1. Accessing List

Understanding how to create and access elements in a list:

A list in Python is an ordered collection that can store multiple items of different data types. Lists are mutable, meaning their elements can be modified after creation. Lists are created using square brackets [].

Indexing in lists (positive and negative indexing):

Indexing allows accessing individual elements in a list using their position. Positive indexing starts from 0, while negative indexing starts from -1 for the last element.

Slicing a list: accessing a range of elements:

Slicing allows accessing a subset of list elements using the syntax list[start:end]. The 'start' index is inclusive, while the 'end' index is exclusive.

2. List Operations

Common list operations: concatenation, repetition, membership:

Lists support operations like concatenation (+), repetition (*), and membership (in, not in). These allow combining, repeating, or checking the presence of elements.

Understanding list methods like append(), insert(), remove(), pop():

• append(): Adds an element to the end of the list.
• insert(index, element): Inserts an element at a specified position.
• remove(element): Removes the first occurrence of an element.
• pop(index): Removes an element at a specific index (default last).

3. Working with Lists

Iterating over a list using loops:

You can loop through lists using a for loop to process or print each element.

Sorting and reversing a list using sort(), sorted(), and reverse():

• sort(): Sorts the list in ascending order (modifies the list).
• sorted(): Returns a new sorted list (does not modify original).
• reverse(): Reverses the order of list elements.

Basic list manipulations:

You can add, delete, update, or slice list elements dynamically, making them flexible data structures for collection handling.

4. Tuple

Introduction to tuples and immutability:

A tuple is an ordered collection like a list, but it is immutable, meaning elements cannot be changed after creation. Tuples are created using parentheses ().

Creating and accessing elements in a tuple:

Tuples can store multiple data types and can be accessed using indexes like lists.

Basic operations with tuples:

Tuples support concatenation (+), repetition (*), and membership (in, not in).

5. Accessing Tuples

Accessing tuple elements using positive and negative indexing:

Like lists, tuples support both positive and negative indexing. Positive indexes start from 0, while negative indexes start from -1 for the last item.

Slicing a tuple to access ranges of elements:

Tuples support slicing using the syntax tuple[start:end], which returns a new tuple containing the specified range of elements.

6. Dictionaries

Introduction to dictionaries: key-value pairs:

A dictionary in Python is an unordered collection of key-value pairs. Each key must be unique, and values can be of any data type. Dictionaries are created using curly braces {}.

Accessing, adding, updating, and deleting dictionary elements:

Elements can be accessed using keys, new pairs can be added, existing ones updated, and removed using del or pop().

Dictionary methods:

• keys(): Returns all keys in the dictionary.
• values(): Returns all values in the dictionary.
• items(): Returns key-value pairs as tuples.

7. Working with Dictionaries

Iterating over a dictionary using loops:

You can iterate through keys, values, or key-value pairs using loops such as for key in dict: or for key, value in dict.items().

Merging two lists into a dictionary:

Two lists can be combined into a dictionary using the zip() function or loops, where one list provides keys and the other provides values.

Counting occurrences of characters in a string using dictionaries:

A dictionary can be used to count occurrences by setting characters as keys and incrementing their counts as values.

8. Functions

Defining functions in Python:

Functions are defined using the 'def' keyword. They help organize code into reusable blocks. Syntax: def function_name(parameters):

Different types of functions:

• With parameters and with return value.
• With parameters and without return value.
• Without parameters and with return value.
• Without parameters and without return value.

Anonymous functions (lambda functions):

Lambda functions are single-line anonymous functions defined using the 'lambda' keyword. Syntax: lambda arguments: expression

9. Modules

Introduction to Python modules and importing modules:

Modules are files containing Python code (functions, variables, classes) that can be reused in other programs. They are imported using the 'import' statement.

Standard library modules: math, random:

Python’s standard library provides modules like math for mathematical operations and random for generating random numbers.

Creating custom modules:

Custom modules are user-defined Python files that can be imported into other scripts. A module file typically contains functions and constants related to a specific task.