M – 30 Tuple (Creation and Basic operations) 1/4

Definition

A tuple in a Python is a collection of elements, also known as objects. Like a list, it is an ordered sequence of different data types such as integers, floats, strings etc. But the main differences between tuples and lists are –

- the lists are mutable, whereas the tuples are immutable.
- the lists use square brackets, while the tuples use parentheses.

Creating a tuple

We can create a tuple by putting different comma-separated values. Optionally we can put these comma-separated values between parentheses also. For example −

```
# tup1 is a tuple of integers
tup1 = (4,9,16,25,36)

# tup2 is a is a tuple of strings
tup2 = (‘CS’, ‘Maths', ‘Phy', 'Chem', ’Eng’)

# tup3 is a tuple of mixed data types
tup3 = (‘AECS’, ‘Kudankulam’, “Tamil Nadu”,  627120)

# tup4 is a tuple with a list as an element
tup4 = (50, 60, 70, [80,90])

# tup5 is a tuple with a tuple as an element
Tup5 = (10, 20, 30, 40, 50, (60,70))

#tup6 is a default type of tuple
tup6 = 10, 20, 30, 40                # no parentheses
```

When we don’t use [ ] or ( ), the elements are taken by default as a tuple.

The empty tuple is written with parentheses without any value.

```
tup1 = ()
```

To create a tuple with a single element, we have to include a comma, even though there is only one value in the tuple.

```
tup1 = (50,)
```

We can also create a tuple from the list as illustrated below.

```
#Python code
List = [1,2,3,4,5]          # create a list
tup1 = tuple(List)          # convert a list into tuple
print(tup1)                 # display tuple
```

The output of this code will be 1, 2, 3, 4, 5

The range function can also be used to create a tuple as depicted below.

```
#Python code
tup1 = tuple(range(1,10,2))
print(tup1)
```

The output of this code will be 1, 3, 5, 7, 9

Updating Tuples

As the Tuples are immutable, we cannot update or change the values of tuple elements. However, we are able to extract part of existing tuples in order to create new tuples as illustrated below.

```
tup1 = (4, 16, 36);
tup2 = ('Cat', 'Rat', “Mat”, “Bat”);
# Following action is not valid for tuples
 tup1[0] = 2;
```

However, an element of type list of a tuple is mutable.

#Python code

```
tup1 = (2, 3, 5, 7, [11, 13])
tup1[4][1]=17
print(tup1)
```

The output will be (2, 3, 5, 7, [11, 17])

Further,, we can create a new tuple as shown below.

#Python code

```
tup1 = (4, 16, 36);
tup2 = ('Cat', 'Rat', “Mat”, “Bat”);
tup3 = tup1 + tup2;
print (tup3)
```

When the above code is executed, it produces the following output.

```
(4, 16, 36, Cat, Rat, Mat, Bat)
```

Delete Tuple Elements

From a tuple, it is not possible to remove an individual tuple element. However, the entire tuple can be removed or deleted by using the in-built method del as shown in the following code.

#Python code

```
tup1= (‘Maths’, 'Physics', 'Chemistry', 2019, 2020)
print (tup1)

del tup1
print( "After deleting tup1 : ")
print (tup1)
```

The execution of this code displays the output as shown below.

```
(Maths, Physics, Chemistry, 2019, 2020)

After deleting tup :
Traceback (most recent call last):
  File "test.py", line 9, in <module>
    print (tup1)
NameError: name 'tup'1 is not defined
```

Note an exception raised, this is because after del tup1, tuple does not exist any more.

Basic Tuples Operations

Concatenation

The Python permits us to combine / concatenate two or more tuples by using concatenation operator '+'. Further, a new tuple can also created which consists of the resultant value of this concatenation operation.

```
#Python Code
tup1 = (10, 20, 30, 40)
tup2 = (50, 60, 70)
print(tup1+tup2)
```

The output will be (10, 20, 30, 40,50, 60, 70)

```
#Python Code
tup1 = ('A','E','I')
tup2 = ('O','U')
tup3 =  tup1+tup2
```

print(tup3)

The output will be (A, E, I, O, U)

The concatenation operator can also be used for extending a tuple with 1 or more elements as shown below.

```
#Python Code
tup1 = (10, 20, 30, 40)
tup2 =  tup1 + (50,)

print(tup2)
```

The output will be (10, 20, 30, 40, 50)

```
#Python Code
tup1 = (2,3,5,7,11)
tup2 =  tup1 + (13, 17, 19)
print(tup2)
```

The output will be (2,3,5,7,11, 13, 17, 19)

Repetition

This operation is carried out with the symbol '*', which is used to repeat the tuple elements. This operator needs the first operand to be a tuple and second operand to be an integer, which specifies number of times, the tuple is to be repeated.

```
#Python Code
tup1 = (10, 20, 30)
print(tup1*2)
```

The output will be (10, 20, 30, 10, 20, 30)

```
#Python Code
tup1 = (‘TANQ’)
print(tup1*4)
```

The output will be (‘TANQ’, ‘TANQ’, ‘TANQ’, ‘TANQ’)

Membership

It uses operator ‘in’ to check whether the given element is present in the tuple or not and returns True if the element is present, otherwise returns False. The operator ‘not in’ returns True, if the element is not present in the tuple, otherwise returns False.

```
#Python code
tup1 = (“Chennai”, “Tiruchi”, “Madurai”)

chk1 = “Tiruchi” in tup1
chk2=  “Chennai” not in tup1
print(chk1)
print(chk2)
```

The output will be

```
True
False
```

Indexing/ Slicing a tuple

Like the elements of a string or a list, the values of a tuple can be accessed by using slicing or indexing, which can be carried out by using positive or negative values. Let’s consider a Python code as given below.

```
#Python code
tup1 = (‘Maths’, 'Physics', 'Chemistry', 2019, 2020)
tup2 = (1, 2, 3, 4, 5, 6, 7, 8)
```

```
print( "tup1[0] =  ", tup1[0])            # first element
print ("tup2[2:7] = ", tup2[2:7])       #index 2 to 6
print (" tup2[0:len(tup2)] = ", tup2[0:len(tup2)]])    #all the elements
print ("tup2[ :5] = ", tup2[:5])       # from index 0
print ("tup2[2:] = ", tup2[2:])       # till the last element
print ("tup1[-4:-1] =  ", tup1[-4:-1])     #negative indexing
print(“tup2[::2] =”,tup2[::2])
        #even position elements
print(“tup2[::-1] =”,tup2[::-1])         #elements in the reverse order
```

When the above code is executed, it produces the following output.

```
tup1[0] =   Maths
tup2[2:7] =  (3, 4, 5, 6, 7)
tup2[0:len(tup2)] =  (1, 2, 3, 4, 5, 6, 7, 8)
tup2[ :5] =  (1, 2, 3, 4, 5)
tup2[2:] =  (3, 4, 5, 6, 7, 8)
tup1[-4:-1] =   ('Physics', 'Chemistry', 2019)
tup2[::2] =  (1, 3, 5, 7)
tup2[::-1] =  (8, 7, 6, 5, 4, 3, 2, 1)
```

Like a list, the elements of a tuple can be accessed by using for as shown below.

```
#Python code
tup = (20, 40, 60, 80)

for I in tup:
    print(I, end = ‘  ‘)
```

The execution of this code will display the output as

```
20 40 60 80
```

Tuple Assignment

It is one of the features of Python which permits us to assign elements of a tuple with the variables of a tuple. The number of elements to be assigned should be equivalent to numbers of variables which assign the values.

```
#Python code
#Tuple assignment
(n1,n2,n3) = (10, 20, 30)
print(n1,’,’,n2,’,’,n3)
SchRec = (35, 'Himanshi', 'AECS Kudankulam', 627120)
(Rno,Name,SchAddr, Pin) = SchRec
print(Rno, ‘, ’, Name,’, ’, SchAddr, ‘, ’, Pin)
```

OUTPUT

```
10 20 30
35, Himanshi, AECS Kudankulam, 627120
```

Further, the expressions can be evaluated and assigned with a tuple as illustrated in the following code.

```
#Python code
#Evaluate expressions
#Assign with tuple
Ar = (15+5, 15*5, 15/5, 15-5)
(add, mul, div, sub) = Ar
print('Sum= ',add, ', Product= ', mul, ' Division= ', div, ' Diff.= ', sub)
```

OUTPUT

```
Sum= 20,  Product= 75, Division= 3.0, Diff.= 10
```

0o0o0o0o0o0o0