Module – 3

(Collections, functions and Modules)

- What is List? How will you reverse a list?

- Lists are used to store multiple items in a single variable.
- 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.
- Lists in python are most used data structures.
- we need to reverse a list, delete an element from the list, adding or inserting an element to the list or to merge two lists in python.
- Python comes with a number of methods and functions that allow you to reverse a list, either directly or by iterating over the list object.
- You'll learn how to reverse a Python list by using the `reversed()` function, the `.reverse()` method, list indexing, for loops, list comprehensions, and the slice method

- How will you remove last object from a list? Suppose list1 is [2, 33, 222, 14, and 25], what is list1 [-1]?

- In Python, you can remove the last object from a list using the pop() method.
- The pop() method removes the item at the specified index Here's an example:

```
    list1 = [2, 33, 222, 14, 25]
last_item = list1.pop()
print("Removed item:", last_item)
```

```
print("Updated list:", list1)
```

- # Differentiate between append () and extend ()methods?

- append() method:
    - append() is used to add a single element to the end of a list.
    - The syntax is list.append(element)
    - It modifies the original list by adding the specified element to the end.

- extend() method:
    - extend() is used to append the elements of an iterable (e.g., a list, tuple, or string) to the end of the list.
    - The syntax is list.extend(iterable).
    - It modifies the original list by adding all the elements of the iterable to the end.
- append() adds a single element to the end of a list.
- extend() adds all the elements from an iterable to the end of a list.

- # How will you compare two lists?

- Comparing two lists involves assessing their similarities, differences, and relationships. Depending on the context, you may want to check for elements that are common to both lists, elements unique to each list, or the overall equality of the lists. Here are several ways to compare two lists:
- Equality Check:
- Check if the two lists are exactly the same in terms of both elements and their order.

```
list1 = [1, 2, 3]
list2 = [1, 2, 3]
if list1 == list2:
   print("The lists are equal.")
```

- Set Comparison:
- Convert the lists to sets and compare them to find common and unique elements

```
set1 = set(list1)
set2 = set(list2)

common_elements = set1.intersection(set2)
unique_elements_list1 = set1.difference(set2)
unique_elements_list2 = set2.difference(set1)
```

• How will you create a dictionary using tuples in python?

- In Python, you can create a dictionary using tuples as keys by using tuples as the keys in the dictionary.

```
my_dict = {('apple', 3): 'red', ('banana', 5): 'yellow',
('grape', 10): 'purple'}

print(my_dict[('apple', 3)])  # Output: red
print(my_dict[('banana', 5)])  # Output: yellow
print(my_dict[('grape', 10)])  # Output: purple

my_dict[('orange', 8)] = 'orange'

print(my_dict)
```

- In this example, each key in the dictionary is a tuple, and the corresponding values are strings representing the color of the fruit. You can access, modify, or add elements to the dictionary using tuples as keys just like you would with any other dictionary in Python.

• How Do You Traverse Through A Dictionary Object In Python?

- In Python, you can traverse through a dictionary using various methods. Here are some common ways:
  - Using a for loop:

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
for key in my_dict:
    value = my_dict[key]
    # Do something with key and value
```

  - Using items() method:

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
for key, value in my_dict.items():
    # Do something with key and value
```

- Etc.
- Choose the method that best fits your needs and the specific operation you want to perform while traversing the dictionary

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

- Python, you can check the presence of a key in a dictionary using the in keyword or the get() method. Here's how you can do it:

- Using the in keyword

```
my_dict = {'name': 'John', 'age': 25, 'city': 'New York'}
if 'age' in my_dict:
    print("Key 'age' is present.")
else:
    print("Key 'age' is not present.")
```

- Choose the method that fits your needs and coding style.

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

- The zip() method in Python is used to combine multiple iterables (such as lists, tuples, or strings) element-wise. It creates an iterator that generates tuples containing elements from the input iterables, where the i-th tuple contains the i-th element from each of the input iterables.
- By using zip(), you can efficiently iterate over multiple sequences simultaneously and process corresponding elements together. This is particularly useful when you want to pair elements from different iterables or perform operations on elements at the same position in multiple iterables.
- In summary, zip() facilitates parallel iteration and grouping of elements from multiple iterables.

• How Many Basic Types Of Functions Are Available In Python?

- Python, there are several basic types of functions, each serving a specific purpose. Here are some of the common types:

- Built-in Functions: These are functions that are part of the Python language itself and can be used without importing any module. Examples include print(), len(), type(), etc.
- User-Defined Functions: These are functions created by the user to perform a specific task. You can define your functions using the def keyword.
- Anonymous Functions (Lambda Functions): These are small, unnamed functions defined using the lambda keyword. They are often used for short, simple operations.
- Higher-Order Functions: Functions that take other functions as arguments or return functions. Examples include map(), filter(), and reduce().
- These are some of the basic types of functions in Python. Each serves a different purpose and can be used in various scenarios based on the requirements of your program.

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

- In Python, you can use the random.choice() function from the random module to pick a random item from a list or tuple. Here's an example:

```
import random
my_list = [1, 2, 3, 4, 5]
random_item = random.choice(my_list)
print("Random item:", random_item)
```

- This code selects a random item from the my_list and prints it. Similarly, you can apply the same method to a tuple:

```
my_tuple = (10, 20, 30, 40, 50)
random_item_from_tuple = random.choice(my_tuple)
```

```
print("Random item from tuple:", random_item_from_tuple)
```

- By using random.choice(), you ensure that each item in the list or tuple has an equal probability of being selected.

• How can you pick a random item from a range?

- To pick a random item from a range in various programming languages, you can use language-specific functions or methods. Here are examples in a few popular programming languages:
- Python:
- In Python, you can use the random.choice() function from the random module.

```
import random

my_range = range(1, 11)  # Example range from 1 to 10
random_item = random.choice(my_range)

print(random_item)
```

- JavaScript:
- In JavaScript, you can use Math.random() to generate a random number and then use it to index into the array.

```
var myRange = Array.from({length: 10}, (_, i) => i + 1);  // Example range from 1 to 10
var randomItem = myRange[Math.floor(Math.random() * myRange.length)];

console.log(randomItem);
```

• How can you get a random number in python?

- To generate a random number in Python, you can use the random module. Here's a simple example:
- python
- import random

```
random_number = random.randint(1, 100)
print(random_number)
```

- This code uses the randint function from the random module to generate a random integer between 1 and 100 (inclusive). You can adjust the range as needed for your specific use case.

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

- In programming, the starting value for generating random numbers is typically referred to as the seed. The seed is used to initialize the random number generator (RNG) algorithm. Setting the seed to a specific value ensures that the sequence of random numbers generated is reproducible. If you use the same seed, you will get the same sequence of random numbers each time you run your program.
- The method for setting the seed depends on the programming language you're using. I'll provide examples in a few common programming languages:
- Python (using the random module):

```
import random

# Set the seed
random.seed(42)
# Generate random numbers
rand_num = random.random()
```

- How will you randomizes the items of a list in place?

- To randomize the items of a list in place in Python, you can use the shuffle function from the random module. Here's an example:

```
import random
my_list = [1, 2, 3, 4, 5]
random.shuffle(my_list
print(my_list)
```

- In this example, the shuffle function is applied directly to the list my_list, and it modifies the list in place by rearranging its elements randomly.
- Keep in mind that the shuffle function modifies the original list and doesn't return a new list.
- Remember to import the random module before using the shuffle function.