Tuples

- Python **Tuple** is a collection of objects separated by commas.
- In some ways, a tuple is similar to a Python list in terms of indexing, nested objects, and repetition but the main difference between both is Python tuple is immutable, unlike the Python list which is mutable.

❖ Features of Python Tuple

- ➢ Tuples are an immutable data type, meaning their elements cannot be changed after they are generated.
- ➢ Each element in a tuple has a specific order that will never change because tuples are ordered sequences.

❖ Forming a Tuple:

- All the objects-also known as "elements"-must be separated by a comma, enclosed in parenthesis ().
- Although parentheses are not required, they are recommended.
- Any number of items, including those with various data types (dictionary, string, float, list, etc.), can be contained in a tuple.

Example:

```
# Python program to show how to create a tuple
# Creating an empty tuple
empty_tuple = ()
print("Empty tuple: ", empty_tuple)

# Creating tuple having integers
int_tuple = (4, 6, 8, 10, 12, 14)
```

```
print("Tuple with integers: ", int_tuple)

# Creating a tuple having objects of different data types
mixed_tuple = (4, "Python", 9.3)
print("Tuple with different data types: ", mixed_tuple)

# Creating a nested tuple
nested_tuple = ("Python", {4: 5, 6: 2, 8:2}, (5, 3, 5, 6))
print("A nested tuple: ", nested_tuple)
```

Output:

```
Empty tuple:  ()
Tuple with integers:  (4, 6, 8, 10, 12, 14)
Tuple with different data types:  (4, 'Python', 9.3)
A nested tuple:  ('Python', {4: 5, 6: 2, 8: 2}, (5, 3, 5, 6))
```

❖ What is Immutable in Tuples?

- Tuples in Python are similar to Python lists but not entirely.
- Tuples are immutable and ordered and allow duplicate values.

Accessing Values in Python Tuples

- Tuples in Python provide two ways by which we can access the elements of a tuple.

➢ Python Access Tuple using a Positive Index

Using square brackets we can get the values from tuples in Python.

Example:

```
var = ("Topper", "World")


print("Value in Var[0] = ", var[0])
print("Value in Var[1] = ", var[1])
```

Output:

```
Value in Var[0] =  Topper
Value in Var[1] =  World
```

➢ Access Tuple using Negative Index

In the above methods, we use the positive index to access the value in Python, and here we will use the negative index within [].

Example:

```
var = (1, 2, 3)
print("Value in Var[-1] = ", var[-1])
print("Value in Var[-2] = ", var[-2])
print("Value in Var[-3] = ", var[-3])
```

Output:

```
Value in Var[-1] =  3
Value in Var[-2] =  2
Value in Var[-3] =  1
```

❖ Tuples in a loop

We can also create a tuple with a single element in it using loops.

Example:

```
# python code for creating tuples in a loop
tup = ('Topper',)

# Number of time loop runs
n = 5
for i in range(int(n)):
        tup = (tup,)
        print(tup)
```

Output:

```
(('Topper',),)
(((('Topper',),),),)
((((('Topper',),),),),)
(((((('Topper',),),),),),)
((((((('Topper',),),),),),),)
```

❖ Advantages of Tuples

- Tuples take less time than lists do.
- Due to tuples, the code is protected from accidental modifications. It is desirable to store non-changing information in "tuples" instead of "records" if a program expects it.

- A tuple can be used as a dictionary key if it contains immutable values like strings, numbers, or another tuple. "Lists" cannot be utilized as dictionary keys because they are mutable.