# Module – 3 (Collections, functions and Modules)

## Q-1: What is List? How will you reverse a list?

A list is a collection of different kinds of values or items. Python lists are mutable, we can change their elements after Creating

### How to Reverse a List

To reverse a list, you can do it in a few simple ways:

1. **Using `reverse ()` Method**: This changes the original list to be in reverse order

```
my_list = [1, 2, 3]
my_list.reverse()
print(my_list)
```

2. **Using Slicing**: This creates a new reversed list.

```
my_list = [1, 2, 3]
reversed_list = my_list[::-1]
print(reversed_list)
```

3. **Using `reversed ()` Function**:

```
my_list = [1, 2, 3]
reversed_list = list(reversed(my_list))
print(reversed list)
```

## Q-2: How will you remove last object from a list?

To remove the last item from a list, use the pop() method. Here's a simple example:

```
my_list = [1, 2, 3, 4, 5]

my_list.pop()

print(my_list)
```

- **pop () Method**: This method removes the item at the end of the list by default and updates the list.
- **Result**: The `my_list` will be updated to `[1, 2, 3, 4]` after removing the last item, which was 5.

## Q-3: Differentiate between append () and extend () methods?

### `append()`

- **What It Does**: Adds just one item to the end of the list.
- **When to Use It**: Use it when you want to add a single element, like a number or string, or even another list.

### `extend()`

- **What It Does**: Adds each element from another iterable (like a list or string) to the end of the list.
- **When to Use It**: Use it when you want to add multiple items from another iterable, not as one item but as separate elements.

## Q_5: How will you compare two lists?

### 1. Check for Equality

You can see if two lists are exactly the same by checking if they have the same elements in the same order. If both lists are identical in terms of content and sequence, they are considered equal.

### 2. Compare Elements (Ignoring Order)

If the order of elements doesn't matter, you can check if two lists contain the same elements. This can be done by sorting both lists first and then comparing them, or by converting them to sets and comparing the sets. If the lists contain the same elements regardless of order, they are considered equivalent.

### 3. Check for Subset

To find out if all elements of one list are present in another list, you can see if one list is a subset of the other. This means checking if every element in the smaller list exists in the larger list.

### 4. Element-wise Comparison

For a more detailed comparison, you can look at each element in the lists one by one. This can help you identify which specific elements are different if the lists are not identical.

These methods allow you to compare lists in different ways depending on whether you care about the order of elements or if you just want to know if they contain the same items.

## Q-18: What is tuple? Difference between list and tuple.

A tuple in Python is an immutable collection of items, meaning once it's created, you can't change its contents. Tuples are defined with parentheses `()`.

**Example**: `(1, 2, 3, 'apple')`

### Differences Between List and Tuple

4. **Mutability**:
   - **List**: Can be changed after creation (you can add, remove, or modify elements).
   - **Tuple**: Cannot be changed after creation (it's fixed).

5. **Syntax**:
    - **List**: Created with square brackets `[]`.
    - **Tuple**: Created with parentheses `()`.
6. **Performance**:
    - **List**: Typically slower due to its ability to change.
    - **Tuple**: Generally faster and uses less memory since it's immutable.
7. **Methods**:
    - **List**: Has many methods to modify its contents.
    - **Tuple**: Has fewer methods since it can't be changed.

## Q-35: How Do You Traverse Through A Dictionary Object In Python?

To traverse through a dictionary in Python, you can use different methods to access its keys, values, or both. Here's how you can do it:

### 1. Traversing Keys

To loop through the keys of a dictionary, you can use the `.keys()` method or simply iterate over the dictionary directly.

```
my_dict = {'a': 1, 'b': 2, 'c': 3}

for key in my_dict.keys():
    print(key)

for key in my_dict:
    print(key)
```

### 2. Traversing Values

To loop through the values in a dictionary, use the `.values()` method.

```python
my_dict = {'a': 1, 'b': 2, 'c': 3}

for value in my_dict.values():
    print(value)
```

### 3. Traversing Keys and Values

To loop through both keys and values, use the `.items()` method, which returns key-value pairs.

```python
my_dict = {'a': 1, 'b': 2, 'c': 3}

for key, value in my_dict.items():
    print(f"Key: {key}, Value: {value}")
```

## Q-36: How Do You Check the Presence of a Key in A Dictionary?

To check if a key is present in a dictionary, you can use several methods:

### 1. Using the in Operator

The most common and straightforward way is to use the `in` operator. It checks if the key exists in the dictionary and returns `True` or `False`.

```python
my_dict = {'a': 1, 'b': 2, 'c': 3}

if 'a' in my_dict:
    print("Key 'a' is present.")
else:
    print("Key 'a' is not present.")
```

## 2. Using the `.get()` Method

You can also use the `.get()` method. If the key exists, `.get()` returns its value; if not, it returns None (or a default value if specified).

```python
my_dict = {'a': 1, 'b': 2, 'c': 3}

if my_dict.get('a') is not None:
    print("Key 'a' is present.")
else:
    print("Key 'a' is not present.")
```

## 3. Using the `.keys()` Method

Although less common for this specific task, you can check if a key is in the dictionary by checking if it is in the dictionary's keys.

```python
my_dict = {'a': 1, 'b': 2, 'c': 3}

if 'a' in my_dict.keys():
    print("Key 'a' is present.")
else:
    print("Key 'a' is not present.")
```

## Q-43: Why Do You Use the Zip () Method in Python?

The `zip()` method in Python is used to combine multiple iterables (like lists or tuples) into a single iterable where each element is a tuple containing elements from the corresponding position in each iterable. It's useful for parallel iteration and grouping related data.

**Parallel Iteration**

- **Purpose**: Allows you to loop through multiple sequences at the same time.
- **Example**:

```
names = ['Alice', 'Bob', 'Charlie']
scores = [85, 90, 88]

for name, score in zip(names, scores):
    print(f"{name}: {score}")
```

### Creating Dictionaries

- **Purpose**: Combine two lists into a dictionary where one list represents keys and the other represents values.
- **Example**

```
keys = ['name', 'age', 'city']
values = ['Alice', 30, 'New York']

my_dict = dict(zip(keys, values))
print(my_dict)
```

### Transposing Data

- **Purpose**: Switch rows and columns in a list of tuples or a list of lists.
- **Example**

```
matrix = [(1, 2, 3), (4, 5, 6), (7, 8, 9)]

transposed = list(zip(*matrix))
print(transposed)
```

## Q-52: How Many Basic Types Of Functions Are Available In Python?

In Python, there are generally **three basic types of functions** that you can define and use:

- **Built-in Functions**:
  - **Definition**: These are functions that come pre-defined with Python and are available for use without needing to import any modules.
  - **Examples**: print(), len(), type(), max(), min()
  - **Usage**: You use these functions directly in your code to perform common tasks.

- **User-Defined Functions**:
  - **Definition**: These are functions that you define yourself to perform specific tasks in your code.
  - **Syntax**: You use the def keyword to create them.
  - **Example**

```
def greet(name):
    return f"Hello, {name}!"
```

- **Lambda Functions**:
  - **Definition**: These are small, anonymous functions defined with the lambda keyword. They are often used for short, throwaway functions.
  - **Syntax**: The function is defined in a single line.
  - **Example**

```
add = lambda x, y: x + y
print(add(3, 5))  # Output: 8
```

## Q-53: How can you pick a random item from a list or tuple?

To pick a random item from a list or tuple in Python, you can use the random module, which provides functions for generating random numbers and making random selections.

### Using random.choice()

The random.choice() function can be used to select a random item from a non-empty sequence (like a list or tuple).

**Steps**:

1. Import the random module.
2. Use random.choice() with your list or tuple.

## Q-54: How can you pick a random item from a range?

To pick a random item from a range in Python, you can use the random module. Specifically, you'll use the random.randint() function to generate a random integer within the range, or random.randrange() if you want more control over the range's start, stop, and step.

### Using random.randint()

1. **Import the random module**.
2. **Generate a random integer** within the specified range using random.randint(start, end)

## Q-55: How can you get a random number in python?

To get a random number in Python, you can use the random module, which provides several functions for generating random numbers. Here's how you can get random numbers using different methods:

### 1. Random Integer

Use random.randint(start, end) to get a random integer within a specified range.

### 2. Random Choice from a Sequence

Use random.choice(sequence) to get a random element from a list, tuple, or other sequence types.

### 3. Random Number with Specific Step

Use random.randrange(start, stop, step) to get a random integer within a range and with a specific step.

## Q-56: How will you set the starting value in generating random numbers?

To set the starting value or seed for generating random numbers in Python, you use the random.seed() function. Setting a seed ensures that the sequence of random numbers generated is reproducible, which is useful for debugging or when you want consistent results across runs.

### How to Set the Seed

1. **Import the random module**.
2. **Call random.seed(seed_value)** with a specific seed value.

### Why Use a Seed?

- **Reproducibility**: Ensures that you get the same sequence of random numbers each time you run your code, which is important for debugging and testing.
- **Consistency**: Useful for experiments or simulations where you need consistent results

## Q-57: How will you randomizes the items of a list in place?

To randomize the items of a list in place in Python, you use the random.shuffle() function. This function modifies the original list by shuffling its items, meaning it changes the order of the elements randomly.

### Steps to Randomize a List

- **Import the random module**.
- **Call random.shuffle()** with your list as the argument.