tuple ¶

In [ ]:

```
Properties  :
    1. immutable  (can not change items once tuple is created )
    2. ordered
    3. duplicates allowed
    4. enclosed by ( )
    5. seperated by commas  ,
    6. there might be items with different Data Types

```

In [ ]:

```
Tuple Functions :
    1. index()
    2. count()


```

In [2]:

```
list1=[10,50,60,30,90]
print(list1)
list1[2]=333
print(list1)
```

```
[10, 50, 60, 30, 90]
[10, 50, 333, 30, 90]
```

Type of tuple

In [5]:

```
tuple1=(30,50,70,90,110,130)
print(type(tuple1))

```

```
<class 'tuple'>
```

length of Tuple

In [8]:

```python
#length of tuple
t1=(30,50,70.65,90,'Python',1+8j,[9,3,4,5],(7,6,9))
print(len(t1))
print(t1)
```

```
8
(30, 50, 70.65, 90, 'Python', (1+8j), [9, 3, 4, 5], (7, 6, 9))
```

In [9]:

```python
# duplicates allowed
t2=(30,'Python',70.65,90,'Python',1+8j,[9,3,4,5],(7,6,9),1+8j,90,90)
print(len(t2))
print(t2)
```

```
11
(30, 'Python', 70.65, 90, 'Python', (1+8j), [9, 3, 4, 5], (7, 6, 9), (1+8
j), 90, 90)
```

Accessing Items in Tuple

In [14]:

```python
t2=(30,'Python',70.65,90,'Python')
print(t2[0])
print(t2[1])
print(t2[2])
print(t2[3])
print(t2[4])
```

```
30
Python
70.65
90
Python
```

In [ ]:

```python

```

Tuple is immutable (not changeable)

In [11]:

```python
t2=(30,'Python',70.65,90,'Python')
t2[1]='Java'
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Cell In[11], line 2
      1 t2=(30,'Python',70.65,90,'Python')
----> 2 t2[1]='Java'

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

Slicing

```
tuple[start_index : end_index : step_size]
step_size : optional
start_index  >> default value  0       >>  inclusive
end_index    >> default len of tuple   >>  exclusive


```

In [17]:

```python
t2=(30,'Python',70.65,90,'Python')

t2[1:3]   #  1  to 2

```

Out[17]:

```
('Python', 70.65)
```

In [18]:

```python
t2[1:4]   # 1 to 3
```

Out[18]:

```
('Python', 70.65, 90)
```

In [19]:

```python
t2[:]
```

Out[19]:

```
(30, 'Python', 70.65, 90, 'Python')
```

In [20]:

```python
t2[:4]   #  0  to 3
```

Out[20]:

```
(30, 'Python', 70.65, 90)
```

In [21]:

```python
t2[2:]   # 2 to end (len-1)
```

Out[21]:

```
(70.65, 90, 'Python')
```

In [28]:

```python
tuple1=(10,20,30,40,50,60,70,80,90,100)

print(tuple1[2 : 8])  #   2 to 7
print(tuple1[2 : 8 : 1])  # step size 1  >> 1-1 =0

print(tuple1[ :  : 1])  # step size 1  >> 1-1 =0
print(tuple1[ :  : 2])  # step size 2  >> 2-1 =1
print(tuple1[ :  : 3])  # step size 3  >> 3-1 =2

```

```
(30, 40, 50, 60, 70, 80)
(30, 40, 50, 60, 70, 80)
(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
(10, 30, 50, 70, 90)
(10, 40, 70, 100)
```

In [30]:

```python
print(tuple1[ :  : -1])  # step size -1  >> -1+1 =0
```

```
(100, 90, 80, 70, 60, 50, 40, 30, 20, 10)
```

In [31]:

```python
print(tuple1[ :  : -2])  # step size -2  >> -2+1 =-1
```

```
(100, 80, 60, 40, 20)
```

In [33]:

```python
tup1=(10,20,30,40,50)
print("Original Tuple : ", tup1)
tup2=tup1[::-1]
print("New Tuple with Reversed items :  ", tup2)

```

```
Original Tuple :  (10, 20, 30, 40, 50)
New Tuple with Reversed items :   (50, 40, 30, 20, 10)
```

Tuple to list

In [40]:

```
t1=(20,40,60,80,100)
print(t1,type(t1))

l1=t1
print(l1,type(l1))
```

```
(20, 40, 60, 80, 100) <class 'tuple'>
(20, 40, 60, 80, 100) <class 'tuple'>
```

In [41]:

```
t1=(20,40,60,80,100)
print(t1,type(t1))

l1=list(t1)  # tuple t1 is converted to list by type casting
print(l1,type(l1))
```

```
(20, 40, 60, 80, 100) <class 'tuple'>
[20, 40, 60, 80, 100] <class 'list'>
```

In [38]:

```
a=100
b=float(a)
type(b)
```

Out[38]:

```
float
```

In [43]:

```
t1=(20,40,60,80,100)
print(t1,type(t1))

l1=list(t1)  # tuple t1 is converted to list by type casting
print(l1,type(l1))

l1.append(200)
print(l1,type(l1))

l1.extend([300,400,500])
print(l1,type(l1))
```

```
(20, 40, 60, 80, 100) <class 'tuple'>
[20, 40, 60, 80, 100] <class 'list'>
[20, 40, 60, 80, 100, 200] <class 'list'>
[20, 40, 60, 80, 100, 200, 300, 400, 500] <class 'list'>
```

list to tuple

In [45]:

```
newtuple=tuple(l1)
newtuple
```

Out[45]:

```
(20, 40, 60, 80, 100, 200, 300, 400, 500)
```

In [48]:

```
subject_list=['C','CPP','Python','Java']
print(subject_list, type(subject_list))

subject_tuple=tuple(subject_list)
print(subject_tuple, type(subject_tuple))

```

```
['C', 'CPP', 'Python', 'Java'] <class 'list'>
('C', 'CPP', 'Python', 'Java') <class 'tuple'>
```

In [ ]:

```

```

String to list

In [53]:

```
string1="Python and Machine Learing Training by Shri Software"
my_list=string1.split()
print(my_list)

```

```
['Python', 'and', 'Machine', 'Learing', 'Training', 'by', 'Shri', 'Softwa
re']
```

In [55]:

```
string1="Python and Machine Learing"
my_list=list(string1)
print(my_list)
```

```
['P', 'y', 't', 'h', 'o', 'n', ' ', 'a', 'n', 'd', ' ', 'M', 'a', 'c',
'h', 'i', 'n', 'e', ' ', 'L', 'e', 'a', 'r', 'i', 'n', 'g']
```

list to string

In [57]:

```
list1=['P', 'y', 't', 'h', 'o', 'n']
str1=str(list1)  # This is not right
str1
```

Out[57]:

```
"['P', 'y', 't', 'h', 'o', 'n']"
```

list to string using join function

In [59]:

```
list1=['P', 'y', 't', 'h', 'o', 'n']
string2=''.join(list1)
string2
```

Out[59]:

```
'Python'
```

In [60]:

```
list1=['P', 'y', 't', 'h', 'o', 'n']
string2=''.join(list1)
string2
```

Out[60]:

```
'P@y@t@h@o@n'
```

In [62]:

```
list1=['P', 'y', 't', 'h', 'o', 'n']
string2='  '.join(list1)
string2
```

Out[62]:

```
'P  y  t  h  o  n'
```

In [68]:

```
str1="Python"
str2="Training"
str1+str2
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Cell In[68], line 4
      2 str2="Training"
      3 str1+str2
----> 4 str1-str2

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

In [66]:

```
str1="Python"
str2="Training"
str1*str2
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Cell In[66], line 3
      1 str1="Python"
      2 str2="Training"
----> 3 str1*str2

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

In [67]:

```
str1="Python"
str1*5
```

Out[67]:

```
'PythonPythonPythonPythonPython'
```

Delete Items in Tuple

In [70]:

```
t1=('Python', 'and', 'Machine', 'Learing', 'Training')
print(t1)
print(type(t1))
```

```
('Python', 'and', 'Machine', 'Learing', 'Training')
<class 'tuple'>
```

In [71]:

```
t1.remove('and')
```

```
---------------------------------------------------------------------------
--
AttributeError                            Traceback (most recent call las
t)
Cell In[71], line 1
----> 1 t1.remove('and')

AttributeError: 'tuple' object has no attribute 'remove'
```

In [72]:

```
del t1[1]
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Cell In[72], line 1
----> 1 del t1[1]

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

In [73]:

```
del t1
```

In [74]:

```
t1
```

```
---------------------------------------------------------------------------
--
NameError                                 Traceback (most recent call las
t)
Cell In[74], line 1
----> 1 t1

NameError: name 't1' is not defined
```

In [77]:

```python
t1=('Python', 'and', 'Machine', 'Learing', 'Training')
print(t1)
print(type(t1))

l1=list(t1)
l1.remove('and')

t1=tuple(l1)
print(t1)
print(type(t1))

```

```
('Python', 'and', 'Machine', 'Learing', 'Training')
<class 'tuple'>
('Python', 'Machine', 'Learing', 'Training')
<class 'tuple'>
```

Tuple Functions

In [ ]:

```
1. index()
2. count()


```

1.index()

In [78]:

```python
t1=(20, 40, 60, 80, 100, 200, 300, 400, 500)
t1.index(100)
```

Out[78]:

```
4
```

In [79]:

```python
t1.index(400)
```

Out[79]:

```
7
```

In [80]:

```
t1.index(900)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[80], line 1
----> 1 t1.index(900)

ValueError: tuple.index(x): x not in tuple
```

2.count()

In [85]:

```
t1=(20, 40, 100, 80, 100, 200, 100, 400, 200)
print(t1.count(100))
print(t1.count(200))
print(t1.count(400))
print(t1.count(900))
```

```
3
2
1
0
```

Accessing Tuple items using for loop

In [86]:

```
t1=(20, 40, 100, 80, 100, 200, 100, 400, 200)
for item in t1:
    print(item)
```

```
20
40
100
80
100
200
100
400
200
```

In [87]:

```
subject_tuple=('C','CPP','Python','Java')
for subject in subject_tuple:
    print(subject)
```

```
C
CPP
Python
Java
```

In [89]:

```
subject_tuple=('C','CPP','Python','Java')
index=0
for subject in subject_tuple:
    print(f"Subject at index {index} is {subject}")
    index=index+1
```

```
Subject at index 0 is C
Subject at index 1 is CPP
Subject at index 2 is Python
Subject at index 3 is Java
```

In [93]:

```
subject_tuple=('C','CPP','Python','Java')

for index,subject in enumerate(subject_tuple):
    print(f"Subject at index {index} is {subject}")
    print('-'*70)

```

```
Subject at index 0 is C
----------------------------------------------------------------------
Subject at index 1 is CPP
----------------------------------------------------------------------
Subject at index 2 is Python
----------------------------------------------------------------------
Subject at index 3 is Java
----------------------------------------------------------------------
```

Copy

In [95]:

```
t1=('C','CPP','Python','Java')
t2=t1
print(t1,id(t1))
print(t2,id(t2))
```

```
('C', 'CPP', 'Python', 'Java') 1854365447440
('C', 'CPP', 'Python', 'Java') 1854365447440
```

```
there is no copy function  in tuple data type
```

```
for that we can use deep copy concept
```

deep copy

In [103]:

```python
import copy
tup1=('C','CPP','Python',[7,9,10],'Java')
tup2=copy.deepcopy(tup1)
print(tup1,id(tup1))
print(tup2,id(tup2))

tup1[3][1]=999
print(tup1,id(tup1))
print(tup2,id(tup2))



```

```
('C', 'CPP', 'Python', [7, 9, 10], 'Java') 1854365022016
('C', 'CPP', 'Python', [7, 9, 10], 'Java') 1854365022096
('C', 'CPP', 'Python', [7, 999, 10], 'Java') 1854365022016
('C', 'CPP', 'Python', [7, 9, 10], 'Java') 1854365022096
```

multiply tuple / list with integer

In [107]:

```python
t1=('C','CPP','Python','Java')
print(t1*5)

l1=[10,20,30,40,50]
print(l1*5)
```

```
('C', 'CPP', 'Python', 'Java', 'C', 'CPP', 'Python', 'Java', 'C', 'CPP',
'Python', 'Java', 'C', 'CPP', 'Python', 'Java', 'C', 'CPP', 'Python', 'Ja
va')
[10, 20, 30, 40, 50, 10, 20, 30, 40, 50, 10, 20, 30, 40, 50, 10, 20, 30,
40, 50, 10, 20, 30, 40, 50]
```

sorted ()

In [114]:

```
t3=(10,2,30,4,50,8)
print(t3,type(t3))

t3_new= sorted(t3) # it will return list always
print(t3_new,type(t3_new))

print("Original Tupel : " , t3)
print("Sorted  Tupel Ascending  : " , t3_new)

```

```
(10, 2, 30, 4, 50, 8) <class 'tuple'>
[2, 4, 8, 10, 30, 50] <class 'list'>
Original Tupel :  (10, 2, 30, 4, 50, 8)
Sorted  Tupel Ascending  :  [2, 4, 8, 10, 30, 50]
```

In [115]:

```
t3_new= sorted(t3, reverse=True) #  Desc
print("Original Tupel : " , t3)
print("Sorted  Tupel Descending  : " , t3_new)
```

```
Original Tupel :  (10, 2, 30, 4, 50, 8)
Sorted  Tupel Descending  :  [50, 30, 10, 8, 4, 2]
```

reversed ()

In [120]:

```
t4=(10,2,30,4,50,8)
t4_rev=tuple(reversed(t4))

print("Original Tupel : " , t4)
print("Reversed Tuple  : " , t4_rev)
```

```
Original Tupel :  (10, 2, 30, 4, 50, 8)
Reversed Tuple  :  (8, 50, 4, 30, 2, 10)
```

In [121]:

```
sum(t4)
```

Out[121]:

```
104
```

In [122]:

```
min(t4)
```

Out[122]:

```
2
```

In [123]:

```
max(t4)
```

Out[123]:

```
50
```

In [ ]:

```

```