# Tuple Data Type

## Date: 04-05-2020 Day 1

**Topics Covered:**

### Introduction

1. **Creation of Tuple objects**

### Accessing elements of tuple

1. **Tuple vs immutability**

### Mathematical operators for tuple

1. **Important functions of Tuple**
   1. len()
   2. count()
   3. index()
   4. sorted()
   5. min()
   6. max()
   7. cmp()

### Tuple Packing and Unpacking

1. **Tuple Comprehension**

### Differences between List and Tuple

1. **Introduction:**
   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.
   7. 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).
   8. We can represent Tuple elements within Parenthesis and with comma seperator.

#### Note :

Parenethesis are optional but recommended to use.

#### Eg :

In [1]:

t**=**10,20,30,40

print(t)

print(type(t))

(10, 20, 30, 40)

<class 'tuple'>

In [2]:

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

print(t)

print(type(t))

(10, 20, 30, 40)

<class 'tuple'>

In [3]:

t **=** ()

print(type(t))

<class '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:

In [4]:

t**=**(10)

print(t)

print(type(t))

10

<class 'int'>

t**=**(10,)

print(t)

print(type(t))

(10,)

<class 'tuple'>

**Q. Which of the following are valid/Invalid tuples?**

|  |  |  |
| --- | --- | --- |
| In [7]: |  | |
| t**=**() | *#* | *valid* |
| t**=**10,20,30,40 | *#* | *valid* |
| t**=**10 | *#* | *not valid* |
| t**=**10, | *#* | *valid* |
| t**=**(10) | *#* | *notvalid* |
| t**=**(10,) | *#* | *valid* |
| t**=**(10,20,30,40) | *#* | *valid* |
| t**=** (10,20,30,) | *#* | *valid* |
| In [1]: |  |  |
| t **=** (10,20,30,) |  |  |
| print(t) |  |  |
| print(type(t)) |  |  |
| (10, 20, 30) |  |  |
| <class 'tuple'> |  |  |

## Date: 05-05-2020 Day 2

### Creation of Tuple objects

#### 1. t=()

creation of empty tuple

In [2]:

t**=**()

print(t)

print(type(t))

()

<class 'tuple'>

#### 2. t=(10,)

t=10, ====> creation of single valued tuple ,parenthesis are optional,but it should ends with comma.

In [3]:

t **=** (10,)

print(t)

print(type(t))

(10,)

<class 'tuple'>

#### 3. t=10,20,30 or t=(10,20,30)\*

creation of multi values tuples & parenthesis are optional.

In [4]:

t **=** 10,20,30

print(t)

print(type(t))

(10, 20, 30)

<class 'tuple'>

#### 4. By using tuple() function:

if you have any sequence (i.e., string, list, range etc.,) which can be easily converted into a tuple by using

**tuple()** function.

In [5]:

list**=**[10,20,30]

t**=**tuple(list) print(t)

print(type(t))

(10, 20, 30)

<class 'tuple'>

In [9]:

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

print(type(t))

(10, 12, 14, 16, 18)

<class 'tuple'>

In [10]:

t **=** tuple('karthi') print(t)

print(type(t))

('k', 'a', 'r', 't', 'h', 'i')

<class 'tuple'>

### 3. Accessing elements of tuple:

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

#### By using index:

In [7]:

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

print(t[0]) *#10*

print(t[**-**1]) *#60*

print(t[100]) *#IndexError: tuple index out of range*

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

#### By using slice operator:

In [8]:

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*

|  |  |  |
| --- | --- | --- |
| (30, | 40, | 50) |
| (30, | 40, | 50, 60) |
| (10,  **Eg:** | 30, | 50) |

In [11]:

t**=** tuple('karthikeya') print(t[0])

print(t[1:5:1])

print(t[**-**2:**-**5:**-**1])

k

('a', 'r', 't', 'h')

('y', 'e', 'k')

### Tuple vs immutability:

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

#### Eg :

In [12]:

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

### Mathematical operators for tuple:

We can apply + and \* operators for tuple

#### Concatenation Operator(+):

In [13]:

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

t2**=**(40,50,60)

t3**=**t1**+**t2 print(t3)

*# (10,20,30,40,50,60)*

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

In [15]:

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)

#### Multiplication operator (or) repetition operator(\*):

In [14]:

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 of Tuple:

#### len():

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

It is used to return number of elements present in the tuple.

In [16]:

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

print(len(t)) *#4*

4

#### count():

To return number of occurrences of given element in the tuple

In [17]:

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.

In [18]:

t**=**(10,20,10,10,20)

print(t.index(10)) print(t.index(30))

*# 0*

*# ValueError: tuple.index(x): x not in tuple*

0

#### ---------------------------------------------------------------------------

**ValueError** Traceback (most recent call last)

**<ipython-input-18-e5a94eb1d82a>** in <module> 1 t**=(10,20,10,10,20)**

1. 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

#### sorted():

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

In [30]:

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

print(sorted(t))

*# sorted() is going to return list*

[10, 20, 30, 40]

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**()**

1. print**(**t**)**

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

**Reason:** Tuple is Immutable. We cannot modify the tuple contents.

In [34]:

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]

In [36]:

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)

In [37]:

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:

In [20]:

t1**=**sorted(t,reverse**=True**)

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

[40, 30, 20, 10]

#### 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.

In [35]:

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

print(min(t)) *#10*

print(max(t)) *#40*

10

40

In [40]:

t **=** ('karthi')

print(min(t))

print(max(t))

*# based on unicode values these functions will work.*

a t

In [39]:

t **=** ('kArthi')

print(min(t))

print(max(t))

A

t

#### 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.

|  |  |  |
| --- | --- | --- |
| In [41]: |  | |
| 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

In [46]:

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

In [47]:

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

t2**=**(5,50,60)

print(t1**<**t2)

False

### Tuple Packing and Unpacking:

#### Tuple packing :

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

#### Eg:

In [48]:

a**=**10 b**=**20 c**=**30 d**=**40

t**=**a,b,c,d

print(t) *#(10, 20, 30, 40)*

(10, 20, 30, 40)

Here a,b,c,d are packed into a tuple t. This is nothing but **tuple packing.**

#### Tuple unpacking :

Tuple unpacking is the reverse process of tuple packing.

We can unpack a tuple and assign its values to different variables.

#### Eg :

In [49]:

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

a,b,c,d**=**t

print("a=",a,"b=",b,"c=",c,"d=",d)

a= 10 b= 20 c= 30 d= 40

**Note :** This concept is also applicable for any sequence (i.e., string,list,set etc.,) concept also.

#### Unpacking:

In [1]:

t**=**[10,20,30,40]

a,b,c,d**=**t

print("a=",a,"b=",b,"c=",c,"d=",d)

a= 10 b= 20 c= 30 d= 40

In [3]:

t**=**{10,20,30,40}

a,b,c,d**=**t

print("a=",a,"b=",b,"c=",c,"d=",d)

a= 40 b= 10 c= 20 d= 30

In [2]:

t**=**'abcd' a,b,c,d**=**t

print("a=",a,"b=",b,"c=",c,"d=",d)

a= a b= b c= c d= d

#### Packing:

In [4]:

a **=** 10

b **=** 20

c **=** 30

d **=** 40

t **=**[a,b,c,d]

print(type(t)) print(t)

<class 'list'>

[10, 20, 30, 40]

In [7]:

a **=** 10

b **=** 20

c **=** 30

d **=** 40

t **=**{a,b,c,d} *# for 'set' order is not important*

print(type(t)) print(t)

<class 'set'>

{40, 10, 20, 30}

In [6]:

a **=** 10

b **=** 20

c **=** 30

d **=** 40

t **=**'a,b,c,d' print(type(t)) print(t)

<class 'str'> a,b,c,d

#### 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)*

**---------------------------------------------------------------------------**

**ValueError** Traceback (most recent call last)

**<ipython-input-50-11ffc4f6133a>** in <module> 1 t**=(10,20,30,40)**

#### ----> 2 a,b,c=t #ValueError: too many values to unpack (expected 3)

**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.

In [9]:

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

**for** x **in** t:

print(x)

<class 'generator'> 1

4

9

16

25

#### Eg :

**Q. Write a program to take a tuple of numbers from the keyboard and print its sum and average.**

In [11]:

t**=**eval(input("Enter Tuple of Numbers:")) print(type(t))

l**=**len(t) sum**=**0

**for** x **in** t:

sum **=** sum **+** x

print("The Sum=",sum)

print("The Average=",sum**/**l)

Enter Tuple of Numbers:(10,20,30,40)

<class 'tuple'> The Sum= 100

The Average= 25.0

In [12]:

t**=**eval(input("Enter Tuple of Numbers:")) print(type(t))

l**=**len(t) sum**=**0

**for** x **in** t:

sum **=** sum **+** x

print("The Sum=",sum)

print("The Average=",sum**/**l)

Enter Tuple of Numbers:100,200,220,300

<class 'tuple'> The Sum= 820

The Average= 205.0

### 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.

