Day 4 Tutorial : Modular Programming.

Task 1: Knowledge and understanding (Function)

1. What is modular programming, and why is it essential in software development?

Modular programming is a software development approach that involves breaking down a complex program or system into smaller, self-contained modules or components. Each module is designed to perform a specific function or task, and these modules can be developed and tested independently before being integrated into the larger software system.

2. Explain the concept of a function in Python. How does it contribute to modular programming?

- A function in Python is a group of statements that together perform a task. Functions can be used to organize code into reusable units, which can make code easier to read, maintain, and debug. Functions can also be used to encapsulate complex logic, which can make code more efficient and less error-prone.
- Modular programming is a software design technique that promotes breaking down a program into smaller, self-contained modules or functions. Each module or function should have a specific purpose and should ideally perform a single task.

3. Identify 3 advantages of using functions in programs?

* Reusability: Functions can be reused in different parts of a program or even in other programs. This can save time and effort when developing software.

* Maintainability: Smaller modules are easier to understand, test, and maintain than large monolithic code.

* Modularity: Functions allow you to break down a program into smaller, more manageable pieces. This can make the code easier to read, understand, and debug.

4. Describe the key elements of a function definition in Python, including function name, parameters, and the function body.

- Function name: The function name is a unique identifier for the function. It must be a valid Python identifier, which means that it can only contain letters, numbers, and underscores.
- Parameters: Parameters are the inputs to the function. They are defined in parentheses after the function name. Parameters can be optional or required.
- Function body: The function body is the code that the function executes. It is indented after the function definition line.

5. How is information passed to a function, and how is data returned from a function?

- Information is passed to a function through its parameters. Parameters are defined in parentheses after the function name when the function is defined. When the function is called, values are passed to the parameters in the order that they are defined.

- The function would then execute the code in the function body and return a string greeting the user by name.

  Functions can also return values. This is done using the return statement. The return statement can be used to return a single value or multiple values.

6. What is the difference between built-in functions and user-defined functions in Python?

- Built-in functions are functions that are already defined in the Python language. They are available to use in any Python program without having to define them. There are many built-in functions in Python, including functions for common tasks such as printing to the console, working with strings and numbers, and manipulating files.

- User-defined functions are functions that are defined by the programmer. They can be used to encapsulate custom logic or to create reusable code snippets. User-defined functions can take any number of parameters and can return any type of value.

7. Discuss the scope of variables within functions. What is the difference between local and global variables?

- The scope of a variable in Python determines where the variable can be accessed. There are two types of scope in Python: local and global.

- Local variables are variables that are defined inside a function. They can only be accessed from within the function in which they are defined.

- Global variables are variables that are defined outside of any function. They can be accessed from anywhere in the program, including from within functions.

Task 2: Practical (Function)

1. Write a program that defines functions for basic mathematical operations (addition, subtraction, multiplication, and division). Allow the user to input two numbers and select an operation to perform. Use separate functions for each operation.

```
# 1. Write a program that defines functions for basic mathematical
operations (addition,
# subtraction, multiplication, and division). Allow the user to input two
numbers and
# select an operation to perform. Use separate functions for each
operation.

def add(a, b):
  """Adds two numbers together.

  Args:
    a: The first number.
    b: The second number.

  Returns:
    The sum of a and b.
  """

  sum = a + b
  return sum

def subtract(a, b):
  """Subtracts two numbers.

  Args:
    a: The first number.
    b: The second number.

  Returns:
    The difference of a and b.
  """

  difference = a - b
  return difference

def multiply(a, b):
  """Multiplies two numbers together.

  Args:
    a: The first number.
    b: The second number.

  Returns:
    The product of a and b.
  """

  product = a * b
  return product

def divide(a, b):
  """Divides two numbers.
```

```
  Args:
    a: The first number.
    b: The second number.

  Returns:
    The quotient of a and b.
  """

  quotient = a / b
  return quotient

def main():
  """Prompts the user to input two numbers and select an operation to
perform."""

  # Get the two numbers from the user
  num1 = float(input("Enter the first number: "))
  num2 = float(input("Enter the second number: "))

  # Get the operation from the user
  operation = input("Select an operation (+, -, *, /): ")

  # Perform the operation and display the result
  if operation == "+":
    result = add(num1, num2)
  elif operation == "-":
    result = subtract(num1, num2)
  elif operation == "*":
    result = multiply(num1, num2)
  elif operation == "/":
    result = divide(num1, num2)
  else:
    print("Invalid operation.")
    return

  print("The result is:", result)

if __name__ == "__main__":
  main()
```

2. Write a program that defines functions to find the sum, maximum and minimum values in a list of numbers.

```
def find_sum(numbers):
    """
    Find the sum of a list of numbers.

    Args:
        numbers (list): A list of numbers.

    Returns:
        float: The sum of the numbers in the list.
    """
    return sum(numbers)


def find_max(numbers):
```

```
    """
    Find the maximum value in a list of numbers.

    Args:
        numbers (list): A list of numbers.

    Returns:
        float: The maximum value in the list.
    """
    if not numbers:
        return None
    return max(numbers)


def find_min(numbers):
    """
    Find the minimum value in a list of numbers.

    Args:
        numbers (list): A list of numbers.

    Returns:
        float: The minimum value in the list.
    """
    if not numbers:
        return None
    return min(numbers)


# Example usage:
numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
total = find_sum(numbers)
maximum = find_max(numbers)
minimum = find_min(numbers)

print(f"List: {numbers}")
print(f"Sum: {total}")
print(f"Maximum: {maximum}")
print(f"Minimum: {minimum}")
```

3. Write a program that defines function to;
a. count the number of vowels in a given string,
b. length of the string
c. to reverse the string.
Test these functions.

```
# 3. Write a program that defines function to;
# a. count the number of vowels in a given string,
# b. length of the string
# c. to reverse the string.
# Test these functions.
def count_vowels(input_string):
    """
    Count the number of vowels in a given string.
```

```
    Args:
        input_string (str): The input string.

    Returns:
        int: The number of vowels in the string.
    """
    vowels = "AEIOUaeiou"
    count = 0
    for char in input_string:
        if char in vowels:
            count += 1
    return count


def string_length(input_string):
    """
    Find the length of a given string.

    Args:
        input_string (str): The input string.

    Returns:
        int: The length of the string.
    """
    return len(input_string)


def reverse_string(input_string):
    """
    Reverse a given string.

    Args:
        input_string (str): The input string.

    Returns:
        str: The reversed string.
    """
    return input_string[::-1]


# Test the functions
input_str = "Hello, World!"
vowel_count = count_vowels(input_str)
length = string_length(input_str)
reversed_str = reverse_string(input_str)

print(f"Input String: {input_str}")
print(f"Number of Vowels: {vowel_count}")
print(f"Length of the String: {length}")
print(f"Reversed String: {reversed_str}")
```

4. Create a program that defines a function for sorting a list of names in alphabetical order.
Allow the user to enter a list of names, and use the sorting function to display the names in sorted order.

```python
# 4. Create a program that defines a function for sorting a list of names
in alphabetical order.
# Allow the user to enter a list of names, and use the sorting function to
display the names
# in sorted order.
def sort_names(names):
  """Sorts a list of names in alphabetical order.

  Args:
    names: A list of names.

  Returns:
    A list of names sorted in alphabetical order.
  """

  names.sort()
  return names

def main():
  """Prompts the user to input a list of names and displays the names in
sorted order."""

  # Get the list of names from the user
  names = input("Enter a list of names, separated by commas: ").split(",")

  # Sort the names
  sorted_names = sort_names(names)

  # Display the sorted names
  print("The sorted names are:", sorted_names)

if __name__ == "__main__":
  main()
```