Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

Academic Year : 2020-2021

TUPLE DATA TYPE

Guido Van Rossum

Learning Mantra

If you really strong in the basics, then remaining things will become so easy.

Agenda:

1. Accessing elements of tuple
2. Tuple vs immutability
3. Mathematical operators for tuple
4. Important functions/Methods of Tuple

3. Accessing elements of tuple:

- We can access elements of a tuple either by using **index** or by using **slice operator**.

1. By using index:

Eg:

```
t=(10,20,30,40,50,60)

print(t[0])

print(t[-1])

print(t[100])
```

```
10
60

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-7-a762027e6505> in <module>
      2 print(t[0])            #10
      3 print(t[-1])           #60
----> 4 print(t[100])          #IndexError: tuple index out of range

IndexError: tuple index out of range
```

2. By using slice operator:

Eg:

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

```
(30, 40, 50)
(30, 40, 50, 60)
(10, 30, 50)
```

Eg:

```
t= tuple('karthikeya')
print(t[0])
print(t[1:5:1])
print(t[-2:-5:-1])
```

```
k
('a', 'r', 't', 'h')
('y', 'e', 'k')
```

4. Tuple vs immutability

- Once we creates a tuple, we cannot change its content. Hence tuple objects are immutable.

Eg:

t=(10,20,30,40)

t[1]=70

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-12-b9dcfa3c846d> in <module>
      1 t=(10,20,30,40)
----> 2 t[1]=70

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

5. Mathematical operators for tuple

- We can apply + and * operators for tuple.

1. Concatenation Operator(+):

Eg:

```
t1=(10,20,30)
t2=(40,50,60)
t3=t1+t2
print(t3)
```

```
(10, 20, 30, 40, 50, 60)
```

Eg:

t1 = 10,20,30,40

t2 = 10,20,30,40

t3 = t1 + t2 # because list and tuple allow duplicates, so you will get all the elements

print(t3)

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

2. Multiplication operator (or) repetition operator(*)

Eg:

```
t1=(10,20,30)

t2=t1*3

print(t2)     #(10,20,30,10,20,30,10,20,30)
```

```
(10, 20, 30, 10, 20, 30, 10, 20, 30)
```

6. Important functions/Methods of Tuple

1. len():

- It is an in-built function of Python, if you provide any sequence (i.e., strings, list,tuple etc.,), in that how many elements are there that will be returned this function.
- It is used 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():

- It 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))

print(t.index(30))
```

```
0
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-18-e5a94eb1d82a> in <module>
      1 t=(10,20,10,10,20)
      2 print(t.index(10))    # 0
----> 3 print(t.index(30))    # ValueError: tuple.index(x): x not in tuple

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

4. sorted():

- It is used to sort elements based on default natural sorting order (Ascending order).

Eg:

```
t =(10,30,40,20)

print(sorted(t))        # sorted() is going to return list
```

Output:

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

Eg:

t =(10,30,40,20)

t.sort()

print(t)

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-31-6dd56d99cf24> in <module>
      1 t =(10,30,40,20)
----> 2 t.sort()
      3 print(t)

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

Eg:

```
t=(40,10,30,20)
print(id(t))
print(type(t))
t=sorted(t)
print(id(t))
print(type(t))
print(t)          # Result is in List form.
```

```
2653757219768
<class 'tuple'>
2653757029192
<class 'list'>
[10, 20, 30, 40]
```

Eg:

```
t=(40,10,30,20)

t1=sorted(t)

print(type(t1))

print(t1)

print(type(t))

print(t)
```

```
<class 'list'>
[10, 20, 30, 40]
<class 'tuple'>
(40, 10, 30, 20)
```

Eg:

```
t=(40,10,30,20)

t1=tuple(sorted(t))

print(type(t1))

print(t1)

print(type(t1))

print(t)
```

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

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

Eg:

```
t=(40,10,30,20)

t1=sorted(t,reverse=True)

print(t1)          #[40, 30, 20, 10]
```

5. min() and max() functions

- These functions return minimum and maximum values according to default natural sorting order.
- These functions will works on tuple with respect to homogeneous elements only.

Eg:

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

Eg:

```
t = ('karthi')          # based on Unicode values these functions will work.

print(min(t))           # a

print(max(t))           # t
```

Eg:

```
t = ('kArthi')

print(min(t))           # A

print(max(t))           # t
```

6. cmp():

- It compares the elements of both tuples.
- If both tuples are equal then returns 0.
- If the first tuple is less than second tuple then it returns -1.
- If the first tuple is greater than second tuple then it returns +1.

Eg:

```
t1=(10,20,30)

t2=(40,50,60)

t3=(10,20,30)

print(cmp(t1,t2)) # -1

print(cmp(t1,t3)) # 0

print(cmp(t2,t3)) # +1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-41-558f5c41fd64> in <module>
      2 t2=(40,50,60)
      3 t3=(10,20,30)
----> 4 print(cmp(t1,t2)) # -1
      5 print(cmp(t1,t3)) # 0
      6 print(cmp(t2,t3)) # +1

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

Note : cmp() function is available only in Python 2 but not in Python 3

Eg:

```
t1=(10,20,30)

t2=(40,50,60)

t3=(10,20,30)

print(t1==t2)

print(t1==t3)

print(t2==t3)

print(t1<t2) # true, because it compares only first element.
```

```
False
True
False
True
```

Eg:

```
t1=(10,20,30)

t2=(5,50,60)

print(t1<t2)          # False
```

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You