TOPS TECHNOLOGY

Python – Collections, functions and Modules

Presented By:
Nandni Vala

Tuple

1.Introduction to tuples, immutability.

- ➢ **Introduction to Tuples**
- ➢ A **tuple** is a collection in Python that is:
- ➢ **Ordered**: Elements maintain a specific sequence.
- ➢ **Immutable**: Once created, the elements cannot be modified (no changes, additions, or deletions).
- ➢ **Allows Duplicates**: Tuples can have repeated values.
- ➢ Tuples are defined using **parentheses** ().

- **Immutability of Tuples**
- **Immutability** means that the elements of a tuple cannot be changed after it is created.
- You **cannot modify, add, or remove** elements, but you can access and use the tuple as it is.
- Example:
- my_tuple = (1, 2, 3)
- # Trying to modify a tuple will raise an error:
- my_tuple[1] = 5 # Error: 'tuple' object does not support item assignment
-

2.Creating and accessing elements in a tuple.

- **Creating a Tuple**
- **Empty Tuple**:
- empty_tuple = ()
- print(empty_tuple)
- Output: ()
- **Tuple with Multiple Elements**:
- my_tuple = (1, 2, 3)
- print(my_tuple)
- Output: (1, 2, 3)

- **Single Element Tuple**:
- Use a trailing comma to create a tuple with one element.
- single_element = (5)
- print(single_element)
- Output: (5)
- **Mixed Data Types**:
- Tuples can hold elements of different types.
- mixed_tuple = (1, "hello", 3.14)
- print(mixed_tuple)
- Output: (1, 'hello', 3.14)

- **Accessing Elements in a Tuple :**
- **Indexing**: Access specific elements using an index (starts from 0).
- my_tuple = (10, 20, 30)
- print(my_tuple[1])
- Output: 20
- **Negative Indexing**: Access elements from the end of the tuple using negative indices.
- print(my_tuple[-1])
- Output: 30
- **Slicing**: Access a range of elements using slicing (start:end:step).
- print(my_tuple[0:2])
- Output: (10, 20)
- print(my_tuple[::-1])
- Output: (30, 20, 10)

3.Basic operations with tuples: concatenation, repetition, membership

- **Concatenation :**
- Combines two or more tuples into a single tuple.
- Uses the + operator.
- tuple1 = (1, 2, 3)
- tuple2 = (4, 5, 6)
- result = tuple1 + tuple2
- print(result)
- Output: (1, 2, 3, 4, 5, 6)

- **Repetition :**
- Repeats the elements of a tuple a specified number of times.
- Uses the * operator.
- tuple1 = (1, 2, 3)
- result = tuple1 * 3
- print(result)
- Output: (1, 2, 3, 1, 2, 3, 1, 2, 3)
- **Membership :**
- Checks whether an element exists in a tuple.
- Uses the in or not in operators.
- my_tuple = (10, 20, 30)
- print(20 in my_tuple)   # Output: True