LOGIC BUILDING PROGRAMS

Week - 1

1. Write a Python program to check whether a given number is even or odd.

Algorithm:

- Start
- Input a number from the user.
- Check divisibility:
  - If number % 2 == 0 ⟶ the number is Even.
  - Else ⟶ the number is Odd.
- Output the result.
- End

```
# Program to check whether a number is even or odd

# Step 1: Take input from user
num = int(input("Enter a number: "))

# Step ŚŚCheck divisibility by 2
if num % 2 == 0:
    print(f"{num} is Even")
else:
    print(f"{num} is Odd")
```

```
Enter a number: 10
10 is Even
```

2. Write a Python program to check whether a number is positive, negative, or zero.

Algorithm:

- Start
- Input a number from the user.
- Check conditions:
  - If the number > 0 ⟶ it is Positive.
  - If the number < 0 ⟶ it is Negative.
  - If the number == 0 ⟶ it is Zero.
- Output the result.
- End

```
# Program to check whether a number is positive, negative, or zero

# Step 1: Take input from user
num = float(input("Enter a number: "))

# Step 2: Check conditions
if num > 0:
    print(f"{num} is Positive")
elif num < 0:
    print(f"{num} is Negative")
else:
    print("The number is Zero")
```

```
Enter a number: 12
12.0 is Positive
```

3. Write a Python program to find the largest among three numbers.

Algorithm:

- Start
- Input three numbers from the user.
- Compare the numbers:
  - If the first number is greater than or equal to the other two ⟶ it is the largest.
  - Else if the second number is greater than or equal to the other two ⟶ it is the largest.
  - Else ⟶ the third number is the largest.
- Output the largest number.
- End

```python
# Program to find the largest among three numbers

# Step 1: Take input from user
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
num3 = float(input("Enter third number: "))

# Step 2: Compare numbers
if (num1 >= num2) and (num1 >= num3):
    largest = num1
elif (num2 >= num1) and (num2 >= num3):
    largest = num2
else:
    largest = num3

# Step 3: Display result
print(f"The largest number is {largest}")
```

```
Enter first number: 12
Enter second number: 45
Enter third number: 33
The largest number is 45.0
```

4. Write a Python program to check whether a given number is a prime number.

Algorithm:

- Start
- Input a number from the user.
- Check special cases:
    - If the number ≤ 1 ⟶ it is not prime.
- Check divisibility:
    - Loop from 2 to √number.
    - If the number is divisible by any of these ⟶ it is not prime.
    - Otherwise ⟶ it is prime.
- Output the result.
- End

```python
# Program to check whether a number is prime

# Step 1: Take input from user
num = int(input("Enter a number: "))

# Step 2: Handle special cases
if num <= 1:
    print(f"{num} is not a Prime number")
else:
    # Step 3: Check divisibility
    is_prime = True
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            is_prime = False
            break

    # Step 4: Display result
    if is_prime:
        print(f"{num} is a Prime number")
    else:
        print(f"{num} is not a Prime number")
```

```
Enter a number: 7
7 is a Prime number
```

Week - 2

5. Write a Python program to find the factorial of a number.

Algorithm:

- Start
- Input a number from the user.
- Check special cases:
  - If the number < 0 ⟶ factorial does not exist.
  - If the number = 0 or 1 ⟶ factorial = 1.
- Otherwise:
  - Multiply all integers from 1 up to the number.
- Output the factorial.
- End

```python
# Program to find factorial of a number

# Step 1: Take input from user
num = int(input("Enter a number: "))

# Step 2: Initialize factorial
factorial = 1

# Step 3: Compute factorial
if num < 0:
    print("Factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1, num + 1):
        factorial *= i
    print(f"The factorial of {num} is {factorial}")
```

```
Enter a number: 5
The factorial of 5 is 120
```

```python
# Recursive function to find factorial
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Input from user
num = int(input("Enter a number: "))

if num < 0:
    print("Factorial does not exist for negative numbers")
else:
    print(f"The factorial of {num} is {factorial(num)}")
```

```
Enter a number: 0
The factorial of 0 is 1
```

6. Write a Python program to check whether a number is a palindrome.

Algorithm:

- Start
- Input a number from the user.
- Reverse the digits of the number.
    - Method: repeatedly extract last digit using % 10, build reversed number.
- Compare the reversed number with the original.
    - If equal → it is a Palindrome.
    - Else → it is Not a Palindrome.
- Output the result.
- End

```python
# Program to check palindrome using string conversion

num = input("Enter a number: ")

if num == num[::-1]:
    print(f"{num} is a Palindrome")
else:
    print(f"{num} is Not a Palindrome")
```

```
Enter a number: 123
123 is Not a Palindrome
```

7. Write a Python program to check whether a given string is a palindrome.

Algorithm:

- Start
- Input a string from the user.
- Normalize the string (optional):
  - Convert to lowercase.
  - Remove spaces if you want to check phrases.
- Reverse the string.
- Compare the original string with the reversed string:
  - If equal → it is a Palindrome.
  - Else → it is Not a Palindrome.
- Output the result.
- End

```
# Program to check whether a given string is a palindrome

# Step 1: Take input from user
text = input("Enter a string: ")

# Step 2: Reverse the string
reverse_text = text[::-1]

# Step 3: Compare original and reversed
if text == reverse_text:
    print(f'"{text}" is a Palindrome')
else:
    print(f'"{text}" is Not a Palindrome')
```

```
Enter a string: hello
"hello" is Not a Palindrome
```

Week - 3

8. Write a Python program to print the Fibonacci series up to N terms.

Algorithm:

- Start
- Input the number of terms N.
- Initialize the first two terms: 0 and 1.
- Print the first two terms.
- Repeat until N terms are printed:
  - Compute the next term as the sum of the previous two.
  - Print the term.
  - Update the previous two terms.
- End

```
# Program to print Fibonacci series up to N terms

# Step 1: Take input from user
n_terms = int(input("Enter the number of terms: "))

# Step 2: Handle special cases
if n_terms <= 0:
    print("Please enter a positive integer")
elif n_terms == 1:
    print("Fibonacci series up to 1 term:")
    print(0)
else:
    print(f"Fibonacci series up to {n_terms} terms:")
    a, b = 0, 1
    print(a, b, end=" ")
    for _ in range(2, n_terms):
        c = a + b
        print(c, end=" ")
        a, b = b, c
```

```
Enter the number of terms: 5
Fibonacci series up to 5 terms:
0 1 1 2 3
```

9. Write a Python program to find the sum of digits of a number.

Algorithm:

- Start
- Input a number from the user.
- Initialize a variable sum = 0.
- Repeat until the number becomes 0:
    - Extract the last digit using digit = number % 10.
    - Add the digit to sum.
    - Remove the last digit using integer division number //= 10.
- Output the sum.
- End

```
# Program to find sum of digits using string conversion

num = input("Enter a number: ")
sum_of_digits = sum(int(digit) for digit in num if digit.isdigit())

print(f"The sum of digits of {num} is {sum_of_digits}")
```

```
Enter a number: 1234
The sum of digits of 1234 is 10
```

10. Write a Python program to count vowels and consonants in a string.

Algorithm:

- Start
- Input a string from the user.
- Initialize two counters: vowels = 0, consonants = 0.
- Define the set of vowels: a, e, i, o, u (both uppercase and lowercase).
- Loop through each character in the string:
    - If the character is a letter:
        - If it is in the vowel set ⟶ increment vowels.
        - Else ⟶ increment consonants.
- Output the counts of vowels and consonants.
- End

```python
# Program to count vowels and consonants in a string

# Step 1: Take input from user
text = input("Enter a string: ")

# Step 2: Initialize counters
vowels = 0
consonants = 0

# Step 3: Define vowels
vowel_set = "aeiouAEIOU"

# Step 4: Loop through characters
for char in text:
    if char.isalpha():  # check if character is a letter
        if char in vowel_set:
            vowels += 1
        else:
            consonants += 1

# Step 5: Display result
print(f"Number of vowels: {vowels}")
print(f"Number of consonants: {consonants}")
```

```
Enter a string: HELLO WORLD
Number of vowels: 3
Number of consonants: 7
```

Week - 4

11. Write a Python program to reverse a string without using built-in functions.

Algorithm:

- Start
- Input a string from the user.
- Initialize an empty string rev = "".
- Loop through the original string from the last character to the first:
    - Append each character to rev.
- Output the reversed string.
- End

```python
# Reverse a string using while loop

text = input("Enter a string: ")
rev = ""
i = len(text) - 1

while i >= 0:
    rev += text[i]
    i -= 1

print(f"The reversed string is: {rev}")\
```

```
Enter a string: python
The reversed string is: nohtyp
```

12. Write a Python program to count the occurrence of each character in a string.

Algorihm:

- Start
- Input a string from the user.
- Initialize an empty dictionary to store character counts.
- Loop through each character in the string:
    - If the character is already in the dictionary → increment its count.
    - Else → add it to the dictionary with count = 1.
- Output the dictionary with character occurrences.
- End

```python
from collections import Counter

text = input("Enter a string: ")
char_count = Counter(text)

print("Character occurrences:")
for char, count in char_count.items():
    print(f"'{char}' : {count}")
```

```
Enter a string: hello
Character occurrences:
'h' : 1
'e' : 1
'l' : 2
'o' : 1
```

13. Write a Python program to create a simple calculator using conditional statements.

Algorithm:

- Start
- Input two numbers from the user.
- Input an operator (+, -, *, /).
- Use conditional statements to check which operator was entered:
  - If + ⟶ perform addition.
  - If - ⟶ perform subtraction.
  - If * ⟶ perform multiplication.
  - If / ⟶ perform division (check division by zero).
  - Else ⟶ invalid operator.
- Output the result.
- End

```
# Simple calculator using conditional statements

# Step 1: Take input from user
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
operator = input("Enter operator (+, -, *, /): ")

# Step 2: Use conditional statements
if operator == '+':
    print(f"Result: {num1} + {num2} = {num1 + num2}")
elif operator == '-':
    print(f"Result: {num1} - {num2} = {num1 - num2}")
elif operator == '*':
    print(f"Result: {num1} * {num2} = {num1 * num2}")
elif operator == '/':
    if num2 != 0:
        print(f"Result: {num1} / {num2} = {num1 / num2}")
    else:
        print("Error: Division by zero is not allowed!")
else:
    print("Invalid operator! Please choose +, -, *, or /")
```

```
Enter first number: 7
Enter second number: 3
Enter operator (+, -, *, /): *
Result: 7.0 * 3.0 = 21.0
```

Week - 5

14. Write a Python program to implement a menu-driven calculator using a loop (repeat until the user exits).

Algorithm:

- Start
- Loop forever:
  - Display a menu with options: Addition, Subtraction, Multiplication, Division, Exit.
  - Ask the user to choose an option.
  - If the choice is Exit → break the loop.
  - Otherwise → ask for two numbers.
  - Perform the selected operation using conditional statements.
- Display the result.
- End

```
# Menu-driven calculator using loop

while True:
    # Step 1: Display menu
    print("\n===== Simple Calculator =====")
    print("1. Addition")
    print("2. Subtraction")
    print("3. Multiplication")
    print("4. Division")
    print("5. Exit")

    # Step 2: Take user choice
    choice = input("Enter your choice (1-5): ")

    # Step 3: Exit condition
    if choice == '5':
        print("Exiting calculator. Goodbye!")
        break

    # Step 4: Take two numbers
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))

    # Step 5: Perform operation
    if choice == '1':
        print(f"Result: {num1} + {num2} = {num1 + num2}")
```

```
        elif choice == '2':
            print(f"Result: {num1} - {num2} = {num1 - num2}")
        elif choice == '3':
            print(f"Result: {num1} * {num2} = {num1 * num2}")
        elif choice == '4':
            if num2 != 0:
                print(f"Result: {num1} / {num2} = {num1 / num2}")
            else:
                print("Error: Division by zero is not allowed!")
        else:
            print("Invalid choice! Please select a valid option (1-5).")0
```

```
===== Simple Calculator =====
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter your choice (1-5): 1
Enter first number: 10
Enter second number: 5
Result: 10.0 + 5.0 = 15.0
```

```
===== Simple Calculator =====
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter your choice (1-5): 4
Enter first number: 12
Enter second number: 0
Error: Division by zero is not allowed!

===== Simple Calculator =====
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter your choice (1-5): 5
Exiting calculator. Goodbye!
```

15. Write a Python program to generate a multiplication table for a given number (loop until the user stops).

Algorithm:

- Start
- Loop forever:
  - Ask the user to enter a number.
  - Print its multiplication table (e.g., up to 10).
  - Ask if they want to continue.
  - If the answer is "no", break the loop.
- End

```
# Program to generate multiplication table for a given number
# Loop until the user decides to stop

while True:
    # Step 1: Take input from user
    num = int(input("Enter a number to generate its multiplication table: "))

    # Step 2: Print multiplication table up to 10
    print(f"\nMultiplication Table for {num}:")
    for i in range(1, 11):
        print(f"{num} x {i} = {num * i}")

    # Step 3: Ask user if they want to continue
    choice = input("\nDo you want to generate another table? (yes/no): ").lower()
    if choice == "no":
        print("Exiting program. Goodbye!")
        break
```

```
Enter a number to generate its multiplication table: 6

Multiplication Table for 6:
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60

Do you want to generate another table? (yes/no): yes
Enter a number to generate its multiplication table: 9
```

```
Multiplication Table for 9:
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90

Do you want to generate another table? (yes/no): no
Exiting program. Goodbye!
```

16. Write a Python program to print different patterns using loop concepts (e.g., star patterns, number patterns).

Algorithm:

- Start
- Input the number of rows.
- Use nested loops:
    - Outer loop → controls the number of rows.
    - Inner loop → controls the number of columns (characters per row).
- Print stars (*) or numbers depending on the pattern.
- End

```python
# Program to print different patterns using loops

rows = int(input("Enter number of rows: "))

print("\n--- Star Pattern (Right Triangle) ---")
for i in range(1, rows + 1):
    for j in range(i):
        print("*", end=" ")
    print()

print("\n--- Star Pattern (Pyramid) ---")
for i in range(1, rows + 1):
    # spaces
    for j in range(rows - i):
        print(" ", end=" ")
    # stars
    for j in range(2 * i - 1):
        print("*", end=" ")
    print()

print("\n--- Number Pattern (Increasing) ---")
for i in range(1, rows + 1):
    for j in range(1, i + 1):
        print(j, end=" ")
    print()
```

```python
print("\n--- Number Pattern (Repeated Row Number) ---")
for i in range(1, rows + 1):
    for j in range(i):
        print(i, end=" ")
    print()
```

```
Enter number of rows: 5

--- Star Pattern (Right Triangle) ---
*
* *
* * *
* * * *
* * * * *

--- Star Pattern (Pyramid) ---
        *
      * * *
    * * * * *
  * * * * * * *
* * * * * * * * *
```

```
--- Number Pattern (Increasing) ---
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

--- Number Pattern (Repeated Row Number) ---
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

FUNCTION-BASED QUESTIONS

Week - 6

17. Write a Python function that takes a user's name and prints a greeting message.

Algorithm:

- Define a function that takes one parameter: name.
- Inside the function, print a greeting message using that name.
- Call the function with user input.

```python
# Function to greet the user
def greet_user(name):
    print(f"Hello, {name}! Welcome to Python programming.")

# Take input from user
user_name = input("Enter your name: ")

# Call the function
greet_user(user_name)
```

```
Enter your name: sunny
Hello, sunny! Welcome to Python programming.
```

18. Write a Python function that accepts two numbers and returns their sum.

Algorithm:

- Define a function with two parameters.
- Inside the function, add the two numbers.
- Return the result.
- Call the function with user input.

```
def add_numbers(a, b):
    """
    Accepts two numbers and returns their sum.
    """
    return a + b

# Example usage:
result = add_numbers(10, 5)
print("The sum is:", result)
```

```
The sum is: 15
```

Week – 7

19. Write a Python recursive function to find the factorial of a number.

Algorithm: Recursive Factorial

- Start
- Input: Read a number n.
- Check base case:
    - If n == 0 or n == 1, return 1.
- Recursive case:
    - Otherwise, return n * factorial(n - 1).
- Output: Display the result.
- End

```python
def factorial(n):
    """
    Recursive function to find the factorial of a number.

    Parameters:
    n (int): The number to find the factorial of

    Returns:
    int: Factorial of n
    """
    if n == 0 or n == 1:   # Base case
        return 1
    else:                  # Recursive case
        return n * factorial(n - 1)

# Example usage:
num = int(input("Enter a number: "))
print("Factorial of", num, "is:", factorial(num))
```

```
Enter a number: 5
Factorial of 5 is: 120
```

20. Write a Python lambda function to check whether a number is even.

Algorithm:

- Start
- Input: Read a number n.
- Define lambda function:
    - is_even = lambda x: x % 2 == 0
    - This returns True if x is divisible by 2, otherwise False.
- Apply function: Call is_even(n).
- Decision:
    - If result is True, print "n is even".
    - Else, print "n is odd".
- End

```python
# Lambda function to check even number
is_even = lambda x: x % 2 == 0

# Example usage:
num = int(input("Enter a number: "))
if is_even(num):
    print(num, "is even.")
else:
    print(num, "is odd.")
```

```
Enter a number: 20
20 is even.
```

21. Write a Python program to calculate factorial using recursion with input validation.

Algorithm: Recursive Factorial with Input Validation

- Start
- Input: Prompt the user to enter a number n.
- Validation:
  - If the input is not an integer → show error message and ask again.
  - If the input is negative → show error message and ask again.
  - Repeat until a valid non-negative integer is entered.
- Define recursive function factorial(n):
  - Base case: If n == 0 or n == 1, return 1.
  - Recursive case: Otherwise, return n * factorial(n - 1).
- Call function: Compute factorial(n) with the validated input.
- Output: Display the result.
- End

```python
def factorial(n):
    """
    Recursive function to calculate factorial of a number.
    """
    if n == 0 or n == 1:   # Base case
        return 1
    else:                  # Recursive case
        return n * factorial(n - 1)

# Input validation
while True:
    try:
        num = int(input("Enter a non-negative integer: "))
        if num < 0:
            print("❌ Invalid input! Factorial is not defined for negative numbers. Try again.")
        else:
            break
    except ValueError:
        print("❌ Invalid input! Please enter an integer value.")

# Output
print(f"Factorial of {num} is: {factorial(num)}")
```

```
Enter a non-negative integer: -5
❌ Invalid input! Factorial is not defined for negative numbers. Try again.
Enter a non-negative integer: abc
❌ Invalid input! Please enter an integer value.
Enter a non-negative integer: 5
Factorial of 5 is: 120
```

PROJECT / ADVANCED QUESTIONS

Week - 8

22. Write a Python program to create a Library Book Management System using functions.

```python
# Library Book Management System using functions

library = []  # List to store books

# Function to add a book
def add_book(title):
    library.append(title)
    print(f'"{title}" has been added to the library.')

# Function to display all books
def display_books():
    if not library:
        print("The library is empty.")
    else:
        print("Books in the library:")
        for idx, book in enumerate(library, start=1):
            print(f"{idx}. {book}")
```

```
    # Function to search for a book
    def search_book(title):
        if title in library:
            print(f'"{title}" is available in the library.')
        else:
            print(f'"{title}" is not found in the library.')

    # Function to remove a book
    def remove_book(title):
        if title in library:
            library.remove(title)
            print(f'"{title}" has been removed from the library.')
        else:
            print(f'"{title}" is not found in the library.')

    # Main program loop
    def library_system():
        while True:
            print("\n--- Library Menu ---")
            print("1. Add Book")
            print("2. Display Books")
            print("3. Search Book")
            print("4. Remove Book")
            print("5. Exit")
```

```
            choice = input("Enter your choice (1-5): ")

            if choice == "1":
                title = input("Enter book title to add: ")
                add_book(title)
            elif choice == "2":
                display_books()
            elif choice == "3":
                title = input("Enter book title to search: ")
                search_book(title)
            elif choice == "4":
                title = input("Enter book title to remove: ")
                remove_book(title)
            elif choice == "5":
                print("Exiting Library System. Goodbye!")
                break
            else:
                print("Invalid choice. Please try again.")

    # Run the system
    library_system()
```

```
--- Library Menu ---
1. Add Book
2. Display Books
3. Search Book
4. Remove Book
5. Exit
Enter your choice (1-5): 1
Enter book title to add: Python Basics
"Python Basics" has been added to the library.

--- Library Menu ---
1. Add Book
2. Display Books
3. Search Book
4. Remove Book
5. Exit
Enter your choice (1-5): 2
Books in the library:
1. Python Basics

--- Library Menu ---
1. Add Book
2. Display Books
3. Search Book
4. Remove Book
5. Exit
```

```
Enter your choice (1-5): 3
Enter book title to search: Python Basics
"Python Basics" is available in the library.

--- Library Menu ---
1. Add Book
2. Display Books
3. Search Book
4. Remove Book
5. Exit
Enter your choice (1-5): 4
Enter book title to remove: Python Basics
"Python Basics" has been removed from the library.

--- Library Menu ---
1. Add Book
2. Display Books
3. Search Book
4. Remove Book
5. Exit
Enter your choice (1-5): 5
Exiting Library System. Goodbye!
```

Week - 9

23. Write a Python project to build a Calculator using modular programming (separate module for operations).

PROJECT STRUCTURFE:

calculator_project/

|

├── operations.py # Module containing all calculator functions

├── main.py # Main program that uses the operations module

```
# main.py

import operations   # Import our custom module

def calculator():
    print("=== Modular Calculator ===")
    print("Select operation:")
    print("1. Add")
    print("2. Subtract")
    print("3. Multiply")
    print("4. Divide")

    choice = input("Enter choice (1/2/3/4): ")

    try:
        num1 = float(input("Enter first number: "))
        num2 = float(input("Enter second number: "))
    except ValueError:
        print("Invalid input! Please enter numbers only.")
        return

    if choice == '1':
        print(f"Result: {operations.add(num1, num2)}")
    elif choice == '2':
        print(f"Result: {operations.subtract(num1, num2)}")
```

```
    elif choice == '3':
        print(f"Result: {operations.multiply(num1, num2)}")
    elif choice == '4':
        print(f"Result: {operations.divide(num1, num2)}")
    else:
        print("Invalid choice!")

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

```
    === Modular Calculator ===
Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
Enter choice (1/2/3/4): 1
Enter first number: 10
Enter second number: 5
Result: 15.0
```

Week - 10

24. Write a Python program that applies modular programming principles and defines multiple reusable functions.

```
student_management/
|
├── student_utils.py # Module containing reusable functions
└── main.py # Main program that uses the module
```

```python
# main.py

import math_utils
import string_utils

def main():
    print("=== Modular Program Demo ===")
    print("Choose an option:")
    print("1. Math Operations")
    print("2. String Operations")

    choice = input("Enter choice (1/2): ")

    if choice == '1':
        print("\nMath Operations:")
        print("a. Add")
        print("b. Subtract")
        print("c. Multiply")
        print("d. Divide")
        print("e. Factorial")

        op = input("Enter operation (a/b/c/d/e): ")

        if op in ['a', 'b', 'c', 'd']:
            num1 = float(input("Enter first number: "))
            num2 = float(input("Enter second number: "))

            if op == 'a':
                print("Result:", math_utils.add(num1, num2))
            elif op == 'b':
                print("Result:", math_utils.subtract(num1, num2))
            elif op == 'c':
                print("Result:", math_utils.multiply(num1, num2))
            elif op == 'd':
                print("Result:", math_utils.divide(num1, num2))

        elif op == 'e':
            n = int(input("Enter a number: "))
            print("Factorial:", math_utils.factorial(n))

    elif choice == '2':
        print("\nString Operations:")
        print("a. Reverse String")
        print("b. Check Palindrome")
        print("c. Count Vowels")
        print("d. Convert to Uppercase")

        op = input("Enter operation (a/b/c/d): ")
        s = input("Enter a string: ")
```

```
            if op == 'a':
                print("Reversed:", string_utils.reverse_string(s))
            elif op == 'b':
                print("Palindrome:", string_utils.is_palindrome(s))
            elif op == 'c':
                print("Vowel Count:", string_utils.count_vowels(s))
            elif op == 'd':
                print("Uppercase:", string_utils.to_uppercase(s))

        else:
            print("Invalid choice!")

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

```
  === Modular Program Demo ===
Choose an option:
1. Math Operations
2. String Operations
Enter choice (1/2): 2

String Operations:
a. Reverse String
b. Check Palindrome
c. Count Vowels
```

```
d. Convert to Uppercase
Enter operation (a/b/c/d): b
Enter a string: radar
Palindrome: True
```

Week - 11

25. Write a Python program using modular programming principles and demonstrate: • Input validation • Testing (minimum 3 test cases) • Debugging practice with comments.

```
modular_demo/
|
├── math_utils.py # Module with reusable math functions
├── validator.py # Module for input validation
├── test_cases.py # Module for testing
├── main.py # Main program
```

```python
# main.py
# Main program demonstrating modular programming, input validation, and debugging

import math_utils
import validator
import test_cases

def calculator():
    print("=== Modular Calculator ===")
    print("Select operation:")
    print("1. Add")
    print("2. Subtract")
    print("3. Multiply")
    print("4. Divide")

    choice = input("Enter choice (1/2/3/4): ")

    if not validator.validate_choice(choice, ['1', '2', '3', '4']):
        return

    num1 = validator.validate_number(input("Enter first number: "))
    num2 = validator.validate_number(input("Enter second number: "))
```

```python
    # Debugging practice: check if inputs are valid
    # If num1 or num2 is None, it means invalid input was entered
    if num1 is None or num2 is None:
        print("Debug: Invalid numeric input detected. Exiting calculator.")
        return

    if choice == '1':
        print("Result:", math_utils.add(num1, num2))
    elif choice == '2':
        print("Result:", math_utils.subtract(num1, num2))
    elif choice == '3':
        print("Result:", math_utils.multiply(num1, num2))
    elif choice == '4':
        print("Result:", math_utils.divide(num1, num2))

if __name__ == "__main__":
    # Run calculator
    calculator()

    # Run test cases
    test_cases.run_tests()
```

```
    === Modular Calculator ===
Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
Enter choice (1/2/3/4): 4
Enter first number: 10
Enter second number: 0
Result: Error! Division by zero.

=== Running Test Cases ===
Test 1 (Add 10 + 5): Pass
Test 2 (Divide 20 / 4): Pass
Test 3 (Divide 10 / 0): Pass
```

Week - 12

26. Write a Python project for a User Registration System with input validation, testing, and debugging documentation.

```
user_registration_system/
|
├── validator.py # Input validation functions
├── registration.py # Core registration logic
└── main.py # Main program with testing and debugging
```

```python
# main.py
# Main program demonstrating user registration system

import registration
import test_cases

def main():
    print("=== User Registration System ===")
    username = input("Enter username: ")
    password = input("Enter password: ")
    email = input("Enter email: ")

    result = registration.register_user(username, password, email)

    if isinstance(result, str):
        print("Registration failed:", result)
    else:
        print("Registration successful:", result)

    # Run test cases
    test_cases.run_tests()

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

```
   === User Registration System ===
Enter username: Alice123
Enter password: pass123
Enter email: alice@example.com
Debug: Attempting to register user with username=Alice123, email=alice@example.com
Debug: User object created successfully.
Registration successful: User(username=Alice123, email=alice@example.com)

=== Running Test Cases ===
Test 1 (Valid User): Pass
Test 2 (Invalid Username): Pass
Test 3 (Invalid Password): Pass
Test 4 (Invalid Email): Pass
```

Week - 13

27. Write a mini-project in Python incorporating various programming concepts (loops, functions, lists, modules, validation, testing).

Project Structure

- cart.py → Handles cart operations (add, remove, view).
- billing.py → Handles billing and checkout.
- main.py → Entry point for the program.
- test_cart.py → Simple test cases for validation.

```python
# main.py
import cart
import billing

def menu():
    while True:
        print("\n1. Add Item\n2. Remove Item\n3. View Cart\n4. Checkout\n5. Exit")
        choice = input("Enter choice: ")

        if choice == "1":
            name = input("Enter item name: ")
            price = float(input("Enter item price: "))
            qty = int(input("Enter quantity: "))
            print(cart.add_item(name, price, qty))

        elif choice == "2":
            name = input("Enter item name to remove: ")
            print(cart.remove_item(name))

        elif choice == "3":
            print(cart.view_cart())

        elif choice == "4":
            print(billing.generate_bill())
```

```python
        elif choice == "5":
            print("Exiting... Thank you!")
            break
        else:
            print("Invalid choice. Try again.")

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

```
 1. Add Item
2. Remove Item
3. View Cart
4. Checkout
5. Exit
Enter choice: 1
Enter item name: Apple
Enter item price: 10
Enter quantity: 2
```