Module 7) Python – Collections, functions and Modules

Understanding how to create and access elements in a list.
We can create a list by adding elements inside of [] brackets or we can use list constructor to create them. We can access the list items using their index positions [0], [1], etc.

Indexing in lists (positive and negative indexing).
Indexing in lists helps us to get the element on that particular position using the index number. [0], [1], etc. Python also supports negative indexing [-1], [-2], etc.

Slicing a list: accessing a range of elements.
Slicing in python helps us get a range of elements from a starting position to ending position minus 1. Eg: [2:6], [-3, 5], etc.

Common list operations: concatenation, repetition, membership.
Concatenation operator (+) is used to join two lists and form a new list. Repetiton operator (*) is used to repeat the list elements and form a new list, membership operator is used to find a specific element from the list of elements.

Understanding list methods like append(), insert(), remove(), pop().
Append method is used to append a new element at the end of a list. Insert method inserts a new element at the provided index. Remove method removes a specific element based on the element provided. Pop method pops an element based on index provided.

Iterating over a list using loops.
Iterating over a list means going through each element one by one using loops. We can use a for loop to directly access each item, a range-based for loop to access elements by index, or a while loop to iterate using a counter variable.

Sorting and reversing a list using sort(), sorted(), and reverse().
The sort() method sorts the list in place in ascending order by default. The sorted() function returns a new sorted list without changing the original one. The reverse() method reverses the order of elements in the list.

Basic list manipulations: addition, deletion, updating, and slicing.
Addition can be done using append() to add at the end or insert() to add at a specific index. Deletion can be done using remove(), pop(), or del. Updating is done by assigning a new value to a specific index. Slicing allows accessing or modifying a specific range of elements in the list.

Introduction to tuples, immutability.
Tuples are ordered collections of elements similar to lists but enclosed in parentheses (). They are immutable, meaning their elements cannot be changed, added, or removed after creation.

Creating and accessing elements in a tuple.
Tuples can be created by placing elements inside parentheses (), separated by commas. Elements can be accessed using their index positions like [0], [1], etc., similar to lists.

Basic operations with tuples: concatenation, repetition, membership.
Concatenation (+) joins two tuples to form a new one. Repetition (*) repeats the elements of a tuple a given number of times. Membership operators (in, not in) are used to check if an element exists within the tuple.

Accessing tuple elements using positive and negative indexing.
Positive indexing starts from 0 and accesses elements from the beginning, like [0], [1], etc. Negative indexing starts from -1 and accesses elements from the end, like [-1], [-2], etc.

Slicing a tuple to access ranges of elements.
Tuple slicing allows accessing a range of elements using start and end positions with the syntax [start:end], which returns elements from the start index up to but not including the end index. Negative indices can also be used.

Introduction to dictionaries: key-value pairs.
Dictionaries are collections of data stored as key-value pairs, enclosed in curly braces {}. Each key is unique and maps to a value.

Accessing, adding, updating, and deleting dictionary elements.
Elements can be accessed using their key, added or updated by assigning a value to a key, and deleted using del or pop().

Dictionary methods like keys(), values(), and items().
The keys() method returns all keys, values() returns all values, and items() returns key-value pairs as tuples.

Iterating over a dictionary using loops.
We can loop through a dictionary using for loops to access keys, values, or both using items().

Merging two lists into a dictionary using loops or zip().

Two lists can be combined into a dictionary by pairing elements as key-value using a for loop or the zip() function.

Counting occurrences of characters in a string using dictionaries.
A dictionary can store characters as keys and their counts as values, updating the count each time a character appears in the string.

Defining functions in Python.
Functions are defined using the def keyword followed by a name and parentheses. They group code into reusable blocks that can be called when needed.

Different types of functions: with/without parameters, with/without return values.
Functions can have parameters to accept input and can return values using the return statement. Functions without parameters or return values simply perform actions without inputs or outputs.

Anonymous functions (lambda functions).
Lambda functions are small, single-line functions defined using the lambda keyword, often used for quick operations.

Introduction to Python modules and importing modules.
Modules are files containing Python code that can be reused. They can be imported using the import statement.

Standard library modules: math, random.
The math module provides mathematical functions, while the random module is used for generating random numbers and choices.

Creating custom modules.
Custom modules are Python files created by the user that can be imported into other programs to reuse functions or variables.