**Operations On Tuple**

Indexing

-- We can use the index operator [] to access an object in a tuple, where the index starts at 0.

-- The indices of a tuple with five items will range from 0 to 4.

-- An Index Error will be raised assuming we attempt to get to a list from the Tuple that is outside the scope of the tuple record.

-- An index above four will be out of range in this scenario.

-- Because the index in Python must be an integer, we cannot provide an index of a floating data type or any other type. If we provide a floating index, the result will be TypeError.

# *Python program to show how to access tuple elements*

# *Creating a tuple*

tuple\_ = ("Python", "Tuple", "Ordered", "Collection")

print(tuple\_[0])        # *o/p: Python*

print(tuple\_[1])        # *o/p: Tuple*

# *trying to access element index more than the length of a tuple*

*try*:

    print(tuple\_[5])

*except* Exception *as* e:

    print(e)            # *o/p: tuple index out of range*

# *trying to access elements through the index of floating data type*

*try*:

    print(tuple\_[1.0])

*except* Exception *as* e:

    print(e)            # *o/p: tuple indices must be integers or slices, not float*

# *Creating a nested tuple*

nested\_tuple = ("Tuple", [4, 6, 2, 6], (6, 2, 6, 7))

# *Accessing the index of a nested tuple*

print(nested\_tuple[0][3])   # *o/p: 1*

print(nested\_tuple[1][1])   # *o/p: 6*

-- Python's sequence objects support negative indexing.

-- The last thing of the assortment is addressed by - 1, the second last thing by - 2, etc.

# *Python program to show how negative indexing works in Python tuples*

# *Creating a tuple*

tuple\_ = ("Python", "Tuple", "Ordered", "Collection")

# *Printing elements using negative indices*

print("Element at -1 index: ", tuple\_[-1])

# *o/p: Element at -1 index:  Collection*

print("Elements between -4 and -1 are: ", tuple\_[-4:-1])

# *o/p: Elements between -4 and -1 are: ('Python', 'Tuple', 'Ordered')*

Slicing

-- Tuple slicing is a common practice in Python and the most common way for programmers to deal with practical issues.

-- Look at a tuple in Python. Slice a tuple to access a variety of its elements. Using the colon as a straightforward slicing operator (:) is one strategy.

-- To gain access to various tuple elements, we can use the slicing operator colon (:).

# *Python program to show how slicing works in Python tuples*

# *Creating a tuple*

tuple\_ = ("Python", "Tuple", "Ordered", "Immutable", "Collection", "Objects")

# *Using slicing to access elements of the tuple*

print("Elements between indices 1 and 3: ", tuple\_[1:3])

# *o/p: Elements between indices 1 and 3:  ('Tuple', 'Ordered')*

# *Using negative indexing in slicing*

print("Elements between indices 0 and -4: ", tuple\_[:-4])

# *o/p: Elements between indices 0 and -4:  ('Python', 'Tuple')*

# *Printing the entire tuple by using the default start and end values.*

print("Entire tuple: ", tuple\_[:])

# *o/p: Entire tuple:  ('Python', 'Tuple', 'Ordered', 'Immutable', 'Collection', 'Objects')*