Parul® University

Python Programming 05101155

Prof. Bhavika Vaghela, Assistant Professor
Parul Institute of Computer Application - BCA

Parul®
University

DIGITAL LEARNING CONTENT

Unit 5

Tuple and its method

Tuple in Python

- A tuple is a sequence of any type of values (number, string, character, float value or mix).
- Elements of tuple are indexed by integers, so in that respect tuples are a lot like lists.
- The important difference is that tuple is immutable.
- Is used to store the sequence of immutable python objects. Tuple is similar to lists since the value of the items stored in the list can be changed whereas the tuple is immutable and the value of the items stored in the tuple cannot be changed.
- As tuple is immutable mean it cannot be change or update by passing index value also.
- Its index value start with “0”

How to declare tuple?

- To declare tuple we must enclose each and every elements of tuple inside the **parenthesis ().**
- Each element must be **separated by comma “,” delimiter**.
- One can also declare **empty tuple by empty parenthesis** but its simply useless as tuple is immutable so in future it cannot be updated.
- One can also declare tuple is inbuilt function of tuple class **tuple()**
- Again same rule of variable declaration give a tuple name properly which follow each and every rule of variable declaration.

```
Tuple1=(1, 2, 3, 4)
Tuple2=() #empty tuple
type(Tuple2)
<class ‘tuple’>
```

Tuple declaration with single element

If someone want to declare tuple with single element than he or she must have to put comma at the last else it will not treated as tuple.

```
>>> t1 = ('a',)
>>> type(t1)
<type 'tuple'>

>>> tup1=(1,)
>>> type(tup1)
<class 'tuple'>
```

```
>>> t2 = ('a')
>>> type(t2)
<type 'str'>   note : please try it on
your machine.

>>> tup1=(1)
>>> type(tup1)
<class 'int'>
```

Without putting comma at the end it will treat it as string, integer, float etc. you can see it in above example.

Declaration of Tuple cont..

Tuple with multiple element like this

```
T1 = (101, "Ayush", 22)
T2 = ("Apple", "Banana", "Orange")
T3=(10,20,40,34,50,34,56,57)
```

Declaring tuple using built in function tuple() by passing string as argument

```
tup1=tuple('12345')
>>> print(tup1)
('1', '2', '3', '4', '5')
>>> tup1=tuple('wel come to PU')
>>> print(tup1)
('w', 'e', 'l', ' ', 'c', 'o', 'm', 'e', ' ', 't', 'o', ' ', 'P', 'U')
```

Note: in above example white space is also consider as element which is enclosed with ‘’

Accessing element of Tuple

- As we discuss before tuple element have its own index value which is start by 0.
- So to access element of tuple one can pass index value in [ ] bracket.
- It also allow to pass negative index value like string but make sure it should not be float value else it will fire an error.

```
>>>
tuple1=(1,2,3,"xyz","pqr",10.3,45.9)
>>> print(tuple1[3])
xyz
>>> print(tuple1[-2])
10.3
```

```
>>> print(tuple1[2.5])
Traceback (most recent call last):
  File "<pyshell#67>", line 1, in
<module>
    print(tuple1[2.5])
TypeError: tuple indices must be
integers or slices, not float
```

Modify or deleting element of tuple

- As discuss before tuple is immutable so one cannot change it's element by passing index value.
- Yes we can change entire tuple by replacing it with new value or by performing slicing and concatenate operation together.
- To delete entire tuple we can use inbuilt method of tuple class del().
- It will delete the tuple from the memory and after it no longer in use.
- If we try to use deleted tuple further than python interpreter gives an error.

```
>>> del tuple1
>>> print(tuple1)
Traceback (most recent call last):
  File "<pyshell#70>", line 1, in <module>
    print(tuple1)
NameError: name 'tuple1' is not defined
```

Example of modifying tuple

```
>>> t[0] = 'A'
TypeError: object doesn't support item assignment
>>> tup2=tuple("hetvi")
>>> print(tup2)
('h', 'e', 't', 'v', 'i')
>>> tup2=('H',)+tup2[1:]
>>> print(tup2)
('H', 'e', 't', 'v', 'i')
>>> tup2=tup2[:4]+('I',)
>>> print(tup2)
('H', 'e', 't', 'v', 'I')
>>> tup2=tup2[:2]+('T',)
>>> print(tup2)
('H', 'e', 'T')
```

Concatenation and repetition on tuple

- Same as string we can also perform concatenation and repetition operation tuple by using + and * respectively.
- But the thumb rule to perform concatenation operation on tuple both the data type of both the tuple must same it mean you cannot perform concatenation operation on tuple and list.
- For that required two tuple only.
- Concatenation of tuples is done always from the end of the original tuple.
- Other arithmetic operations do not apply on Tuples.

```
>>> tuple1=(1,2,3,4,5)
>>> tuple2=("xyz","pqr",10,30,45.3)
>>> print(tuple1*3)
(1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
>>> print(tuple2 * 2)
('xyz', 'pqr', 10, 30, 45.3, 'xyz', 'pqr', 10, 30, 45.3)
```

Example of tuple concatenation

```
>>> tuple1=(1,2,3,4,5)
>>> tuple2=("xyz","pqr",10,30,45.3)
>>> tuple1 + tuple2
(1, 2, 3, 4, 5, 'xyz', 'pqr', 10, 30, 45.3)

>>> mes="hello"
>>> tuple1 + mes

Traceback (most recent call last):
  File "<pyshell#77>", line 1, in <module>
    tuple1 + mes
TypeError: can only concatenate tuple (not "str") to tuple
```

Slicing or Splitting on tuple

- Same as string tuple is also allow to perform slicing by passing negative and positive index value inside [ ] bracket by using ":" operator.
- It use to print sequence of element from tuple.
- Tuple1=tuple("WEL COME")
- Than it assigning index value as per below

Positive index value

Negative index value

0	1	2	3	4	5	6	7
W	E	L		C	O	M	E
-8	-7	-6	-5	-4	-3	-2	-1

- We can also say backward indexing and forward indexing.
- Slicing always access element right to left direction in tuple as well in string.

Example of slicing on tuple

```
>>> tuple1=tuple("WEL COME")
>>> print(tuple1[::-1])
('E', 'M', 'O', 'C', ' ', 'L', 'E', 'W')
>>> print(tuple1[3:7])
(' ', 'C', 'O', 'M')
>>> print(tuple1[-1:-5])
()
>>> print(tuple1[-5:-1])
(' ', 'C', 'O', 'M')
>>> print(tuple1[0:])
('W', 'E', 'L', ' ', 'C', 'O', 'M', 'E')
```

Iterate tuple using loop

- Same as string tuple is also iterate using both the loop for and while loop
- But while someone iterate tuple using while he/she must have to find its length using **len()** function as while always execute its iteration on some condition.

```
#using for loop
>>> tup2=tuple("hello")
>>> print(tup2)
('h', 'e', 'l', 'l', 'o')
>>> for i in tup2:
        print(i,)
```

```
#using while loop
>>> tup2=tuple("hello")
>>> tuplen=len(tup2)
>>> i=0
>>> while(i<tuplen):
        print(tup2[i])
        i=i+1
```

Nested tuple

Declare tuple within tuple is known as nested tuple

```
>>> tup1=((1,2,3),4,5,('z','v'),10,20)
>>> print(tup1)
((1, 2, 3), 4, 5, ('z', 'v'), 10, 20)
```

Another way to declare nested tuple like.

```
>>> t1=(1,2,3)
>>> t2=(10,20,30)
>>> t3=('a','b','c')
>>> t4=(t1,t2,t3)
>>> print(t4)
((1, 2, 3), (10, 20, 30), ('a', 'b', 'c'))
```

Accessing element of nested tuple

Same as simple tuple to access element of nested tuple one need to pass two index value

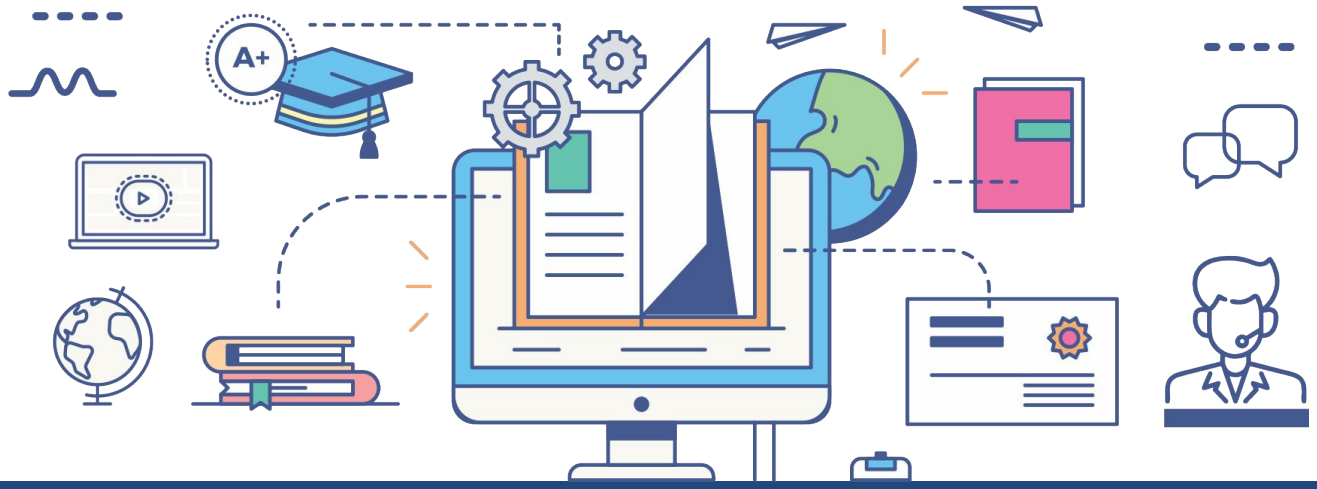
```
>>> print(t4[1][2]) here 1 is nested tuple element number and 2 is its element
30
>>> print(tup1[3][1]) here 3 is nested tuple element and 1 is element number
V
```

Inbuilt function of tuple

Function name	Description	Example
len()	Use to find length of tuple	len(tuple1)
max()	It will return maximum element from tuple	max(tuple1)
min()	It will return minimum element from tuple	min(tuple1)
tuple()	It convert the string or character sequence into tuple	tuple(“hello world”)
cmp(tup 1,tup2)	It will compare both the tuple and return True if tup1 is greater than Tup2	cmp(tup1, tup2)
count()	Count the occurrence of passed element in tuple.	Tup1.count(‘a’)
index()	It will return the index value of passed tuple element	Tup1.index(‘a’)

https://docs.python.org/3/tutorial/datastructures.html#tuples-and-sequences

DIGITAL LEARNING CONTENT



Parul® University

www.paruluniversity.ac.in