Module – 3 (Collections, functions and Modules)

1. What is List? How will you reverse a list?

- A list is a fundamental data structure in Python used to store collections of items. Lists are ordered, mutable (can be changed), and can contain elements of different types, including numbers, strings, and other objects. Lists are defined using square brackets `[]`, and elements inside a list are separated by commas.
- A **list** is any information displayed or organized in a logical or linear formation. Below is an example of a numerical list that shows several steps that need to be performed to accomplish something.
    - First step.
    - Second step.
    - Third and final step.
- Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.
- The **`reverse()`** method is an in-place method that reverses the elements of the list in its original place without creating a new list.
- original_list = [1, 2, 3, 4, 5]

  original_list.reverse()

  print(original_list)

3. Differentiate between append () and extend () methods?

- **`append()` Method:**
    - ❖ `append()` is used to add a single element (item) to the end of a list.
    - ❖ The element to be added is passed as an argument to the `append()` method.
    - ❖ The original list is modified in place, and the method returns '`None`'.

- Python's append() function inserts a single element into an existing list. The element will be added to the end of the old list rather than being returned to a new list. Adds its argument as a single element to the end of a list. The length of the list increases by one.

- **`extend()` Method:**
    - ❖ `extend()` is used to add multiple elements (items) from an iterable (such as a list, tuple, or string) to the end of a list.
    - ❖ The iterable containing elements is passed as an argument to the `extend()` method.
    - ❖ The original list is modified in place.

- The Python's List ***extend()*** Iterates over its argument and adding each element to the list and extending the list. The length of the list increases by a number of elements in its argument.

➢ `append()` is used to add a single element to the end of a list, while `extend()` is used to add multiple elements from an iterable to the end of a list. The key difference is in the number of elements they add and the type of argument they accept.

Basis for Comparison	**Append()**	**Extend()**
Purpose	To add a single entry to the end of a list, use the append() function.	To add additional elements or an iterable to the end of a list, use the extend() function.
Input	accepts only one input element.	accepts as input an iterable (such as a list or tuple).
Operation	The append() function adds the full input to the list as a single item.	extend() adds each item to the list independently after iterating through each one in the input.
Efficiency	Since append() only executes one operation, it is typically quicker and more effective than extend().	When adding elements from numerous iterables or with huge inputs, extend() could take longer.
Time complexity	**Append** has constant time complexity i.e.,O(1)	**Extend** has a time complexity of O(k). Where k is the length of the list which need to be added.

5. How will you compare two lists?

- To check whether two lists contain the same elements or not, we can use the sort() method to sort the elements of the lists first. Then, we can compare the two lists.
- sort() method or the sorted() function with the ==operator
- set() functionswith the ==operator
- Using the sort() Method or the sorted() Function to Compare Lists
- You can use the sort() method or the sorted() function to sort lists with the purpose of comparing them for equality. The sort() method sortsthe list in place, while the sorted()function returns a new list. After sorting, lists that are equal will have the same items in the same index positions. The == operator compares the lists, item by item (element-wise comparison).
- The order of the original list items is not important, because the lists are sorted before comparison.

18. What is tuple? Difference between list and tuple.

- Tuple is collections of things or entry which non changeable in nature and can be defined or observed as () closed brackets using a variable.
- E.g. x = (1, 2, 4, dhrutidhar)
- The list is dynamic, whereas the tuple has static characteristics. This means that lists can be modified whereas tuples cannot be modified, the tuple is faster than the list because of static in nature. Lists are denoted by the square brackets but tuples are denoted as parenthesis.

Differences between List and Tuple in Python

Sno	LIST	TUPLE
1	Lists are mutable	Tuples are immutable
2	The implication of iterations is Time-consuming	The implication of iterations is comparatively Faster
3	The list is better for performing operations, such as insertion and deletion.	A Tuple data type is appropriate for accessing the elements
4	Lists consume more memory	Tuple consumes less memory as compared to the list
5	Lists have several built-in methods	Tuple does not have many built-in methods.
6	Unexpected changes and errors are more likely to occur	In a tuple, it is hard to take place.

35. How Do You Traverse Through A Dictionary Object in Python?

- Dictionary in Python is a collection of data values, used to store data values like a map, unlike other Data Types that hold only a single value as an element, Dictionary holds the key: value pair.
- There are multiple ways to iterate over a dictionary in Python.
    - Access key using the build .keys()
    - Access key without using a key()

- Iterate through all values using .values()
- Iterate through all key, and value pairs using items()
- Access both key and value without using items()
- Print items in Key-Value in pair

➢ You can loop through a dictionary by using a `for` loop.

➢ When looping through a dictionary, the return value are the *keys* of the dictionary, but there are methods to return the *values* as well.

36. How Do You Check The Presence Of A Key In A Dictionary?

➢ **There can be different ways for checking if the key already exists, we have covered the following approaches:**

- Using the Inbuilt method keys()
- Using if and in
- Using has_key() method
- Using get() method

➢ By using build .keys() functions.

- e.g. = states_And_Capitals ={ 'Gujarat': 'Gandhinagar', 'Maharashtra': 'Mumbai', 'Rajasthan': 'Jaipur', 'Bihar': 'Patna'}
  keys = states_And_Capitals.keys()
  print(keys)
  O/P = ['Gujarat', 'Maharashtra', 'Rajasthan', 'Bihar']

➢ By using IF and IN Keyword.

- e.g. = my_dict = {‘a’: 1 , ‘b’: 2, ‘c’: 3}

- key_to_check = ‘b’
- if key_to_check in my_dict:
- print(“This key is present in Dictionary.”)
- else:
- print(“This key is not present in Dictionary.”)

43. Why Do You Use the Zip () Method in Python?

- The `zip()` function in Python is used to combine elements from multiple iterables (e.g., lists, tuples, or strings) into tuples. It returns an iterator of tuples where the i-th tuple contains the i-th element from each of the input iterables. This is particularly useful when you want to iterate over multiple iterables simultaneously.
- Python **zip() method** takes iterable containers and returns a single iterator object, having mapped values from all the containers.
- It is used to map the similar index of multiple containers so that they can be used just using a single entity.
- The zip() function is used to combine two or more lists (or any other iterables) into a single iterable, where elements from corresponding positions are paired together.

- The zip() function returns a zip object, which is an iterator of tuples where the first item in each passed iterator is paired together, and then the second item in each passed iterator are paired together etc.
- e.g.: a = (1,2,3) and b = (‘a’, ‘b’, ‘c’)
  - c = zip (a, b)
  - print(c)
  - O/P = (1: ‘a’, 2: ‘b’, 3: ‘c’)

52. How Many Basic Types Of Functions Are Available In Python?

- There are two types of functions in python: User-Defined Functions -these types of functions are defined by the user to perform any specific task. Built-in Functions - These are pre-defined functions in python.
    - Built-in library function: e.g.: print(), len(), type(), etc
    - User define can be written as “def function name():”
- In Python, there are several types of functions, but they can be broadly categorized into two main types: built-in functions and user-defined functions.

1. **Built-in Functions:**
    - These are functions that are pre-defined in Python and are available for use without the need for explicit declaration or definition. Examples include `print()`, `len()`, `type()`, `int()`, `float()`, and many more. Python's standard library provides a wide range of built-in functions for various tasks.
2. **User-Defined Functions:**
    - These are functions that are defined by the user to perform a specific task. User-defined functions allow you to modularize your code and reuse it in different parts of your program. You define your function using the `def` keyword.

- In addition to standard user-defined functions, there are also lambda functions (anonymous functions) created using the `lambda` keyword. Lambda functions are typically used for short, simple operations.

53. How can you pick a random item from a list or tuple?

- The random item can be picked up using importing random module and using it.
  - E.g.: a = [1,3,5,2,55]
  - Import random
  - Print(random.choice(a))

- This code uses the **`random.choice()`** function from the **`random`** module to pick a random item from the list **`my_list`**. Other languages might have similar functions or methods to achieve the same result.

```
my_tuple = (6, 7, 8, 9, 10)
random_item = random.choice(my_tuple)
print(random_item)
```

54. How can you pick a random item from a range?

- Using random.randrange() function
- Using random.randint() function
- Using random.random() function
- Using random.sample()

- To pick a random item from a range in Python, you can use the **`random.choice()`** function as well. However, since **`range`** is not directly iterable, you need to convert it to a list or tuple before using **`random.choice()`**. Here's an example:
  - `import random`
  - `my_range = range(1, 11)`
  - `random_item = random.choice(list(my_range))`
  - `print(random_item)`

- `range(1, 11)` generates a range from 1 to 10. We convert it to a list using `list(my_range)`, and then we use `random.choice()` to pick a random item from that list.

55. How can you get a random number in python?

- Import random
- N = random.randint(1111,9999) #range
- Print(n)
- O/P: 3424

- This are the functions to get a random number.

- `random.randint()` function
- `random.randrange()` function
- `random.sample()` function
- `random.uniform()` function

56. How will you set the starting value in generating random numbers?

- import random
- n = random.randint(0,100)
- print(n)
- O/P: 77

- **seed[x] – Sets the integer starting value used in generating random numbers. Call this function before calling any other random module function. Returns None.**
- If you want to set the starting value (seed) for generating random numbers in Python, you can use the `random.seed()` function from the `random` module. The seed is an initial value used to initialize the random number generator.

-
  ```python
  import random
  ```

-
  ```python
  random.seed(42)
  ```

-
  ```python
  random_number_1 = random.random()
  ```

-
  ```python
  random_number_2 = random.uniform(1.0, 10.0)
  ```

-
  ```python
  random_integer = random.randint(1, 100)
  ```

-
  ```python
  print(random_number_1)
  ```

-
  ```python
  print(random_number_2)
  ```

-
  ```python
  print(random_integer)
  ```

- By setting the seed with `random.seed(42)`, you ensure that if you run the program multiple times, you'll get the same sequence of random numbers. This can be useful for reproducibility in situations where you want the randomness to be consistent across runs.

- Keep in mind that setting the seed is optional. If you don't set it, Python uses the current system time as the default seed, resulting in a different sequence of random numbers each time you run the program.