UNIT-2

1. Write a program to define a function with multiple return values.

Theory:

In Python, a function can return more than one value at the same time. This is useful when you want to get multiple results from a single function call.

How It Works:

- Define the function using def.
- Inside the function, calculate the values you want to return.
- Use a single return statement with comma-separated values.
- When the function is called, Python returns those values as a tuple. (A tuple in Python is a type of data structure that lets you store a group of items in a single variable.)

Example of tuple:

```
#How to Create a Tuple

my_tuple = (10, "apple", 3.14)


#Accessing Tuple Elements

print(my_tuple[0])  # Output: 10

print(my_tuple[1])  # Output: apple
```

CODE:

```python
def calculate_sum_and_product(a, b):
    sum_result = a + b
    product_result = a * b
    return sum_result, product_result  # Returns two values


# Call the function and unpack the results
x = 10
y = 5
sum_val, product_val = calculate_sum_and_product(x, y)


print("Sum:", sum_val)
print("Product:", product_val)
```

OUTPUT:

```
Sum: 15
Product: 50
```

2. Write a program to define a function using default arguments.

Theory:

In Python, **default arguments** allow you to assign a default value to a function parameter. This means if the caller doesn't provide a value for that parameter, Python will use the default.

Why Use Default Arguments?

- Makes functions **more flexible.**
- Reduces the need for **overloading.**
- Allows you to call the function with **fewer arguments.**

Syntax:

```
def function_name( arg1, arg2  = default_value):
  # function body
```

Explanation:

- arg1: **Required argument** → You must provide this when calling the function.
- arg2=default_value: **Optional argument** → If you don't provide it, Python uses the default value.

CODE:

```python
def calculate_area(length, width=5):
  area = length * width
  print("Length:", length)
  print("Width:", width)
  print("Area:", area)

# Call with both length and width
calculate_area(10, 3)

print()  # Just for spacing

# Call with only length (uses default width)
calculate_area(10)
```

OUTPUT:

```
Length: 10
Width: 3
Area: 30

Length: 10
Width: 5
Area: 50
```

3. Write a program to find the length of the string without using any library functions.

Theory:

Normally, we use Python's built-in len() function to get the length of a string. But if we want to do it **manually**, we can:

- Loop through each character in the string
- Use a counter to count how many characters are present

CODE:

```
def find_length(text):
    count = 0
    for char in text:
        count += 1
    return count


# Example usage
user_input = "Hello World"
length = find_length(user_input)


print("The string is:", user_input)
print("Length of the string:", length)
```

OUTPUT:

```
The string is: Hello World
Length of the string: 11
```

4. Write a program to check if the substring is present in a given string or not.

CODE:

```python
def check_substring(main_string, sub_string):
  # Loop through the main string
  for i in range(len(main_string) - len(sub_string) + 1):
    match = True
    # Check each character of the substring
    for j in range(len(sub_string)):
      if main_string[i + j] != sub_string[j]:
        match = False
        break
    if match:
      print("Substring is present in the string.")
      return
  print("Substring is NOT present in the string.")

# Example usage
main = "Welcome to Python Lab"
sub = "Python"

check_substring(main, sub)
```

OUTPUT:

```
Substring is present in the string.
```

5. Write a program to perform the given operations on a list:

i. addition   ii. Insertion   iii. Slicing.

CODE:

```
# Initial list

my_list = [10, 20, 30]


# i. Addition – Adding an element to the end of the list

my_list.append(40)

print("After Addition:", my_list)


# ii. Insertion – Inserting an element at a specific position

my_list.insert(2, 25)  # Insert 25 at index 2

print("After Insertion:", my_list)


# iii. Slicing – Getting a part of the list

sliced_list = my_list[1:4]  # Get elements from index 1 to 3

print("Sliced List:", sliced_list)
```

OUTPUT:

```
After Addition: [10, 20, 30, 40]

After Insertion: [10, 20, 25, 30, 40]

Sliced List: [20, 25, 30]
```

6. Write a program to perform any 5 built-in functions by taking any list.

Theory:

A list is a built-in data type in Python that lets you store multiple values in a single variable.

Example of a List:

my_list = [10, 20, 30, 40]

CODE:

```
# Sample list

numbers = [5, 2, 9, 1, 7]


# 1. len() – Find the length of the list

print("Length of the list:", len(numbers))


# 2. max() – Find the maximum value

print("Maximum value:", max(numbers))


# 3. min() – Find the minimum value

print("Minimum value:", min(numbers))


# 4. sum() – Find the sum of all elements

print("Sum of all elements:", sum(numbers))


# 5. sorted() – Sort the list in ascending order

print("Sorted list:", sorted(numbers))
```

OUTPUT:

```
Length of the list: 5

Maximum value: 9

Minimum value: 1

Sum of all elements: 24

Sorted list: [1, 2, 5, 7, 9]
```