Module 3

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

Ans: list is a built-in data structure that is used to store a collection of items. Lists are ordered, mutable, and can contain elements of different data types. You can think of a list as a sequence of values enclosed in square brackets [], with each value separated by a comma.

To reverse a list, you can use the follwing method:

1. Using List Slicing
2. Using reverse() Method
3. Using reversed() Function

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

Ans: To remove the last object from a list, you can use the 'pop()' method or list slicing from a list. If you don' need the removed item, you can simple call 'pop()' without specifying an index.

1. Using pop() Method:

This method is used to remove and return the lst item from a list.

If you don't need the removed item, you can simple call 'pop()' without specifying an index.

2 .Using List slicing:

You can also remove the last object from a list by creating a new list that excludes the last element.

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

Ans: append() and extend() are two methods used with lists to add elements, but they behave differently in terms of what they add and how they modify the list. Here's the difference between these two methods:

1. append() Method:
    - The append() method is used to add a single element to the end of a list.
    - It takes one argument, which is the element you want to add, and appends it as a single item to the list.
    - It does not modify the original element if the element being added is another list; it adds the entire list as a single item to the end of the list.

2. extend() method:
- The extend() method is used to add multiple elements to the end of a list.
- It takes an iterable (e.g., a list, tuple, or another iterable) as an argument and adds all the elements from the iterable to the list one by one.
- It modifies the original list in place by adding all the elements from the iterable.

Q:4 Write a Python function to get the largest number, smallest num and sum of all from a list.

Ans:

```
def get_stats(lst):
    if not lst:  # Check if the list is empty
```

```
        return None, None, 0

    largest = lst[0]  # Initialize with the first element
    smallest = lst[0]  # Initialize with the first element
    total = 0

    for num in lst:
        if num > largest:
            largest = num
        if num < smallest:
            smallest = num
        total += num

    return largest, smallest, total

# Example usage:
my_list = [12, 45, 7, 23, 56, 89, 34]
largest, smallest, total = get_stats(my_list)

print("Largest number:", largest)
print("Smallest number:", smallest)
print("Sum of all numbers:", total)
```

Q: 5 How will you compare two lists

Ans:

- **To compare two lists in Python, you can use various methods depending on the specific comparison you want to make. Here are some common ways to compare two lists:**
    1. **Element-wise Comparision:**
       If you want to compare whether the elements of two lists are the same, you can use the equality operator (==) to check if the two lists are equal element by element.
    2. **Identity Comparision:**
       If you want to check if two lists are the same object in memory, you can use the identity operator (‘is’).

3. Order-Insensitinve Comparision:
If you want to compare two lists regardless of the order of their elements, you can use the sorted() function to sort both lists and then compare them.

Q:6 Write a Python program to count the number of strings where the string length is 2 or more and the first and last character are same from a given list of strings.

Ans:

```
def count_strings_with_same_first_last_char(strings):
    count = 0

    for s in strings:
        if len(s) >= 2 and s[0] == s[-1]:
            count += 1

    return count

# Example usage:
string_list = ["aba", "xyz", "aa", "bb", "cdcdc", "bbcb"]
result = count_strings_with_same_first_last_char(string_list)
print("Number of strings with the same first and last character:", result)
```

Q: 7 Write a Python program to remove duplicates from a list.

Ans:

You can remove duplicates from a list in Python using various methods. One common approach is to convert the list to a set (which automatically removes duplicates) and then convert it back to a list.

```
def remove_duplicates(input_list):
    # Convert the list to a set to remove duplicates, and then back to a list
    unique_list = list(set(input_list))
    return unique_list

# Example usage:
my_list = [1, 2, 2, 3, 4, 4, 5]
```

```
result = remove_duplicates(my_list)
print("Original List:", my_list)
print("List with Duplicates Removed:", result)
```

Q:8 Write a Python program to check a list is empty or not.
Ans:

```
def is_empty(lst):
   return not bool(lst)

# Example usage:
my_list = [1, 2, 3]

if is_empty(my_list):
   print("The list is empty.")
else:
   print("The list is not empty.")
```

Q:9 Write a Python function that takes two lists and returns true if they have at least one common member.
Ans:

```
def have_common_member(list1, list2):
   # Using set intersection to find common elements
   common_elements = set(list1) & set(list2)

   # Check if there is at least one common element
   return bool(common_elements)

# Example usage:
list_a = [1, 2, 3, 4]
list_b = [4, 5, 6, 7]

if have_common_member(list_a, list_b):
   print("The lists have at least one common member.")
```

```
else:
    print("The lists do not have any common members.")
```

Q:10 Write a Python program to generate and print a list of first and last 5 elements where the values are square of numbers between 1 and 30.

Ans:

```
# Generate a list of squares of numbers between 1 and 30
squares_list = [i ** 2 for i in range(1, 31)]

# Print the entire list
print("List of squares of numbers between 1 and 30:")
print(squares_list)

# Print the first 5 elements
print("\nFirst 5 elements:")
print(squares_list[:5])

# Print the last 5 elements
print("\nLast 5 elements:")
print(squares_list[-5:])
```

Q:11 Write a Python function that takes a list and returns a new list with unique elements of the first list.

Ans:

```
def get_unique_elements(input_list):
    return list(set(input_list))

# Example usage:
original_list = [1, 2, 2, 3, 4, 4, 5]
```

```
unique_list = get_unique_elements(original_list)

print("Original List:", original_list)
print("List with Unique Elements:", unique_list)
```

Q:12 Write a Python program to convert a list of characters into a string.
Ans:

```
# Function to convert a list of characters to a string
def list_to_string(char_list):
    return ''.join(char_list)

# Example usage:
char_list = ['a', 'b', 'c', 'd', 'e']

# Convert the list to a string
result_string = list_to_string(char_list)

# Print the result
print("List of Characters:", char_list)
print("Resulting String:", result_string)
```

Q:13 Write a Python program to select an item randomly from a list.
Ans:

```
import random

# Function to select a random item from a list
def select_random_item(my_list):
    return random.choice(my_list)

# Example usage:
```

```
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Select a random item from the list
random_item = select_random_item(my_list)

# Print the result
print("Original List:", my_list)
print("Randomly Selected Item:", random_item)
```

Q:14 Write a Python program to find the second smallest number in a list.

Ans:

```
def second_smallest(lst):
    if len(lst) < 2:
        return "List must have at least two elements"

    # Find the smallest and second smallest numbers
    smallest = second_smallest = float('inf')

    for num in lst:
        if num < smallest:
            second_smallest = smallest
            smallest = num
        elif num < second_smallest and num != smallest:
            second_smallest = num

    return second_smallest

# Example usage:
my_list = [5, 2, 8, 1, 3, 7]

result = second_smallest(my_list)
```

```
print("Original List:", my_list)
print("Second Smallest Number:", result)
```

Q:15 Write a Python program to get unique values from a list.

Ans:

```
def get_unique_values(input_list):
    unique_values = list(set(input_list))
    return unique_values

# Example usage:
original_list = [1, 2, 2, 3, 4, 4, 5]

unique_values = get_unique_values(original_list)

print("Original List:", original_list)
print("Unique Values:", unique_values)
```

Q:16 Write a Python program to check whether a list contains a sub list.

Ans:

```
def contains_sublist(main_list, sub_list):
    # Check if sub_list is present in main_list
    for i in range(len(main_list) - len(sub_list) + 1):
        if main_list[i:i+len(sub_list)] == sub_list:
            return True
    return False

# Example usage:
main_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sub_list = [3, 4, 5]
```

```
if contains_sublist(main_list, sub_list):
    print("Main list contains the sub-list.")
else:
    print("Main list does not contain the sub-list.")
```

Q:17 Write a Python program to split a list into different variables.
Ans:

```
# Function to split a list into variables
def split_list(input_list):
    # Unpack the list into individual variables
    var1, var2, var3, *rest = input_list

    # Return the variables
    return var1, var2, var3, rest

# Example usage:
my_list = [1, 2, 3, 4, 5]

# Split the list into variables
variable1, variable2, variable3, rest_of_list = split_list(my_list)

# Print the result
print("Original List:", my_list)
print("Variable 1:", variable1)
print("Variable 2:", variable2)
print("Variable 3:", variable3)
print("Rest of the List:", rest_of_list)
```

Q:18 What is tuple? Difference between list and tuple.
Ans: A tuple is a collection of data type that is similar to a list.
There are key differences between lists and tuples:

1. Mutability
2. Syntax
3. Performance
4. Use Cases

Q:19 Write a Python program to create a tuple with different data types.
Ans:

```
# Creating a tuple with different data types
mixed_tuple = (1, "hello", 3.14, True, [5, 6, 7])

# Printing the tuple
print("Mixed Tuple:", mixed_tuple)

# Accessing elements of the tuple
print("Element 1:", mixed_tuple[0])
print("Element 2:", mixed_tuple[1])
print("Element 3:", mixed_tuple[2])
print("Element 4:", mixed_tuple[3])
print("Element 5:", mixed_tuple[4])
```

Q: 20 Write a Python program to create a tuple with numbers.
Ans:

```
# Creating a tuple with numbers
number_tuple = (1, 2, 3, 4, 5)

# Printing the tuple
print("Number Tuple:", number_tuple)

# Accessing elements of the tuple
print("Element 1:", number_tuple[0])
print("Element 2:", number_tuple[1])
print("Element 3:", number_tuple[2])
print("Element 4:", number_tuple[3])
print("Element 5:", number_tuple[4])
```

Q:21 Write a Python program to convert a tuple to a string.
Ans:

```
# Creating a tuple
my_tuple = (1, 2, 3, 4, 5)

# Converting the tuple to a string
tuple_as_string = ''.join(map(str, my_tuple))

# Printing the result
print("Original Tuple:", my_tuple)
print("Tuple as String:", tuple_as_string)
```

Q:22 Write a Python program to check whether an element exists within a tuple.
Ans:

```
def element_exists_in_tuple(my_tuple, element):
    return element in my_tuple

# Example usage:
my_tuple = (1, 2, 3, 4, 5)

element_to_check = 3

if element_exists_in_tuple(my_tuple, element_to_check):
    print(f"The element {element_to_check} exists in the tuple.")
else:
    print(f"The element {element_to_check} does not exist in the
tuple.")
```

Q:23 Write a Python program to find the length of a tuple.
Ans:

```
# Creating a tuple
my_tuple = (1, 2, 3, 4, 5)

# Finding the length of the tuple
tuple_length = len(my_tuple)

# Printing the result
print("Tuple:", my_tuple)
print("Length of the Tuple:", tuple_length)
```

Q:24 Write a Python program to convert a list to a tuple.
Ans:

```
# Creating a list
my_list = [1, 2, 3, 4, 5]

# Converting the list to a tuple
tuple_from_list = tuple(my_list)

# Printing the result
print("Original List:", my_list)
print("Tuple from List:", tuple_from_list)
```

Q:25 Write a Python program to reverse a tuple.
Ans:

```
# Function to reverse a tuple
```

```
def reverse_tuple(my_tuple):
   return my_tuple[::-1]

# Example usage:
original_tuple = (1, 2, 3, 4, 5)

# Reverse the tuple
reversed_tuple = reverse_tuple(original_tuple)

# Printing the result
print("Original Tuple:", original_tuple)
print("Reversed Tuple:", reversed_tuple)
```

Q:26 Write a Python program to replace last value of tuples in a list.
Ans:

```
# Function to replace the last value of tuples in a list
def replace_last_value(tuples_list, new_value):
   modified_list = [(t[:-1] + (new_value,)) for t in tuples_list]
   return modified_list

# Example usage:
original_list = [(1, 2, 3), (4, 5, 6), (7, 8, 9)]
new_last_value = 99

# Replace the last value in each tuple
modified_list = replace_last_value(original_list, new_last_value)

# Printing the result
print("Original List:", original_list)
print("Modified List with New Last Value:", modified_list)
```

Q:27 Write a Python program to find the repeated items of a tuple.
Ans:
```
def find_repeated_items(my_tuple):
    item_counts = {}
    repeated_items = set()

    for item in my_tuple:
        if item in item_counts:
            item_counts[item] += 1
            repeated_items.add(item)
        else:
            item_counts[item] = 1

    return list(repeated_items)

# Example usage:
my_tuple = (1, 2, 2, 3, 4, 4, 5)

repeated_items = find_repeated_items(my_tuple)

print("Original Tuple:", my_tuple)
print("Repeated Items:", repeated_items)
```

Q:28 Write a Python program to remove an empty tuple(s) from a list of tuples.
Ans:
```
def remove_empty_tuples(tuple_list):
    non_empty_tuples = [t for t in tuple_list if t]
    return non_empty_tuples

# Example usage:
list_of_tuples = [(1, 2, 3), (), (4, 5), (), (6, 7, 8)]
```

```
# Remove empty tuples from the list
filtered_tuples = remove_empty_tuples(list_of_tuples)

# Printing the result
print("Original List of Tuples:", list_of_tuples)
print("List of Tuples without Empty Tuples:", filtered_tuples)
```

Q:29 Write a Python program to convert a list of tuples into a dictionary.

Ans:

```
def list_of_tuples_to_dict(tuple_list):
    my_dict = dict(tuple_list)
    return my_dict

# Example usage:
list_of_tuples = [(1, 'one'), (2, 'two'), (3, 'three'), (4, 'four')]

# Convert the list of tuples to a dictionary
result_dict = list_of_tuples_to_dict(list_of_tuples)

# Printing the result
print("Original List of Tuples:", list_of_tuples)
print("Resulting Dictionary:", result_dict)
```

Q:30 Write a Python script to sort (ascending and descending) a dictionary by value.

Ans:

```
# Function to sort a dictionary by value in ascending order
def sort_dict_by_value_ascending(input_dict):
    return dict(sorted(input_dict.items(), key=lambda item: item[1]))
```

```
# Function to sort a dictionary by value in descending order
def sort_dict_by_value_descending(input_dict):
    return dict(sorted(input_dict.items(), key=lambda item: item[1],
reverse=True))

# Example usage:
my_dict = {'one': 1, 'three': 3, 'five': 5, 'two': 2, 'four': 4}

# Sort the dictionary by value in ascending order
sorted_dict_ascending = sort_dict_by_value_ascending(my_dict)

# Sort the dictionary by value in descending order
sorted_dict_descending = sort_dict_by_value_descending(my_dict)

# Printing the results
print("Original Dictionary:", my_dict)
print("Sorted Dictionary (Ascending):", sorted_dict_ascending)
print("Sorted Dictionary (Descending):", sorted_dict_descending)
```

Q:31 Write a Python script to check if a given key already exists in a dictionary.

Ans:

```
# Function to check if a key exists in a dictionary
def key_exists(my_dict, key):
    return key in my_dict

# Example usage:
my_dict = {'one': 1, 'two': 2, 'three': 3, 'four': 4}

key_to_check = 'two'

# Check if the key exists in the dictionary
if key_exists(my_dict, key_to_check):
    print(f"The key '{key_to_check}' exists in the dictionary.")
else:
    print(f"The key '{key_to_check}' does not exist in the dictionary.")
```

Q:32 Write a Python script to print a dictionary where the keys are numbers
between 1 and 15.
Ans:

```
# Function to create a dictionary with keys from 1 to 15
def create_dictionary():
    my_dict = {i: f"value_{i}" for i in range(1, 16)}
    return my_dict

# Example usage:
result_dict = create_dictionary()

# Printing the result
print("Dictionary with keys from 1 to 15:")
print(result_dict)
```

Q:33 Write a Python program to check multiple keys exists in a dictionary
Ans:

```
# Function to check if multiple keys exist in a dictionary
def check_multiple_keys_exist(my_dict, keys_to_check):
    return all(key in my_dict for key in keys_to_check)

# Example usage:
my_dict = {'one': 1, 'two': 2, 'three': 3, 'four': 4}

keys_to_check = ['two', 'four', 'five']
```

```
# Check if all keys exist in the dictionary
if check_multiple_keys_exist(my_dict, keys_to_check):
    print(f"All keys {keys_to_check} exist in the dictionary.")
else:
    missing_keys = [key for key in keys_to_check if key not in my_dict]
    print(f"The following keys are missing in the dictionary:
{missing_keys}")
```

Q:34 Why Do You Use the Zip () Method in Python?
Ans: This function is used to combine multiple iterables element-wise into tuples.
It pairs elements from each iterable together, creating an iterator that produces tuples conatining elements from the input iterables at the same index.
Common us cases for the zip() method are:

1. Iterating over multiple iterables simultaneously
2. Creating Dictionaries
3. Transposing Data

Q:35 Write a Python program to find the highest 3 values in a dictionary.
Ans:

```
# Function to find the highest 3 values in a dictionary
def highest_3_values(my_dict):
    sorted_items = sorted(my_dict.items(), key=lambda item: item[1],
reverse=True)
    highest_3 = dict(sorted_items[:3])
    return highest_3
```

```
# Example usage:
my_dict = {'one': 10, 'two': 30, 'three': 20, 'four': 15, 'five': 25}

# Find the highest 3 values in the dictionary
result = highest_3_values(my_dict)

# Printing the result
print("Original Dictionary:", my_dict)
print("Highest 3 Values:", result)
```

Q:36 Write a Python function to calculate the factorial of a number (a nonnegative integer).

Ans:

```
def factorial(n):
    if n < 0:
        return "Factorial is not defined for negative numbers"
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result

# Example usage:
number = 5
result = factorial(number)

print(f"The factorial of {number} is: {result}")
```

Q:37 Write a Python function to check whether a number is in a given range

Ans:

```
def is_in_range(number, lower_limit, upper_limit):
    return lower_limit <= number <= upper_limit

# Example usage:
num_to_check = 7
lower_limit = 1
upper_limit = 10

if is_in_range(num_to_check, lower_limit, upper_limit):
    print(f"{num_to_check} is in the range between {lower_limit} and
{upper_limit}.")
else:
    print(f"{num_to_check} is outside the range between {lower_limit}
and {upper_limit}.")
```

Q:38 Write a Python function that checks whether a passed string is palindrome or not.

Ans:

```
def is_palindrome(s):
    # Convert the string to lowercase and remove spaces
    s = s.lower().replace(" ", "")

    # Check if the string is equal to its reverse
    return s == s[::-1]

# Example usage:
string_to_check = "A man a plan a canal Panama"

if is_palindrome(string_to_check):
    print(f"'{string_to_check}' is a palindrome.")
else:
    print(f"'{string_to_check}' is not a palindrome.")
```

Q:39 How Many Basic Types Of Functions Are Available In Python?
Ans: There are primarily three types of functions:

1. Built-in Functions
2. User-Defined Functions
3. Anonymous Functions

Q:40 How can you pick a random item from a list or tuple?
Ans: You can use the “random.choice()” function from the “random” module to pick a random item from a list or tuple.

Q:41 How can you pick a random item from a range?
Ans: To pick a random item from a range, you can use the “random.choice()” function along with the “range()” function.

Q:42 How will you set the starting value in generating random numbers?
Ans: You can set the starting value for generating random numbers using the “random.seed()” function from the “random” module. The “seed()” function initializes the random number generator with a given seed value, ensuring that subsequent calls to random functions produce the same sequence of random numbers.

Q:43 Write a Python program to read a random line from a file.

Ans:
```
import random

def read_random_line(file_path):
    with open(file_path, 'r') as file:
        lines = file.readlines()
        if not lines:
            return "File is empty"

        random_line_number = random.randint(0, len(lines) - 1)
        random_line = lines[random_line_number].strip()

        return f"Random Line: {random_line}"

# Example usage:
file_path = 'example.txt'  # Replace with the path to your file

result = read_random_line(file_path)
print(result)
```

Q:44 Write a Python program to convert degree to radian.
Ans:
```
import math

def degrees_to_radians(degrees):
    radians = degrees * (math.pi / 180)
    return radians

# Example usage:
degrees_value = 45
radians_result = degrees_to_radians(degrees_value)
```

```
print(f"{degrees_value} degrees is equal to {radians_result:.4f}
radians.")
```

Q:45 Write a Python program to calculate the area of a parallelogram

Ans:

```
def parallelogram_area(base, height):
    area = base * height
    return area

# Example usage:
base_length = 5
height_length = 8

area_result = parallelogram_area(base_length, height_length)

print(f"The area of the parallelogram with base {base_length} and
height {height_length} is: {area_result}")
```

Q:46 Write a Python program to calculate surface volume and area of a cylinder.

Ans:

```
import math

def cylinder_surface_and_volume(radius, height):
    # Calculate surface area
    surface_area = 2 * math.pi * radius**2 + 2 * math.pi * radius *
height

    # Calculate volume
    volume = math.pi * radius**2 * height
    return surface_area, volume

# Example usage:
cylinder_radius = 3
cylinder_height = 8

surface_area_result, volume_result =
cylinder_surface_and_volume(cylinder_radius, cylinder_height)

print(f"The surface area of the cylinder is: {surface_area_result:.2f}")
print(f"The volume of the cylinder is: {volume_result:.2f}")
```

Q:47 Write a Python program to find the maximum and minimum numbers from the specified decimal numbers.

Ans:

```
# Example decimal numbers
decimal_numbers = [12.34, 5.67, 89.01, 3.45, 67.89]

# Find the maximum and minimum numbers
max_number = max(decimal_numbers)
min_number = min(decimal_numbers)

# Print the results
print("Decimal Numbers:", decimal_numbers)
print("Maximum Number:", max_number)
print("Minimum Number:", min_number)
```