

In [247]:
```
print(l)
print(l3)
print(l4)
print(l5)
```

```
[10, 20, 2.2, True, 'nit']
[200, 100, 76, 9, 4, 3]
[1, 2, 'nit', 'z', 3.3]
['a', 'm', 'o', 'z']
```

In [249]:
```
l
```

Out[249]: [10, 20, 2.2, True, 'nit']

In [251]:
```
10 in l
```

Out[251]: True

In [253]:
```
100 in l
```

Out[253]: False

In [255]:
```
l
```

Out[255]: [10, 20, 2.2, True, 'nit']

In [257]:
```
for i in l:
    print(i)
```

```
10
20
2.2
True
nit
```

In [259]:
```
for i in enumerate(l):
    print(i)
```

```
(0, 10)
(1, 20)
(2, 2.2)
(3, True)
(4, 'nit')
```

In [261]:
```
l
```

Out[261]: [10, 20, 2.2, True, 'nit']

In [263]:
```
l3
```

Out[263]: [200, 100, 76, 9, 4, 3]

In [265]:
```
l6 = l + l3
l6
```

Out[265]: [10, 20, 2.2, True, 'nit', 200, 100, 76, 9, 4, 3]

Tuples

1.Tuple is immutable whereas list is mutable 2.Immutable means we can't change the elements once we assign 3.When we want to store data that should not be changed, a tuple is the best choice.This is because once a tuple is created, its contents cannot be modified. 4.Iterating over the elements of a tuple is faster compared to iterating over a list. 5. we use tuple in GPS Coordinates Calendar Dates Database Records Configuration Settings Network Addresses User Profiles

Tuple Creation

In [7]:
```
t1 = ()  # empty tuple
```

In [9]:
```
t2 = (10,30,60) #tuple of integer numbers
```

In [13]:
```
t3 = (10.77,30.66,60.89) #tuple of float numbers
```

In [15]:
```
t4 = ('one','two','three') #tuple of strings
```

In [17]:
```
t5 = ('asif',35,(50,100),(150,90)) #Nested tuples
```

In [19]:
```
t6 = (100,'asif',17.765) #tuple of mixed daratypes
```

In [21]:
```
t7 = ('asif',25,[50,100],[150,90],{'john','david'},(99,22,33))
```

In [23]:
```
len(t7)
```

Out[23]:
```
6
```

Tuple Indexing

In [33]:
```
t2
```

Out[33]:
```
(10, 30, 60)
```

In [29]:
```
t2[0] #gives the 1st element of the tuple necause indexing starts from zero
```

Out[29]:
```
10
```

In [35]:
```
t4
```

Out[35]:
```
('one', 'two', 'three')
```

In [31]:
```
t4[0]
```

Out[31]:
```
'one'
```

In [37]:
```
t4[0][0] #nested indexing - gives the first character of the first element in the
```

Out[37]:
```
'o'
```

In [39]:
```
t4[-1] #gives the last element of tuple because backward indexing starts from -1
```

Out[39]: 'three'

In [41]:
```
t5
```

Out[41]: ('asif', 35, (50, 100), (150, 90))

In [43]:
```
t5[-1]
```

Out[43]: (150, 90)

Tuple Slicing

In [49]:
```
myt = ('one','two','three','four','five','six','seven','eight')
```

In [51]:
```
myt
```

Out[51]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [53]:
```
myt[0:3] #gives the element from the '0' index to '3rd' index
```

Out[53]: ('one', 'two', 'three')

In [55]:
```
myt[2:5] # gives the element from the 2nd index to the 5th index
```

Out[55]: ('three', 'four', 'five')

In [57]:
```
myt[:3] # when we dont give any statrt index it will give from start to the end ir
```

Out[57]: ('one', 'two', 'three')

In [61]:
```
myt[-1] # when theres negative index given we need to start slicing from the backv
```

Out[61]: 'eight'

In [63]:
```
myt[-3]
```

Out[63]: 'six'

In [65]:
```
myt[:]
```

Out[65]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

Remove & Change Items

In [68]:
```
myt
```

Out[68]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [70]:
```
del myt[0] #tuple is immutable we can't DELETE tuple items
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[70], line 1
----> 1 del myt[0]

TypeError: 'tuple' object doesn't support item deletion
```

In [72]:
```
myt[0] = 1 # Tuple is immutable we can't CHANGE tuple items
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[72], line 1
----> 1 myt[0] = 1

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

In [74]:
```
del myt # Deleting the entire tuple is possible
```

Loop through a tuple

In [81]:
```
myt = ('one','two','three','four','five','six','seven','eight')
myt
```

Out[81]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [83]:
```
for i in myt:
    print(i)
```

```
one
two
three
four
five
six
seven
eight
```

In [85]:
```
for i in enumerate(myt):
    print(i)
```

```
(0, 'one')
(1, 'two')
(2, 'three')
(3, 'four')
(4, 'five')
(5, 'six')
(6, 'seven')
(7, 'eight')
```

Count

In [94]:
```
myt1 = ('one','two','three','four','one','one','two','three')
```

In [96]:
```
myt1.count('one')
```

Out[96]: 3

myt1.count('two')

In [100]:
```
myt1.count('four')
```

Out[100]: 1

In [102]:
```
myt1.count('nine') #because there is no nine in the tuple
```

Out[102]: 0

Tuple Membership

In [107]:
```
myt
```

Out[107]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [109]:
```
'one' in myt #checks if 'one' exists in the tuple returns 'True' if exists or 'Fal
```

Out[109]: True

In [111]:
```
'ten' in myt #because there is no element in the tuple
```

Out[111]: False

In [113]:
```
if 'three' in myt:
    print('Three is present in the tuple')
else:
    print('Three is not present in the tuple')
```

Three is present in the tuple

In [115]:
```
if 'eleven' in myt:
    print('eleven is present in the tuple')
else:
    print('eleven is not present in the tuple')
```

eleven is not present in the tuple

Index Position

In [118]:
```
myt
```

Out[118]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [120]:
```
myt.index('one')
```

Out[120]: 0

In [122]:
```
myt.index('five')
```

Out[122]: 4

In [124]:
```
myt1
```

Out[124]:
```
('one', 'two', 'three', 'four', 'one', 'one', 'two', 'three')
```

In [126]:
```
myt1.index('one')
```

Out[126]:
```
0
```

Sorting

In [137]:
```
myt2 = (43,67,99,12,6,90,67)
```

In [139]:
```
sorted(myt2) # returns the tuple with sorted order i.e Ascending order
```

Out[139]:
```
[6, 12, 43, 67, 67, 90, 99]
```

In [141]:
```
sorted(myt2,reverse=True)# returns the tuple with sorted order i.e Descending orde
```

Out[141]:
```
[99, 90, 67, 67, 43, 12, 6]
```

In [145]:
```
#Tuple is completed
```