Std XI : Computer Science

Tuple
Module 32 (3/4)
Nested Tuples

E-Module by AEES, Mumbai

Definition

In Python, a tuple written inside another tuple is known as a nested tuple. Let’s consider a tuple having 7 elements as shown below.

tup = (10, 20, 30, 40, 50, 60, (100, 200, 300))

Here, the last element consisting of 3 elements written within parentheses is called a nested tuple as it is inside another tuple. The nested tuple with the elements **(100, 200, 300)** can be retrieved by using tuple name with the index value i.e. tup[index] and each element of the nested tuple can be accessed by using tup[index-1][index-2].

```
#Python code
tup = ( 10, 20, 30, 40, 50, 60, (100, 200, 300))
print('Nested tuple : ', tup[6])
print('Nested tuple element : ',tup[6][1])
```

The output of this code will be

```
Nested tuple :  (100, 200, 300)
Nested tuple element :  200
```

Storing records in nested tuple

Each nested tuple can represent a specific data record. For instance, records of many students consisting RollNo, Name and Aggregate can be stored in a nested tuple as depicted below.

min() function

It is used to find/ return the minimum value of the elements stored in the tuple.

```
tup = (-70, -80,10,20, 30)
sma= min(tup)
print('Minimum element  : ', sma)
```

The output of this code will be **Minimum element : -80**

max() function

It is used to find/ return the maximum value among the elements stored in the tuple.

```
tup = (-70, -80,10,20, 30)
big = max(tup)
print('Maximum element  : ', big)
```

The output of this code will be **Maximum element : 30**

```
#Python code to store records
StdRec = ((115,'Kriyansh',485),(114,'Arvind', 460),(113,'Sruti   ',486), (116,
'Krishant', 480),(111, 'Swati   ', 490),(112,'Ishwarya', 489))

print('S. No.', 'RollNo','\t   Name','\tAggregate')
for i in range(len(StdRec)):
    print(i+1,'\t',StdRec[i][0],'\t',StdRec[i][1],'\t',StdRec[i][2])
```

The output of this code will be

```
S. No. RollNo          Name       Aggregate
1        115         Kriyansh      485
2        114         Arvind        460
3        113         Sruti         486
4        116         Krishant      480
5        111         Swati         490
6        112         Ishwarya      489
```

Sorting Nested tuple

As we know, elements of a tuple can be sorted by using sorted() function. When we write this function as given below,

print(sorted(StdRec))

the nested tuple elements will be sorted in the ascending order of the 0th element i.e. RollNo. If we want to arrange the tuple on basis of Name (1st elemen) or Aggregate (2nd element), the lambda expression needs to be used as depicted below.

print(sorted(StdRec, key = lambda a: a[1]))   # Arrange on Name

```
#Python code to store student records
#With each record consists of RollNo, Name and Aggregate
#Arrange each record by RollNo
StdRec = ((115,'Kriyansh',485),(114,'Arvind', 460),(113,'Sruti   ',486),
(116, 'Krishant', 480),(111, 'Swati   ', 490),(112,'Ishwarya', 489))

SOnName=sorted(StdRec)

print('S. No.', 'RollNo','\t   Name','\tAggregate')
 for i in range(len(SOnName)):
    print(i+1,'\t',SOnName[i][0],'\t',SOnName[i][1],'\t',SOnName[i][2])
```

The output of this code will be

```
S. No. RollNo      Name      Aggregate
1        111      Swati       490
2        112      Ishwarya    489
3        113      Sruti       486
4        114      Arvind      460
5        115      Kriyansh    485
6        116      Krishant    480
```

```
#Python code to store student records
#Arrange each record by name
StdRec = ((115,'Kriyansh',485),(114,'Arvind', 460),(113,'Sruti   ',486),
(116, 'Krishant', 480),(111, 'Swati   ', 490),(112,'Ishwarya', 489))
```

```
SOnName=sorted(StdRec, key=lambda a:a[1])
print('S. No.', 'RollNo','\t   Name','\tAggregate')
for i in range(len(SOnName)):
   print(i+1,'\t',SOnName[i][0],'\t',SOnName[i][1],'\t',SOnName[i][2])
```

The output of this code will be

```
S. No. RollNo        Name      Aggregate
1         114       Arvind       460
2         112       Ishwarya     489
3         116       Krishant     480
4         115       Kriyansh     485
5         113       Sruti        486
6         111       Swati        490
```

```
#Python code to store student records
#Arrange each record by Aggregate in descending order
StdRec = ((115,'Kriyansh',485),(114,'Arvind', 460),(113,'Sruti
',486), (116, 'Krishant', 480),(111, 'Swati   ', 490),(112,'Ishwarya',
489))

SOnAgg=sorted(StdRec, reverse=True, key=lambda a:a[2])
print('S No.', 'RollNo','\t   Name','\tAggregate')
for i in range(len(SOnAgg)):
    print(i+1,'\t',SOnAgg[i][0],'\t',SOnAgg[i][1],'\t',SOnAgg[i][2])
```

The output of this code will be

```
S No. RollNo      Name      Aggregate
1     111        Swati       490
2     112        Ishwarya    489
3     113        Sruti       486
4     115        Kriyansh    485
5     116        Krishant    480
6     114        Arvind      460
```

Reading and processing n elements of tuple

The following Python code illustrates how to read and process ‘n’ elements for a tuple.

```
#Python code
#To read n elements for a tuple from the user and,
#Find maximum and minimum among them
n = int(input('Enter number of elements : '))
tup = tuple()       # create empty tuple
for i in range(n):
    print('Enter element ',i+1,end=' : ')
    ele = int(input())    # read each integer element from the user
    tup += (ele,)   # assign the element with the tuple
 print('Given tuple : ',tup)
print('Maximum among them : ',max(tup))
print('Minimum among them : ',min(tup))
```

OUTPUT

```
Enter number of elements : 6
Enter element  1 : 10
Enter element  2 : 20
Enter element  3 : 15
Enter element  4 : 9
Enter element  5 : 30
Enter element  6 : 25
Given tuple :  (10, 20, 15, 9, 30, 25)
Maximum among them :  30
Minimum among them :  9
```

Find frequency of an element in a tuple

It is a process of determining frequency of each element (number of times used) in the tuple.

```
#Python code
#To read n elements for a tuple
#and find frequency of each element in the tuple
n = int(input('Enter number of elements : '))
tup = tuple()
for i in range(n):
    print('Enter element ',i+1,end=' : ')
    ele = int(input())
    tup += (ele,)
print('\nGiven tuple : ',tup)
freq = [None]*n     #create a list with n no values
checked = False
```

```
for i in range(n):
    count = 1
    for j in range(i+1,n):
        if tup[i] == tup[j]:
            count = count + 1
            freq[j]=checked
    if freq[i]!= checked:
        freq[i]=count
print('\nElement','  Frequency')
for i in range(len(freq)):
    if freq[i]!=checked:
        print(tup[i],'\t',freq[i])
```

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
```

Have a nice day !!!