Module -7

# Python – Collections, functions and Modules

Accessing List

1. • Understanding how to create and access elements in a list.

Elements in a list can be accessed using indexing. Python indexes start at 0, so a[0] will access the first element, while negative indexing allows us to access elements from the end of the list. Like index -1 represents the last elements of list.

1. • Indexing in lists (positive and negative indexing).

Positive indexing starts from index 0 for the first list item, adding up. So index 0, 1, 2, 3… for the first, second, third, fourth items etc. Negative indexing starts from index -1 for the last list item, subtracting down in reverse fashion toward the first item.

1. • Slicing a list: accessing a range of elements.

The syntax for accessing the elements of a list is the same as the syntax for accessing the characters of a string. We use the index operator ( [] – not to be confused with an empty list). The expression inside the brackets specifies the index. Remember that the indices start at 0.

# 2. List Operations

1. Common list operations: concatenation, repetition, membership.

* Concatenation operator (+) The (+) operator is used to add to two lists. ...
* Repetition operator (\*) Like string, (\*) operator replicates the string number of specified times. ...
* List Slicing in Python. List slicing returns a slice or part of the list from the given index range x to y. ...
* Membership Operator (in, not in)

1. • Understanding list methods like append(), insert(), remove(), pop().

insert(): Inserts an element at a specified position.

pop(): Removes and returns the element at the specified position (or the last element if no index is specified).

remove(): Removes the first occurrence of a specified element. reverse(): Reverses the order of the elements in the list.

# 3. Working with Lists

1. • Iterating over a list using loops

The simplest and the most common way to iterate over a list is to use a for loop. This method allows us to access each element in the list directly.

2>• Sorting and reversing a list using sort(), sorted(), and reverse().

The sorted() function returns a sorted list of the specified iterable object.

Reverse sorting means organizing the characters in descending order

1. **B**asic list manipulations: addition, deletion, updating, and slicing.

Addition :

Python also provides inbuilt functions that help in addition. If we want to add elements to the list we can use the sum() function. They add the elements to the list. We can pass the variable name to the function, And it will return the sum of all the elements of the list.

Deletion :

n Python, "basic deletion" refers to the act of removing an element from a data structure like a list or dictionary using the del keyword, which essentially breaks the reference to that element, effectively deleting it from memory; you can delete elements by index, slice, or key depending on the data type involved.

Updating :

In Python, "basic updating" refers to the process of changing the value of a variable or modifying elements within a data structure like a list or dictionary, usually done by directly assigning a new value to the variable or using specific methods like .update() depending on the data type involved; essentially, it means updating existing data with new information.

Slicing :

Slicing is the extraction of a part of a string, list, or tuple. It enables users to access the specific range of elements by mentioning their indices.

# 4. Tuple

* 1. • Introduction to tuples, immutability.
* Immutability: One of the key differences between tuples and other Python data structures is that tuples are immutable. Once a tuple is created, its contents cannot be changed. In contrast, lists and dictionaries are mutable, meaning that their contents can be modified after they are created.

2>• Creating and accessing elements in a tuple.

Python tuple can be created by specifying comma separated values inside of parentheses ( ) . Values inside of a tuple cannot be modified once created. Let's create a tuple of the first 5 odd numbers and then try to change one of them to be a number that is not odd.

1. • Basic operations with tuple: concatenation, repetition, membership

In Python, basic operations with tuples include concatenation (joining two tuples together using the "+" operator), repetition (repeating a tuple a specified number of times using the "\*" operator), and membership (checking if a specific element exists within a tuple using the "in" operator).

# 5. Accessing Tuples’

1>• Accessing tuple elements using positive and negative indexing.

Positive Indexing: Starts from 0 and goes up to n-1 (where n is the length of the sequence). Negative Indexing: Starts from -1 for the last element and goes up to -n for the first element.

2>• Slicing a tuple to access ranges of elements.

Slicing refers to an operator that fetches a sub-collection from an entire collection by slicing it from the start to stop index. Slicing in a tuple aims to fetch a specific range or slice of sub-elements from the tuple. It works the same way as a String slicing or any other sequence of elements.

# 6. Dictionaries

1>• Introduction to dictionaries: key-value pairs

Dictionaries are used to store data values in key:value pairs. A dictionary is a collection which is ordered\*, changeable and do not allow duplicates.

A key-value pair is a data type that includes two pieces of data that have a set of associated values and a group of key identifiers. Within a key-value pair, there are two related data elements. The first is a constant used to define the data set . The other is a value, which is a variable belonging to the data set.

2>• Accessing, adding, updating, and deleting dictionary elements.

Accessing :

The syntax for accessing the elements of a list is the same as the syntax for accessing the characters of a string. We use the index operator ( [] – not to be confused with an empty list). The expression inside the brackets specifies the index. Remember that the indices start at 0.

Adding :

You can use the = assignment operator to add a new key to a dictionary: dict[key] = value. If a key already exists in the dictionary, then the assignment operator updates, or overwrites, the value.

Updating :

: The update() method is used to modify existing key-value pairs or add new pairs from another dictionary. When using update() , if the key already exists in the dictionary, its value will be updated; if it does not exist, the key-value pair will be added.

Deleting :

Python's del statement is used to delete variables and objects in the Python program. Iterable objects such as user-defined objects, lists, set, tuple, dictionary, variables defined by the user, etc. can be deleted from existence and from the memory locations in Python using the del statement.

3>• Dictionary methods like keys(), values(), and items().

Keys() :

Returns a view of the keys in the dictionary.

Values() :

Returns a view of the values in the dictionary

Items():

Returns a view of the key-value pairs in the dictionary

# 7. Working with Dictionaries

* 1. • Iterating over a dictionary using loops.

You can directly iterate over the keys of a Python dictionary using a for loop and access values with dict\_object[key] . You can iterate through a Python dictionary in different ways using the dictionary methods . keys() , . values()

2>• Merging two lists into a dictionary using loops or zip()

Iterate through both lists simultaneously using zip and for each pair, add the first element as the key and second as the value to the dictionary. Explanation: An empty dictionary res is created, and zip(a, b) is used to iterate through both lists yielding pairs of keys and values.

3>• Counting occurrences of characters in a string using dictionaries

**Logic:**

1. Prompt the user to enter a string.
2. Create an empty dictionary called “dic”.
3. Iterate through each character in the given string using a for loop.
4. Check if the character already exists in the “dic” dictionary. ...
5. If the character is not yet in the dictionary, add it as a new key with a value of 1.

# 8. Functions

1. • Defining functions in Python.

The four steps to defining a function in Python are the following: Use the keyword def to declare the function and follow this up with the function name. Add parameters to the function: they should be within the parentheses of the function. End your line with a colon. Add statements that the functions should execute.

2>• Different types of functions: with/without parameters, with/without return values

**Function with parameters and a return value:** This type of function accepts input values (parameters) and performs calculations on them, then returns a single output value to the calling code.

**Function with parameters but no return value (void function):** This function takes input parameters but does not explicitly return any value; it usually performs an action like printing to the console or modifying data directly.

3>• Anonymous functions (lambda functions).

The expression in the lambda function multiplies every item in the list (doubles) and returns its value to final\_li. In conclusion, the Python lambda (anonymous) function is a no-name function declared in a single line. It can have only one expression and is used when a short-term function is required.

# 9. Modules :

1>• Introduction to Python modules and importing modules.

* A Python module is a single file (.py) containing Python code that can be organized into functions, classes, and variables, acting as a self-contained unit of code.
* **Standard Library modules:**

Python comes with a built-in set of modules (like math, random, datetime) that you can directly import and use in your programs.

* **Creating your own modules:**

You can also create your own modules by writing Python code in a separate file, which can then be imported into other scripts.

* **Import statement:**

To use a module in your code, you use the import keyword followed by the module name.

Example of importing a module:

* **Module search path:**

When you import a module, Python searches for the corresponding file in a list of directories specified in the sys.path variable.

* **Importing specific parts of a module:**

To avoid cluttering your namespace, you can import only specific functions or classes from a module using the from ... import syntax:

2>• Standard library modules: math, random

Math module

* Provides access to mathematical functions for floating-point math, such as trigonometric functions, logarithms, and other common mathematical operations
* Functions similarly to a digital calculator
* Gives access to the underlying C library functions for floating-point math
* Includes functions like:
  + sqrt() Returns the square root of a number
  + atan() Returns the arc tangent of a number in radians
  + atan2() Returns the arc tangent of y/x in radians
  + atanh() Returns the inverse hyperbolic tangent of a number
  + ceil() Rounds a number up to the nearest integer

Random module

* Generates pseudo-random numbers
* Used for simulations, games, random sampling, and other applications where unpredictability is required
* Includes functions like:
  + random() Generates a random float value ranging from 0.0 to 1.0
  + randint(start, stop) Returns a random integer in [start, stop]
  + randrange(start, stop[,step]) Returns a random element in [start, stop] with [step] being optional
  + shuffle(x) Shuffles the sequence x
  + 3> Creating custom modules.
  + o create a Python module, write the desired code and save that in a file with .py extension. Let's understand it better with an example: Example: Let's create a simple calc.py in which we define two functions, one add and another subtract.