Logic Building Program

week -1

Write a python program to check whether given number is even or odd

algorithm

Start

Read an integer n

If n % 2 == 0

Print "Even"

Else

Print "Odd"

Stop

In [10]:
```
num = int(input("Enter a number: "))

if num % 2 == 0:
    print("The number is Even")
else:
    print("The number is Odd")
```

```
The number is Odd
```

write a python program to check whether given number is positive,negative or zero

algorithm

In [ ]:
```
Start
```

```
Read a number n

If n > 0

Print "Positive"

Else if n < 0

Print "Negative"

Else

Print "Zero"
stop
```

In [2]:
```
num = int(input("Enter a number: "))

if num > 0:
    print("The number is Positive")
elif num < 0:
    print("The number is Negative")
else:
    print("The number is Zero")
```

```
The number is Positive
```

write a python program to find largest among three numbers

algorithm

In [ ]:
```
Start

Read three numbers a, b, and c

If a ≥ b and a ≥ c

Print "a is the largest"

Else if b ≥ a and b ≥ c

Print "b is the largest"

Else

Print "c is the largest"

Stop
```

In [1]:
```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
c = int(input("Enter third number: "))

if a >= b and a >= c:
    print("Largest number is:", a)
elif b >= a and b >= c:
    print("Largest number is:", b)
else:
    print("Largest number is:", c)
```

```
Largest number is: 7
```

write a python program to check whether given number is prime number

algorithm

In [ ]:
```
Start

Read a number n

If n ≤ 1

Print "Not a Prime Number" and Stop

For i from 2 to n-1

If n % i == 0

Print "Not a Prime Number" and Stop

If no divisor is found

Print "Prime Number"

Stop
```

In [2]:
```
num = int(input("Enter a number: "))

if num <= 1:
    print("Not a prime number")
else:
    for i in range(2, num):
        if num % i == 0:
            print("Not a prime number")
            break
    else:
        print("Prime number")
```

Prime number

week -2

write a python program to find factorial of number

Algorithm

In [ ]:
```
Start

Read a number n

If n < 0

Print "Factorial not defined" and Stop

Set fact = 1

For i from 1 to n

fact = fact × i

Print fact

Stop
```

In [11]:
```
num = int(input("Enter a number: "))
factorial = 1

for i in range(1, num + 1):
    factorial *= i

print("Factorial is:", factorial)
```

```
Factorial is: 5040
```

write a python program to check whether a number is pallindrome

Algorithm

In [ ]:
```
Start

Read a number n

Store n in a temporary variable temp

Set rev = 0

While n > 0

digit = n % 10

rev = rev * 10 + digit
n = n // 10

If rev == temp

Print "Palindrome Number"

Else

Print "Not a Palindrome Number"

Stop
```

In [4]:
```
num = int(input("Enter a number: "))
temp = num
rev = 0

while temp > 0:
    rev = rev * 10 + temp % 10
    temp //= 10

if num == rev:
    print("Palindrome number")
else:
    print("Not a palindrome number")
```

```
Palindrome number
```

write a python program to check whether a given string is a pallindrome

algorithm

In [ ]:
```
Start

Read a string s

Reverse the string

If the original string is equal to the reversed string

Print "Palindrome String"

Else

Print "Not a Palindrome String"
stop
```

In [5]:
```
text = input("Enter a string: ")

if text == text[::-1]:
    print("Palindrome string")
else:
    print("Not a palindrome string")
```

```
Palindrome string
```

week -3

write a python program to print the Fibonacci series up to N terms

algorithm

In [ ]:
```
Start

Read a number N (number of terms)

Initialize a = 0 and b = 1
```

```
Print a and b

For i from 3 to N

c = a + b

Print c

a = b

b = c

Stop
```

In [6]:
```
n = int(input("Enter number of terms: "))

a, b = 0, 1
print("Fibonacci series:")

for i in range(n):
    print(a, end=" ")
    a, b = b, a + b
```

```
Fibonacci series:
0 1 1 2 3 5 8
```

write a python program to find the sum of digits of a number

Algorithm

In [ ]:
```
Start

Read a number n

Initialize sum = 0

While n > 0

digit = n % 10

sum = sum + digit

n = n // 10
stop
```

In [7]:
```
num = int(input("Enter a number: "))
sum_digits = 0
```

```
while num > 0:
    sum_digits += num % 10
    num //= 10

print("Sum of digits:", sum_digits)
```

```
Sum of digits: 7
```

write a python program to count vowels and consonants in a string

algorithm

In [ ]:
```
Start

Read the input string from the user.

Initialize two counters: vowels = 0 and consonants = 0.

Convert the string to lowercase (to handle both uppercase and lowercase letter

For each character ch in the string:

If ch is an alphabet:

If ch is in "aeiou", increment vowels.

Otherwise, increment consonants.

Print the counts of vowels and consonants.

End
```

In [8]:
```
text = input("Enter a string: ")
vowels = 0
consonants = 0

for ch in text.lower():
    if ch.isalpha():
        if ch in "aeiou":
            vowels += 1
        else:
            consonants += 1

print("Vowels:", vowels)
print("Consonants:", consonants)
```

```
Vowels: 6
Consonants: 6
```

week -4

write a python program to reverse a string without using built -in functions.

algorithm

In [ ]:
```
Start

Read the input string from the user.

Initialize an empty string reversed_str = "".

Loop through the original string from the last character to the first:

Append each character to reversed_str.

Print reversed_str.

End
```

In [1]:
```
# Reverse a string without using built-in functions

string = input("Enter a string: ")
reversed_string = ""

# Loop through the string backwards
for i in range(len(string) - 1, -1, -1):
    reversed_string += string[i]

print("Reversed string:", reversed_string)
```

```
Reversed string: ahseenaM
```

write a python program to count the occurrence of each character in the string

algorithm

In [ ]:
```
Start
```

```
Read the input string s

Create an empty dictionary count

For each character ch in the string:

If ch is already in count:

Increment count[ch] by 1

Else:

Add ch to count with value 1

Display each character and its count

Stop
```

In [2]:
```python
# Count the occurrence of each character in a string

string = input("Enter a string: ")
char_count = {}

for char in string:
    if char in char_count:
        char_count[char] += 1
    else:
        char_count[char] = 1

# Display the result
for char, count in char_count.items():
    print(f"'{char}' appears {count} times")
```

```
'M' appears 1 times
'a' appears 2 times
'n' appears 1 times
'e' appears 2 times
's' appears 1 times
'h' appears 1 times
```

write a python program to create a simple calculator using conditional statements

Algorithm

In [ ]:
```
Start
```

```
Read two numbers num1 and num2

Display calculator menu:

1 → Addition

2 → Subtraction

3 → Multiplication

4 → Division

Read the user's choice

Use conditional statements:

If choice is 1, perform num1 + num2

Else if choice is 2, perform num1 - num2

Else if choice is 3, perform num1 * num2

Else if choice is 4:

If num2 ≠ 0, perform num1 / num2
Display the result

Stop
```

In [3]:
```python
# Simple calculator using conditional statements

num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
operator = input("Enter operator (+, -, *, /): ")

if operator == '+':
    print("Result:", num1 + num2)
elif operator == '-':
    print("Result:", num1 - num2)
elif operator == '*':
    print("Result:", num1 * num2)
elif operator == '/':
    if num2 != 0:
        print("Result:", num1 / num2)
    else:
        print("Error: Division by zero!")
else:
    print("Invalid operator!")
```

```
Result: 10.0
```

week -5

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

algorithm

In [ ]:

```
Start

Repeat the following steps until the user chooses to exit:

Display the menu:

1 → Addition

2 → Subtraction

3 → Multiplication

4 → Division

5 → Exit

Read the user’s choice

If choice is 5:

Display “Exiting calculator”

Go to step 3 (Stop)

Else:

Read two numbers num1 and num2

Use conditional statements:

If choice = 1, compute num1 + num2

Else if choice = 2, compute num1 - num2

Else if choice = 3, compute num1 * num2

Else if choice = 4:
```

```
If num2 ≠ 0, compute num1 / num2

Else display “Division by zero not allowed”

Else display “Invalid choice”

Display the result (if valid)

Stop
```

while True: print("\n--- Calculator Menu ---") print("1. Addition") print("2. Subtraction") print("3. Multiplication") print("4. Division") print("5. Exit")

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

    if choice == '5':
        print("Exiting calculator...")
        break

    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))

    if choice == '1':
        print("Result:", num1 + num2)
    elif choice == '2':
        print("Result:", num1 - num2)
    elif choice == '3':
        print("Result:", num1 * num2)
    elif choice == '4':
        if num2 != 0:
            print("Result:", num1 / num2)
        else:
            print("Division by zero not allowed")
    else:
        print("Invalid choice")
```

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

In [5]:

```
while True:
    num = int(input("\nEnter a number to print its table: "))

    for i in range(1, 11):
        print(num, "x", i, "=", num * i)

    choice = input("Do you want another table? (yes/no): ")
```

```
    if choice.lower() != 'yes':
        print("Program stopped.")
        break
```

```
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
Program stopped.
```

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

star pattern (Right Triangle)

In [6]:
```
rows = int(input("Enter number of rows: "))

for i in range(1, rows + 1):
    print("*" * i)
```

```
*
**
***
****
```

Inverted star pattern

In [7]:
```
rows = int(input("Enter number of rows: "))

for i in range(rows, 0, -1):
```

```
    print("*" * i)
```

```
*****
****
***
**
*
```

number pattern

In [8]:
```
rows = int(input("Enter number of rows: "))

for i in range(1, rows + 1):
    for j in range(1, i + 1):
        print(j, end=" ")
    print()
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
```

Function based questions

week -6

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

algorithm

In [ ]:
```
Start

Define a function greet(name)

Inside the function:

Print a greeting message using the given name

Read the user’s name
```

```
Call the function by passing the name as an argument

Stop
```

In [9]:
```
def greet_user(name):
    print("Hello,", name + "! Welcome.")

# Example call
user_name = input("Enter your name: ")
greet_user(user_name)
```

```
Hello, maneesha! Welcome.
```

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

algorithm

In [ ]:
```
Start

Define a function add_numbers(a, b)

Inside the function:

Calculate sum = a + b

Return sum

Read two numbers from the user

Call the function by passing the two numbers

Display the returned result

Stop
```

In [10]:
```
def add_numbers(a, b):
    return a + b

# Example call
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
result = add_numbers(num1, num2)
print("Sum:", result)
```

```
Sum: 10
```

week-7

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

Algorithm

In [ ]:
```
Start

Define a recursive function factorial(n)

Check the base condition:

If n == 0 or n == 1, return 1

Else:

Return n * factorial(n - 1)

Read a number from the user

Call the function with the given number

Display the returned factorial value

Stop
```

In [11]:
```python
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

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

```
Factorial: 5040
```

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

algorithm

In [ ]:
```
Start

Read a number n from the user

Define a lambda function:

If n % 2 == 0, return True

Else, return False

Call the lambda function with the given number

If the result is True, display “Number is Even”

Else, display “Number is Odd”

Stop
```

In [12]:
```
is_even = lambda x: x % 2 == 0

# Example call
num = int(input("Enter a number: "))

if is_even(num):
    print("The number is Even")
else:
    print("The number is Odd")
```

```
The number is Odd
```

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

algorithm

In [ ]:
```
Start
```

```
Read an integer n from the user

Validate the input:

If n < 0, display “Factorial not defined for negative numbers” and stop

Define a recursive function factorial(n)

Inside the function:

If n == 0 or n == 1, return 1 (base case)

Else, return n * factorial(n - 1) (recursive case)

Call the function with the validated input

Display the factorial result

Stop
```

def factorial(n): if n == 0 or n == 1: return 1 else: return n * factorial(n - 1)

num = int(input("Enter a number: "))

if num < 0: print("Factorial is not defined for negative numbers") else: print("Factorial of", num, "is:", factorial(num))

Project /advanced questions

week -8

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

algorithm

In [ ]:

```
Start

Create an empty list books to store book names

Define the following functions:
```

```
add_book()

Read book name

Add the book to the list

remove_book()

Read book name

If book exists, remove it

Else display “Book not found”

display_books()

If list is empty, display “No books available”

Else display all books

Repeat until the user chooses to exit:
Display menu:

1 → Add Book

2 → Remove Book

3 → Display Books

4 → Exit

Read user choice

Call the corresponding function

Stop
```

In [1]:

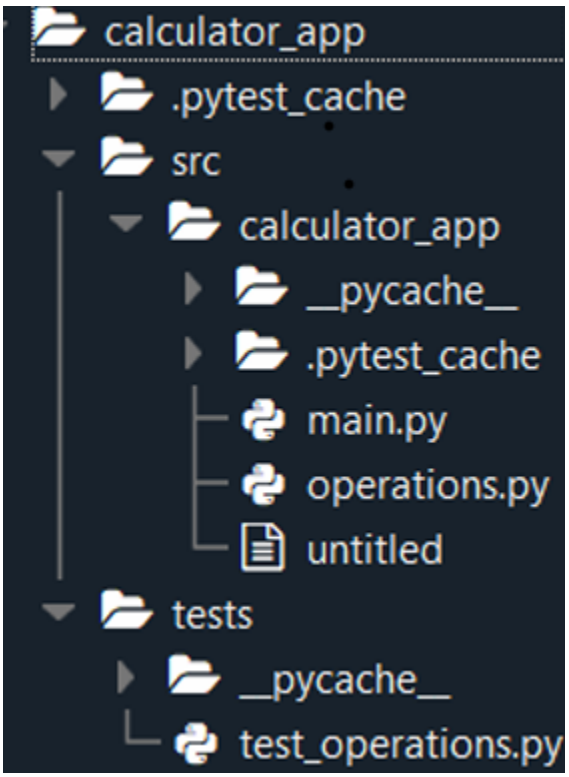
```
library = []

def add_book():
    book = input("Enter book name: ")
    library.append(book)
    print("Book added successfully.")

def remove_book():
    book = input("Enter book name to remove: ")
    if book in library:
        library.remove(book)
        print("Book removed successfully.")
    else:
        print("Book not found.")

def display_books():
```

```
    if not library:
        print("Library is empty.")
    else:
        print("Books in Library:")
        for book in library:
            print("-", book)

while True:
    print("\n--- Library Management Menu ---")
    print("1. Add Book")
    print("2. Remove Book")
    print("3. Display Books")
    print("4. Exit")

    choice = input("Enter your choice: ")

    if choice == '1':
        add_book()
    elif choice == '2':
        remove_book()
    elif choice == '3':
        display_books()
    elif choice == '4':
        print("Exiting Library System.")
        break
    else:
        print("Invalid choice")
```

```
--- Library Management Menu ---
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Exiting Library System.
```

week-9

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

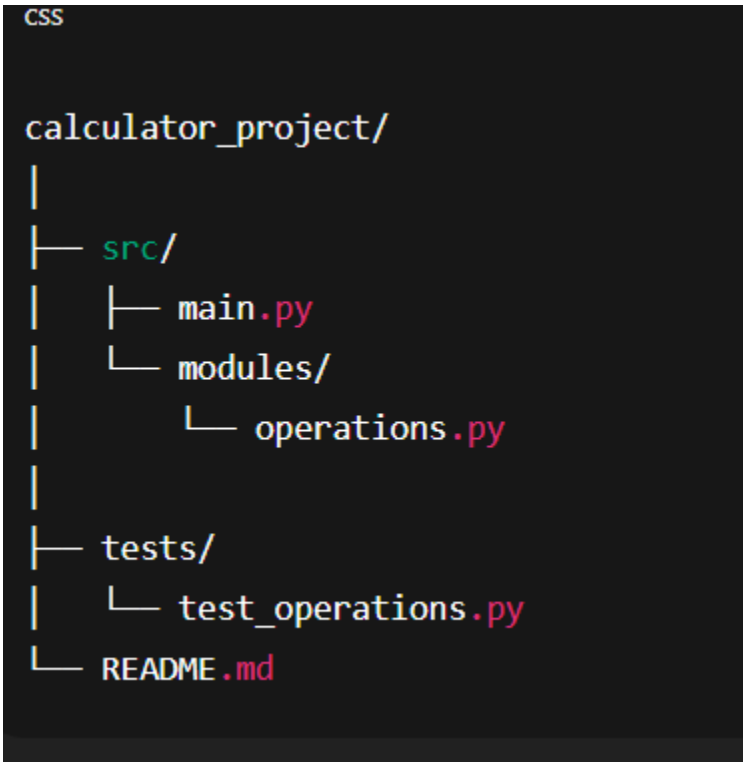


In [ ]:
```python
# src/calculator_app/operations.py
def add(a: float, b: float) -> float:
    return a + b
def subtract(a: float, b: float) -> float:
    return a - b
def multiply(a: float, b: float) -> float:
    return a * b
def divide(a: float, b: float) -> float:
    if b == 0:
        raise ValueError("Cannot divide by zero.")
    return a / b

# src/calculator_app/main.py
import os
import pytest
from src.calculator_app.operations import add, subtract, multiply, divide
def calculator():
    print("Welcome to Mini Calculator!\n")
```

```
while True:
    print("\nChoose operation:")
    print("1: Add")
    print("2: Subtract")
    print("3: Multiply")
    print("4: Divide")
    print("5: Exit")
    choice = input("Enter choice (1/2/3/4/5): ")
    if choice == "5":
        print("Exiting calculator...\n")
        break
    try:
        a = float(input("Enter first number: "))
        b = float(input("Enter second number: "))
    except ValueError:
        print("Please enter valid numbers!\n")
        continue
    if choice == "1":
        print(f"Result: {add(a, b)}\n")
    elif choice == "2":
        print(f"Result: {subtract(a, b)}\n")
    elif choice == "3":
        print(f"Result: {multiply(a, b)}\n")
    elif choice == "4":
        try:
            print(f"Result: {divide(a, b)}\n")
        except ValueError as e:
            print(f"Error: {e}\n")
        else:
            print("Invalid choice! Please try again.\n")
def run_tests():
    print("Running automated tests...")

    # Absolute path to test_operations.py
    project_root = os.path.abspath(os.path.join(os.path.dirname(__file__), "..
    test_file_path = os.path.join(project_root, "tests", "test_operations.py")

    # Run pytest programmatically
    result = pytest.main([test_file_path, "-q", "--tb=short"])
    if result == 0:
        print("All tests passed! ✅")
    else:
        print("Some tests failed! ❌")

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


# tests/test_operations.py
import pytest
from src.calculator_app.operations import add, subtract, multiply, divide
def test_add():
```

```
    assert add(2, 3) == 5
    assert add(-1, 1) == 0
    assert add(0, 0) == 0
def test_subtract():
    assert subtract(5, 3) == 2
    assert subtract(0, 5) == -5
def test_multiply():
    assert multiply(2, 4) == 8
    assert multiply(-1, 5) == -5
    assert multiply(0, 100) == 0
def test_divide():
    assert divide(10, 2) == 5
    assert divide(9, 3) == 3
def test_divide_by_zero():
    with pytest.raises(ValueError):
        divide(5, 0)
```

b



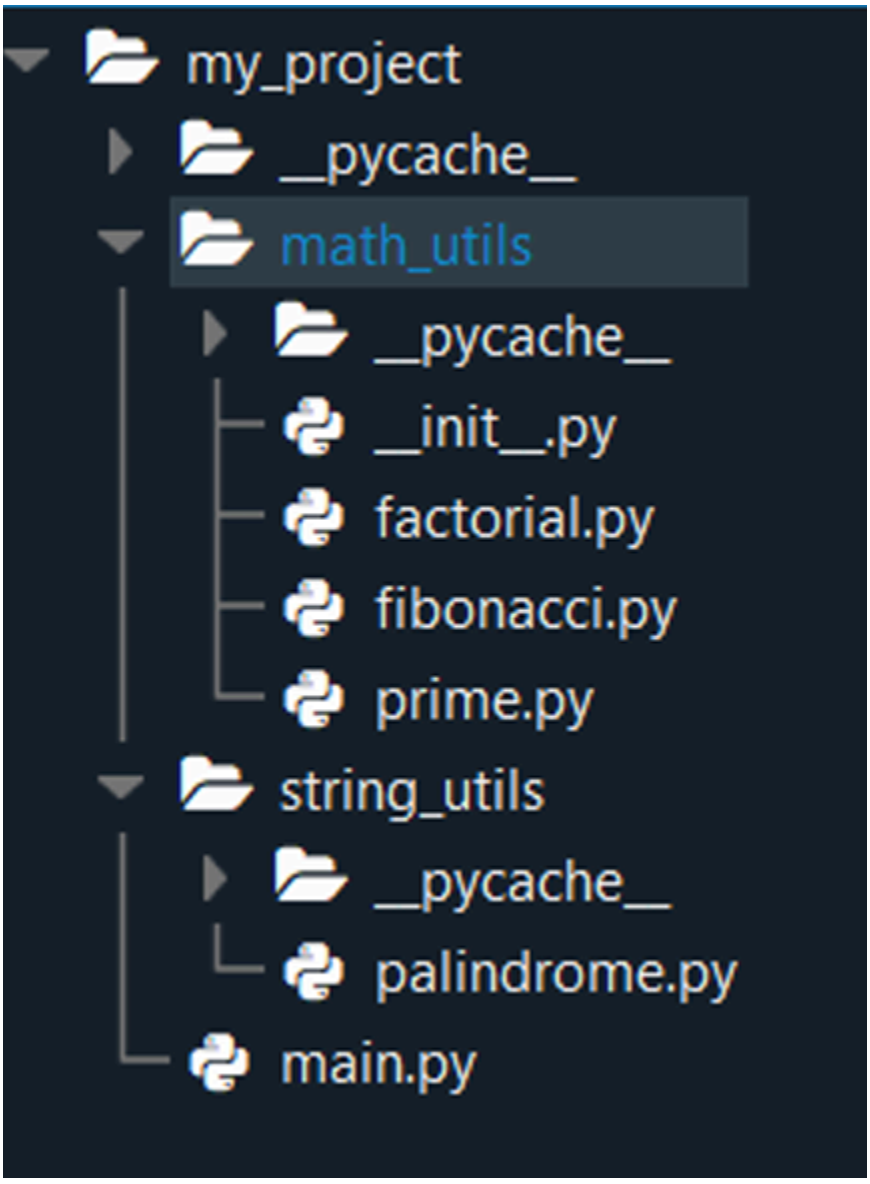
In [ ]:
```
# src/modules/operations.py

def add(a, b):
    """Return the sum of two numbers"""
    return a + b

def subtract(a, b):
    """Return the difference of two numbers"""
    return a - b
```

```
def multiply(a, b):
    """Return the product of two numbers"""
    return a * b

def divide(a, b):
    """Return the division of two numbers"""
    if b == 0:
        return "Error! Division by zero."
    return a / b

# src/main.py

from modules import operations

def calculator():
    while True:
        print("\n--- Calculator Menu ---")
        print("1. Addition")
        print("2. Subtraction")
        print("3. Multiplication")
        print("4. Division")
        print("5. Exit")

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

        if choice == '5':
            print("Exiting Calculator. Goodbye!")
            break

        if choice in ['1', '2', '3', '4']:
            try:
                num1 = float(input("Enter first number: "))
                num2 = float(input("Enter second number: "))

                if choice == '1':
                    print("Result:", operations.add(num1, num2))
                elif choice == '2':
                    print("Result:", operations.subtract(num1, num2))
                elif choice == '3':
                    print("Result:", operations.multiply(num1, num2))
                elif choice == '4':
                    print("Result:", operations.divide(num1, num2))
            except ValueError:
                print("Invalid input! Please enter numbers only.")
        else:
            print("Invalid choice! Please select a number between 1-5.")

if __name__ == "__main__":
    calculator()

# tests/test_operations.py
```

```python
from src.modules import operations

# Test addition
assert operations.add(5, 3) == 8
assert operations.add(-2, 2) == 0

# Test subtraction
assert operations.subtract(10, 4) == 6
assert operations.subtract(0, 5) == -5

# Test multiplication
assert operations.multiply(3, 4) == 12
assert operations.multiply(-2, 5) == -10

# Test division
assert operations.divide(10, 2) == 5
assert operations.divide(5, 0) == "Error! Division by zero."

print("All tests passed successfully!")
```

week-10

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



In [ ]:
```python
# -*- coding: utf-8 -*
# math_utils/factorial.py
def factorial(n):
"""Return factorial of a number using recursion."""
if n == 0 or n == 1:
```

```
return 1
return n * factorial(n - 1)

#  -*- coding: utf-8 -*
# math_utils/fibonacci.py
def generate_fibonacci(limit):
"""Generate Fibonacci sequence up to a limit."""
sequence = [0, 1]
while sequence[-1] + sequence[-2] <= limit:
sequence.append(sequence[-1] + sequence[-2]
                return sequence

# -*- coding: utf-8 -*
# math_utils/prime.py
def is_prime(n):
"""Check if a number is prime."""
if n <= 1:
return False
for i in range(2, int(n ** 0.5) + 1):
if n % i == 0:
return False
return True

# -*- coding: utf-8 -*
# string_utils/palindrome.py

def is_palindrome(s):
"""Check if a string is palindrome (case-insensitive)."""
s = s.replace(" ", "").lower()
return s == s[::-1]

# -*- coding: utf-8 -*
# main.py
#import my_project
from math_utils.prime import is_prime
from math_utils.fibonacci import generate_fibonacci
from math_utils.factorial import factorial
from string_utils.palindrome import is_palindrome
def main():
while True:
print("\n🧩Logical Toolkit")
print("1. Check Prime Number")
print("2. Generate Fibonacci Sequence")
print("3. Calculate Factorial")
print("4. Check Palindrome String")
print("5. Exit")
choice = input("Enter your choice: ")
if choice == '1':
num = int(input("Enter a number: "))
print(f"{num} is {'prime' if is_prime(num) else 'not prime'}.")
elif choice == '2':
limit = int(input("Generate Fibonacci up to: "))
print("Sequence:", generate_fibonacci(limit))
```

```
elif choice == '3':
num = int(input("Enter a number: "))
print(f"Factorial of {num} = {factorial(num)}")
elif choice == '4':
text = input("Enter a string: ")
print(f"'{text}' is {'a palindrome' if is_palindrome(text) else 'not a palindr
elif choice == '5':
print("Exiting... Goodbye!")
break
else:
print("Invalid choice. Try again.")

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

week-11

Write a Python program using modular programming principles and demonstrate:

Input validation

Testing (minimum 3 test cases)

Debugging practice with comments

In [ ]:
```
# =========================
# Module: operations.py
# =========================

def add(a, b):
    """Returns the sum of two numbers"""
    return a + b


def divide(a, b):
    """
    Returns the division of two numbers.
    Raises ValueError if b is zero.
    """
    if b == 0:
        # Debugging: division by zero detected
```

```
        raise ValueError("Division by zero is not allowed")
    return a / b


# ========================
# Module: validation.py
# ========================

def validate_number(value):
    """
    Validates and converts input to float.
    Raises ValueError if conversion fails.
    """
    try:
        return float(value)
    except ValueError:
        # Debugging: invalid input detected
        raise ValueError("Invalid number entered")


# ========================
# Main Program
# ========================

def main():
    print("Simple Modular Calculator")

    try:
        # Input validation
        num1 = validate_number(input("Enter first number: "))
        num2 = validate_number(input("Enter second number: "))

        print("Addition:", add(num1, num2))
        print("Division:", divide(num1, num2))

    except ValueError as e:
        # Debugging: catch and display errors
        print("Error:", e)


# ========================
# Testing (Minimum 3 Test Cases)
# ========================

def test_operations():
    # Test Case 1: Addition
    assert add(5, 3) == 8
    print("Test Case 1 Passed")

    # Test Case 2: Division
    assert divide(10, 2) == 5
    print("Test Case 2 Passed")
```

```
    # Test Case 3: Division by zero
    try:
        divide(10, 0)
    except ValueError:
        print("Test Case 3 Passed (Division by zero handled)")


# ========================
# Program Execution
# ========================

if __name__ == "__main__":
    test_operations()  # Run test cases
    main()             # Run main program
```

In [ ]:

```
Test Case 1 Passed
Test Case 2 Passed
Test Case 3 Passed (Division by zero handled)
Simple Modular Calculator
Enter first number: 10
Enter second number: 2
Addition: 12.0
Division: 5.0
```

week-12

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

In [ ]:

```
# ========================
# Module: validation.py
# ========================

def validate_username(username):
    """
    Username must be at least 4 characters long.
    """
    if len(username) < 4:
        # Debugging: Username length issue
        raise ValueError("Username must be at least 4 characters long")
    return username


def validate_password(password):
    """
```

```
    Password must be at least 6 characters long.
    """
    if len(password) < 6:
        # Debugging: Weak password detected
        raise ValueError("Password must be at least 6 characters long")
    return password


def validate_age(age):
    """
    Age must be a number and >= 18.
    """
    try:
        age = int(age)
    except ValueError:
        # Debugging: Non-numeric age input
        raise ValueError("Age must be a number")

    if age < 18:
        # Debugging: Underage user detected
        raise ValueError("User must be at least 18 years old")

    return age


# =========================
# Module: user.py
# =========================

registered_users = []  # Temporary in-memory storage


def register_user(username, password, age):
    """
    Registers a user after validation.
    """
    user = {
        "username": username,
        "password": password,
        "age": age
    }
    registered_users.append(user)
    return True


# =========================
# Testing Module: tests.py
# =========================

def run_tests():
    # Test Case 1: Valid user registration
    assert validate_username("mansa") == "mansa"
    assert validate_password("secret12") == "secret12"
```

```
    assert validate_age(20) == 20
    print("Test Case 1 Passed: Valid inputs")

    # Test Case 2: Invalid username
    try:
        validate_username("abc")
    except ValueError:
        print("Test Case 2 Passed: Invalid username handled")

    # Test Case 3: Underage user
    try:
        validate_age(15)
    except ValueError:
        print("Test Case 3 Passed: Underage user handled")


# =========================
# Main Program: main.py
# =========================

def main():
    print("=== User Registration System ===")

    try:
        username = validate_username(input("Enter username: "))
        password = validate_password(input("Enter password: "))
        age = validate_age(input("Enter age: "))

        register_user(username, password, age)
        print("✅ User registered successfully!")

    except ValueError as error:
        # Debugging: Display validation errors
        print("❌ Registration failed:", error)


# =========================
# Program Execution
# =========================

if __name__ == "__main__":
    run_tests()   # Run test cases
    main()        # Run main program
```

In [ ]:

```
Test Case 1 Passed: Valid inputs
Test Case 2 Passed: Invalid username handled
Test Case 3 Passed: Underage user handled
=== User Registration System ===
Enter username: mansa
Enter password: password123
Enter age: 21
user registed succeessfully !
```