UNIT 4

Python Tuple and Dictionary

Introduction to Tuple:-

A tuple is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple: - it is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

```
# Different types of tuples

# Empty tuple
my_tuple = ()
print(my_tuple)

# Tuple having integers
my_tuple = (1, 2, 3)
print(my_tuple)

# tuple with mixed datatypes
my_tuple = (1, "Hello", 3.4)
print(my_tuple)

# nested tuple
my_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
print(my_tuple)
```

Output

```
()
(1, 2, 3)
(1, 'Hello', 3.4)
('mouse', [8, 4, 6], (1, 2, 3))
```

A tuple can also be created without using parentheses. This is known as tuple packing.

```
my_tuple = 3, 4.6, "dog"
print(my_tuple)

# tuple unpacking is also possible
a, b, c = my_tuple

print(a)      # 3
print(b)      # 4.6
print(c)      # dog
```

Output

```
(3, 4.6, 'dog')
3
4.6
dog
```

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
print "tup1[0]: ", tup1 [0];
print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result −

```
tup1[0]:  physics
tup2[1:5]:  [2, 3, 4, 5]
```

Updating Tuples

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates −

```
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');

# Following action is not valid for tuples
# tup1[0] = 100;

# So let's create a new tuple as follows
tup3 = tup1 + tup2;
print tup3;
```

When the above code is executed, it produces the following result −

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the **del** statement. For example −

```
tup = ('physics', 'chemistry', 1997, 2000)
print tup
del tup
print "After deleting tup : "
print tup
```

This produces the following result. Note an exception raised, this is because after **del tup** tuple does not exist any more −

```
('physics', 'chemistry', 1997, 2000)
After deleting tup :
Traceback (most recent call last):
  File "test.py", line 9, in <module>
    print tup;
NameError: name 'tup' is not defined
```

Tuple Operations:-

Concatenation : Tuples can be concatenated by using '+' operator, the result is new tuple and not a string.

e.g. tuple1 = (1,2,3)

tuple2 = ("x","y","z")

tuple3 = tuple1 + tuple2

print(tuple3)

It concatenates the tuple1 and tuple2 and produces the result as new tuple

(1,2,3,,'x','y','z')

Repetition : Sometime we need to replicate the data in a tuple. The multiplication operator is used for repetition to make multiple copies of a tuple and joins them all together.

tup1 =(1,2,3)

re_tupl = (tup1,) * 2

print(re_tup1)

It replicates the contents of tuple tupl1 and prints the result as a new tuple

(1,2,3,1,2,3)

Membership : It is used to determine if a specified item is present in a tuple. Membership in a tuple is checked using the 'in' keyword:

```
thistuple = ("apple", "watermelon", "banana")
if  "apple" in thistuple:
print("Yes, apple is in the fruits tuple")
```

In above code tuple thistuple contain name of fruits. Command 'if' checks whether apple is member of thistuple. If apple is in thistuple, it prints the give message.

After executing the code it gives result : 'Yes, apple is in the fruits tuple

e.g.
```
if 3 in (1,2,3)
print("Element present")
```

After execution of above code the result is : Element present

Iteration : We can iterate over tuples using a simple for loop. Tuples are **sequence data type** and we can iterate over sequences.

e.g.
```
for x in (1, 2, 3):
print x
```

The result is : 1,2,3

e.g.
```
fruit_names = ("apple", "watermelon", "banana")
for fruit in fruit_names
print(fruit)
```

The result is : 'apple', 'watermelon' 'banana'

Built-in Tuples functions : Following are built in tuple functions :

len( ) : len( ) is a buit-in tuple function to compute number of items present in given.

e.g.
```
fruit_names = ("apple", "watermelon", "banana")
print(len(fruit_names))
```

The result is : 3

cmp(tuple1,tuple2) : cmp( ) is a built-in tuple function in which elements of tuple1 are compared with elements of tuple2 , if both lists are same it returns 0 otherwise it returns 1.

e.g.
```
tuple1 = ('x','y','z')
tuple2 = (1,2,3)
tuple3 = ('x','y','z')
cmp(tuple1,tuple2)
output : 1
cmp(tuple1,tuple3)
output : 0
```

any() : This built-in tuple function returns true if any element present in a tuple and returns false if tuple is empty.

e.g. tuple1 = ('x','y','z')

print("Is there any element is present in tuple :", any(tuple1))

Output : Is there any element is present in tuple : True

e.g. tuple2 = ( )

print("Is there any element is present in tuple :", any(tuple2))

Output : Is there any element is present in tuple : False

min( ) : This tuple function returns smallest element(int) of the tuple

e.g. tuple1=(4,5.7,2.3,5)

print("Smallest element in tuple is :", min(tuple1))

It prints the result as : Smallest element in tuple is : 2.3

max( ) : This tuple function returns largest element(int) of the tuple

e.g. tuple1=(4,5.7,2.3,5)

print("Largest element in tuple is :", max(tuple1))

It prints the result as : Largest element in tuple is : 5.7

sorted( ) : It is used to sort all elements of the tuple.

e.g. tuple1=(4,5.7,2.3,5)

print("Sorted elements in tuple are :", sorted(tuple1))

It prints the result as : Sorted element in the tuple is : (2.3,4,5,5.7)

e.g. tuple1 = ('e', 'a', 'u', 'o', 'i')

print("Sorted characters are:", sorted(tuple1))

When we execute the code output is : Sorted characters are : ('a','e','i','o','u')

sum( ) **:** It returns sum of all elements of tuple.

e.g. Num=(3,5.1,2,9,3.5)

print("Sum of all the numbers in tuple is:", sorted(tuple1))

When we execute this code output is : Sum of all the numbers in the tuples is: 22.6