Lecture#5 Python Programming

Python TUPLE – Pack, Unpack, Compare, Slicing, Delete, Key

What is Tuple Matching in Python?

Tuple Matching in Python is a method of grouping the tuples by matching the second element in the tuples. It is achieved by using a dictionary by checking the second element in each tuple in python programming. However, we can make new tuples by taking portions of existing tuples.

Tuple Syntax

```
Tup = ('Jan','feb','march')
```

To write an empty tuple, you need to write as two parentheses containing nothing

```
tup1 = ();
```

Tuple indices begin at 0, and they can be concatenated, sliced and so on.

In this tutorial, we will learn-

- Packing and Unpacking
- Comparing tuples
- Using tuples as keys in dictionaries
- Deleting Tuples
- Slicing of Tuple
- Built-in functions with Tuple
- Advantages of tuple over list

Tuple Assignment

Python has tuple assignment feature which enables you to assign more than one variable at a time. In here, we have assigned tuple 1 with the persons information like name, surname, birth year, etc. and another tuple 2 with the values in it like number (1,2,3,….,7).

For Example,

(name, surname, birth year, favorite movie and year, profession, birthplace) = Robert

Here is the code,

```
tup1 = ('Robert', 'Carlos','1965','Terminator 1995', 'Actor','Florida');
tup2 = (1,2,3,4,5,6,7);
print(tup1[0])
print(tup2[1:4])
```

- Tuple 1 includes list of information of Robert
- Tuple 2 includes list of numbers in it
- We call the value for [0] in tuple and for tuple 2 we call the value between 1 and 4
- Run the code- It gives name Robert for first tuple while for second tuple it gives number (2,3 and 4)

Packing and Unpacking

In packing, we place value into a new tuple while in unpacking we extract those values back into variables.

```
x = ("Guru99", 20, "Education")    # tuple packing
(company, emp, profile) = x    # tuple unpacking
print(company)
print(emp)
print(profile)
```

Comparing tuples

A comparison operator in Python can work with tuples.

The comparison starts with a first element of each tuple. If they do not compare to =,< or > then it proceed to the second element and so on.

It starts with comparing the first element from each of the tuples

Let's study this with an example-

#case 1

```
a=(5,6)
b=(1,4)
if (a>b):print("a is bigger")
else: print("b is bigger")
```

#case 2

```
a=(5,6)
b=(5,4)
if (a>b):print("a is bigger")
else: print ("b is bigger")
```

#case 3

```
a=(5,6)
b=(6,4)
if (a>b):print("a is bigger")
else: print("b is bigger")
```

Case1: Comparison starts with a first element of each tuple. In this case 5>1, so the output a is bigger

Case 2: Comparison starts with a first element of each tuple. In this case 5>5 which is inconclusive. So it proceeds to the next element. 6>4, so the output a is bigger

Case 3: Comparison starts with a first element of each tuple. In this case 5>6 which is false. So it goes into the else block and prints “b is bigger.”

Using tuples as keys in dictionaries

Since tuples are hashable, and list is not, we must use tuple as the key if we need to create a composite key to use in a dictionary.

Example: We would come across a composite key if we need to create a telephone directory that maps, first-name, last-name, pairs of telephone numbers, etc. Assuming that we have declared the variables as last and first number, we could write a dictionary assignment statement as shown below:

```
directory[last,first] = number
```

Inside the brackets, the expression is a tuple. We could use tuple assignment in a for loop to navigate this dictionary.

```
for last, first in directory:
```

```
print first, last, directory[last, first]
```

This loop navigates the keys in the directory, which are tuples. It assigns the elements of each tuple to last and first and then prints the name and corresponding telephone number.

Tuples and dictionary

Dictionary can return the list of tuples by calling items, where each tuple is a key value pair.

```
a = {'x':100, 'y':200}
b = list(a.items())
print(b)
```

Deleting Tuples

Tuples are immutable and cannot be deleted. You cannot delete or remove items from a tuple. But deleting tuple entirely is possible by using the keyword

```
del
```

Slicing of Tuple

To fetch specific sets of sub-elements from tuple or list, we use this unique function called slicing. Slicing is not only applicable to tuple but also for array and list.

```
x = ("a", "b","c", "d", "e")
print(x[2:4])
```

The output of this code will be (‘c’, ‘d’).

Here is the Python 2 Code for all above example

```
tup1 = ('Robert', 'Carlos','1965','Terminator 1995', 'Actor','Florida');
tup2 = (1,2,3,4,5,6,7);
print  tup1[0]
print  tup2[1:4]

#Packing and Unpacking
x = ("Guru99", 20, "Education")    # tuple packing
(company, emp, profile) = x    # tuple unpacking
print company
print emp
print profile

#Comparing tuples
#case 1
a=(5,6)
b=(1,4)
if (a>b):print "a is bigger"
else: print "b is bigger"

#case 2
a=(5,6)
b=(5,4)
if (a>b):print "a is bigger"
else: print "b is bigger"

#case 3
a=(5,6)
b=(6,4)
if (a>b):print "a is bigger"
else: print "b is bigger"

#Tuples and dictionary
a = {'x':100, 'y':200}
b = a.items()
print b

#Slicing of Tuple
x = ("a", "b","c", "d", "e")
print x[2:4]
```

Built-in functions with Tuple

To perform different task, tuple allows you to use many built-in functions like all(), any(), enumerate(), max(), min(), sorted(), len(), tuple(), etc.

Advantages of tuple over list

- Iterating through tuple is faster than with list, since tuples are immutable.
- Tuples that consist of immutable elements can be used as key for dictionary, which is not possible with list
- If you have data that is immutable, implementing it as tuple will guarantee that it remains write-protected

Summary

Python has tuple assignment feature which enables you to assign more than one variable at a time.

- Packing and Unpacking of Tuples
    - In packing, we place value into a new tuple while in unpacking we extract those values back into variables.
- A comparison operator in Python can work with tuples.
- Using tuples as keys in dictionaries
    - Tuples are hashable, and list are not
    - We must use tuple as the key if we need to create a composite key to use in a dictionary
    - Dictionary can return the list of tuples by calling items, where each tuple is a key value pair
- Tuples are immutable and cannot be deleted. You cannot delete or remove items from a tuple. But deleting tuple entirely is possible by using the keyword “del”
- To fetch specific sets of sub-elements from tuple or list, we use this unique function called slicing