Definition:-

Tuples are used to store multiple items in a single variable. A tuple in python is just like a list. The difference between the two is that we cannot change the elements of a tuple once it is assigned where as we can change the elements of a list. Tuple items are ordered, unchangeable, and allow duplicate values.

How to create a tuple:-

A tuple is created by placing all the items (elements) inside parentheses (), separated by commas. The parentheses are optional, however, it is a good practice to use them. A tuple can have any number of items and they may be of different types (integer, float, list, [string](https://www.programiz.com/python-programming/string), etc.).

**DIFFERENT EXAMPLE OF TUPLE**

# how to create Empty tuple

my\_tuple = ()

print (my\_tuple)

# How to create Tuple having integers

my\_tuple = (5, 6, 7)

print (my\_tuple)

# how to create tuple with mixed datatypes

my\_tuple = (1, "Rajat", 5.6)

print (my\_tuple)

Output:-

()

(5, 6, 7)

(1, 'Rajat', 5.6)

Access Tuple Elements

There are various ways in which we can access the elements of a tuple.

1. Indexing

We can use the index operator [] to access an item in a tuple, where the index starts from 0.

So, a tuple having 7 elements will have indices from 0 to 6. Trying to access an index outside of the tuple index range (7, 8,... in this example) will raise an IndexError.The index must be an integer, so we cannot use float or other types. This will result in TypeError.

**Ex:-**

my\_tuple = (5, 6, 7)

print (my\_tuple [1])

Output: - (6)

2. Negative Indexing

Python allows negative indexing for its sequences.

The index of -1 refers to the last item, -2 to the second last item and so on.

# Negative indexing for accessing tuple elements

my\_tuple = ('r', 'a', 'j', 'a', 't', 'v')

# Output: 'v'

print(my\_tuple[-1])

# Output: 'r'

print(my\_tuple[-6])

3. Slicing

We can access a range of items in a tuple by using the slicing operator colon:

# Accessing tuple elements using slicing

my\_tuple = ('H','a','r','t','r','o','m','r','v')

# elements 2nd to 4th

# Output: ('a', 'r', 't')

print(my\_tuple[1:4])

# Elements beginning to end

# Output: ('H', 'a', 'r', 't’, 'r', 'o', 'n, 'r', 'v')

print(my\_tuple[:])

Changing a Tuple

Tuples are immutable. This means that elements of a tuple cannot be changed once they have been assigned. But, if the element is itself a mutable data type like list, its nested items can be changed.

Deleting a Tuple

As discussed above, we cannot change the elements in a tuple. It means that we cannot delete or remove items from a tuple.

Deleting a tuple entirely, however, is possible using the keyword [del](https://www.programiz.com/python-programming/keyword-list#del).

my\_tuple = ('H','a','r','t','r','o','m','r','v')

del my\_tuple

Advantages of Tuple over List

Since tuples are quite similar to lists, both of them are used in similar situations. However, there are certain advantages of implementing a tuple over a list. Below listed are some of the main advantages:

We generally use tuples for heterogeneous (different) data types and lists for homogeneous (similar) data types.

Since tuples are immutable, iterating through a tuple is faster than with list. So there is a slight performance boost.

Tuples that contain immutable elements can be used as a key for a dictionary. With lists, this is not possible.

If you have data that doesn't change, implementing it as tuple will guarantee that it remains write-protected.