Tuple Data type(3 day)

1. Tuple is exactly same as List except that it is immutable. i.e once we creates Tuple object, we cannot perform any changes in that object.
Hence Tuple is Read Only version of List.
2. If our data is fixed and never changes then we should go for Tuple.
3. Insertion Order is preserved
4. Duplicates are allowed
5. Heterogeneous objects are allowed.
6. We can preserve insertion order and we can differentiate duplicate objects by using index. Hence index will play very important role in Tuple also.
Tuple support both +ve and -ve index. +ve index means forward direction(from left to right) and -ve index means backward direction(from right to left)
7. We can represent Tuple elements within **Parenthesis** and with comma seperator.
Parenethesis are optional but recommended to use.

Eg:
```
t=10,20,30,40
print(t)
print(type(t))
```

Output
```
(10, 20, 30, 40)
<class 'tuple'>
```

```
t=()
print(type(t)) # tuple
```

Note: We have to take special care about single valued tuple.compulsary the value should ends with comma,otherwise it is not treated as tuple.
Eg:
```
t=(10)
print(t)
print(type(t))
```

Output
```
10
<class 'int'>
```
Eg:
```
t=(10,)
print(t)
print(type(t))
```

Output
```
(10,)
<class 'tuple'>
```

Q. Which of the following are valid tuples?

1. t=() #empty tuple
2. t=10,20,30,40 #Valid
3. t=10 #int
4. t=10, #tuple
5. t=(10)#int
6. t=(10,)#tuple
7. t=(10,20,30,40)#tuple

Tuple creation:

1. t=()
   creation of empty tuple
2. t=(10,)
   t=10,
   creation of **single valued tuple** ,parenthesis are optional,should ends with comma
3. t=10,20,30
   t=(10,20,30)
   creation of **multi values tuples** & parenthesis are optional
4. By using tuple() function:

```
list=[10,20,30]
t=tuple(list)
print(t)

t=tuple(range(10,20,2))
print(t)
```

Accessing elements of tuple:

We can access either by index or by slice operator

1. By using index:

```
t=(10,20,30,40,50,60)
print(t[0]) #10
print(t[-1]) #60
print(t[100]) IndexError: tuple index out of range
```

2. By using slice operator:

```
t=(10,20,30,40,50,60)
print(t[2:5])  #(30,40,50)
print(t[2:100]) #( 30,40,50,60)
print(t[::2]) #(10,30,50)
```

Output

```
(30, 40, 50)
(30, 40, 50, 60)
```

(10, 30, 50)

Tuple vs immutability:
Once we creates tuple,we cannot change its content.
Hence tuple objects are immutable.

Eg:
t=(10,20,30,40)
t[1]=70 **TypeError:** 'tuple' object does not support item assignment

Mathematical operators for tuple:
We can apply + and * operators for tuple

1. Concatenation Operator(+):
t1=(10,20,30)
t2=(40,50,60)
t3=t1+t2
print(t3) # (10,20,30,40,50,60)

2. Multiplication operator or repetition operator(*)
t1=(10,20,30)
t2=t1*3
print(t2) #(10,20,30,10,20,30,10,20,30)

Important functions of Tuple:
1. len()
To return number of elements present in the tuple
Eg:
t=(10,20,30,40)
print(len(t)) #4
2. count()
To return number of occurrences of given element in the tuple
Eg:
t=(10,20,10,10,20)
print(t.count(10)) #3
3. index()
returns **index of first occurrence of the given element.**
If the specified element is not available then we will get ValueError.
Eg:
t=(10,20,10,10,20)
print(t.index(10)) #0
print(t.index(30)) **ValueError:** tuple.index(x): x not in tuple

4. sorted()

To sort elements based on default natural sorting order

```
t=(40,10,30,20)
t1=sorted(t)
print(t1)
print(t)
```

Output

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

We can sort according to reverse of default natural sorting order as follows

```
t1=sorted(t,reverse=True)
print(t1) [40, 30, 20, 10]
```

5. min() and max() functions:

These functions return min and max values according to default natural sorting order.

Eg:

```
t=(40,10,30,20)
print(min(t)) #10
print(max(t)) #40
```

Tuple Packing and Unpacking:

We can create a tuple by packing a group of variables.

Eg:

```
t=1,2,3,4,5  #tuple packing
print(t)
a,b,c,d,e=t#tuple Unpacking
print(a,b,c,d,e)
a,b,c,d=t#Value Error
print(a,b,c,d,e)
a,b,c,d,e,f=t#Value Error
print(a,b,c,d,e)
```

```
t=1,2,3,4,5  #tuple packing
print(t)
*a,b,c=t#tuple Unpacking
print(tuple(a),b,c)
```

Note:*a---MultiValue Variable

Note: At the time of tuple unpacking the number of variables and number of values

should be same. ,otherwise we will get ValueError.

Eg:

```
t=(10,20,30,40)
a,b,c=t #ValueError: too many values to unpack (expected 3)
```

Tuple Comprehension:

Tuple Comprehension is not supported by Python.

```
 t= ( x**2 for x in range(1,6))
```

Here we are not getting tuple object and we are getting generator object.

```
t= ( x**2 for x in range(1,6))
print(type(t))
for x in t:
    print(x)
```

Output

```
<class 'generator'>
1
4
9
16
25
```

Differences between List and Tuple:

List and Tuple are exactly same except small difference: List objects are mutable where as

Tuple objects are immutable.

In both cases insertion order is preserved, duplicate objects are allowed, heterogenous objects are allowed, index and slicing are supported.

List	Tuple
1) List is a Group of Comma separated Values within **Square Brackets** and Square Brackets are **mandatory.** Eg: i = [10, 20, 30, 40]	1) Tuple is a Group of Comma separeated Values within **Parenthesis** and Parenthesis are **optional.** Eg: t = (10, 20, 30, 40) t = 10, 20, 30, 40
2) List Objects are **Mutable** i.e. once we creates List Object we can perform any changes in that Object. Eg: i[1] = 70	2) Tuple Objects are I**mmutable** i.e. once we creates Tuple Object we cannot change its content. t[1] = 70 ➔ **ValueError**: tuple object does not support item assignment.
3) If the Content is not fixed and keep on changing then we should go for List.	3) If the content is fixed and never changes then we should go for Tuple.
4) List Objects can not used as Keys for Dictionaries because Keys should be Hashable and Immutable.	4) Tuple Objects can be used as Keys for Dictionaries because Keys should be Hashable and Immutable.