LOGIC BUILDING PROGRAMS

week-1

```
In [ ]: 1.Write a Python program to check whether a given number is even or odd.
```

```
In [ ]: algorithm:
        step1: Start

        step2: Input a number (let’s call it num).

        step3: Process:

        step4: Compute the remainder when num is divided by 2 → num % 2.

              If the remainder is 0, then the number is even.

              Otherwise, the number is odd.

              Output the result (either "Even" or "Odd").

        step5: End
```

```
In [ ]: #code:
        # Program to check if a number is even or odd
        # Take input from the user num = int(input("Enter a number: "))
        # Check if the number is divisible by 2 if num % 2 == 0: print(f"{num} is Even")
        else:
        print(f"{num} is Odd")
```

Input: 4
Output:

Code 4 is Even Input: 7
Output:

Code 7 is Odd

```
In [ ]: 2. Write a Python program to check whether a number is positive, negative, or zero.
```

```
In [ ]: algorithm:
        step1: Start

        step2: Input a number from the user.

        step3: Check condition:

        step4: If the number is greater than 0 → Positive
```

```
        step5: Else if the number is less than 0 → Negative

               Else → Zero

        step6: Output the result (Positive / Negative / Zero).

        step7: Stop
```

In [ ]:

```
#code
# Program to check if a number is positive, negative, or zero
# Take input from the user
num = float(input("Enter a number: "))
# Check conditions
if num > 0: print(f"{num} is Positive")
elif num < 0:
print(f"{num} is Negative")
else:
print(f"{num} is Zero")
```

Input: 5

Output:

Code 5.0 is Positive

In [ ]:

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

In [ ]:

```
algorithm:
step1: Start

step2: Input three numbers: num1, num2, num3.

step3: Compare:

     If num1 >= num2 and num1 >= num3 → num1 is the largest.

    Else if num2 >= num1 and num2 >= num3 → num2 is the largest.

    Else → num3 is the largest.

step4: Output the largest number.

step5: Stop
```

In [ ]:

```
#code:
# Program to find the largest among three numbers

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

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

# Display the result
print(f"The largest number is {largest}")
```

Input:

Code Enter first number: 12 Enter second number: 7 Enter third number: 9 Output:

Code The largest number is 12

In [ ]:
```
4.Write a Python program to check whether a given number is a prime number.
```

In [ ]:
```
algorithm:
step1: Start

step2: Input a number n.

step3: Check if n <= 1:

step4: If true → n is not prime. Go to Step 7.

step5: Initialize a loop variable i = 2.

      Repeat until i <= √n:

step6: If n % i == 0 → n is not prime. Go to Step 7.

     Else → increment i by 1 and continue.

step7: If no divisor is found in the loop → n is prime.

step8: Output whether n is prime or not.

step9: Stop
```

In [ ]:
```
#code:
# Program to check if a number is prime

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

# Prime numbers are greater than 1
if num > 1:
    # Check for factors
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            print(f"{num} is not a Prime number")
            break
    else:
        print(f"{num} is a Prime number")
else:
    print(f"{num} is not a Prime number")
```

Input: Code Enter a number: 7

Output: Code 7 is a Prime number

week-2

In [ ]:
```
5. Write a Python program to find the factorial of a number.
```

In [ ]:
```
algorithm:
step1: Start

step2: Input a number n.

step3: Check if n < 0:
```

```
step4: If true → Output: "Factorial does not exist for negative numbers." → Go to S

step5: Check if n == 0:

step6: If true → Output: "Factorial of 0 is 1." → Go to Step 7.

step7: Initialize fact = 1.

    Repeat for i = 1 to n:

    Multiply fact = fact * i.

step8: Output the value of fact.

step9: Stop
```

In [ ]:

```
#code:
# Program to find the factorial of a number

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

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

Input: Code Enter a number: 5

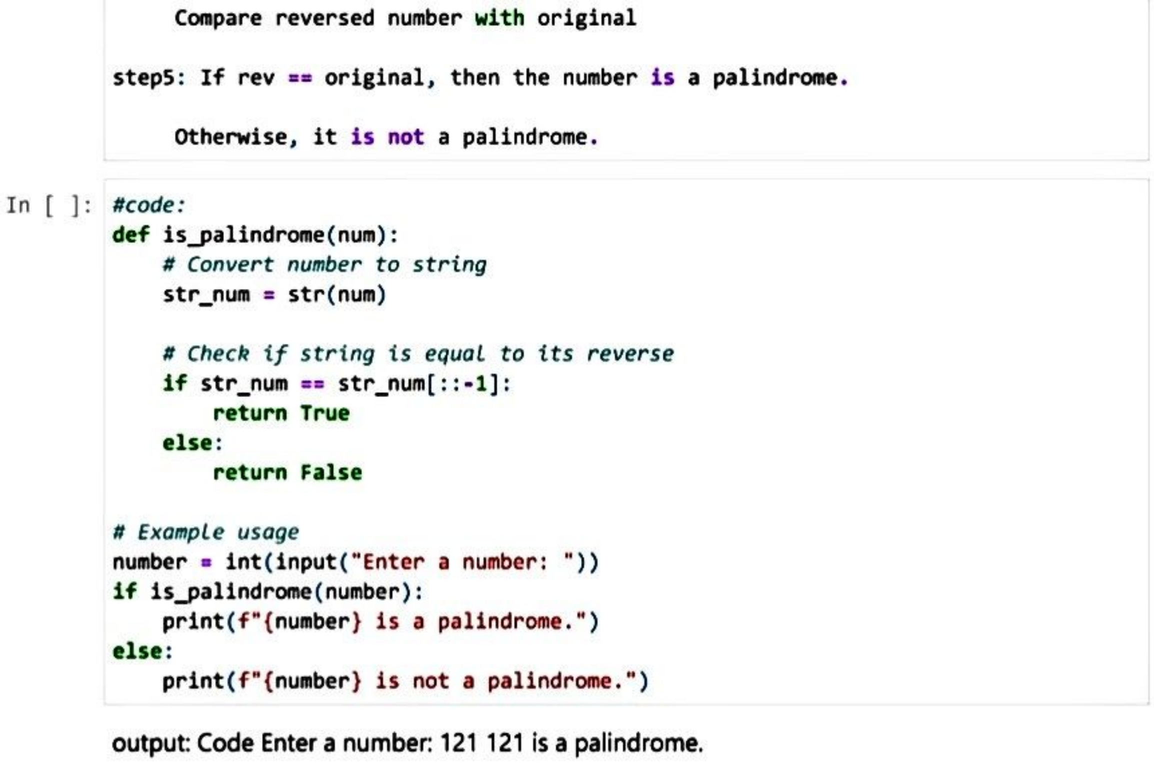
Output: Code The factorial of 5 is 120

In [ ]:

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

In [ ]:

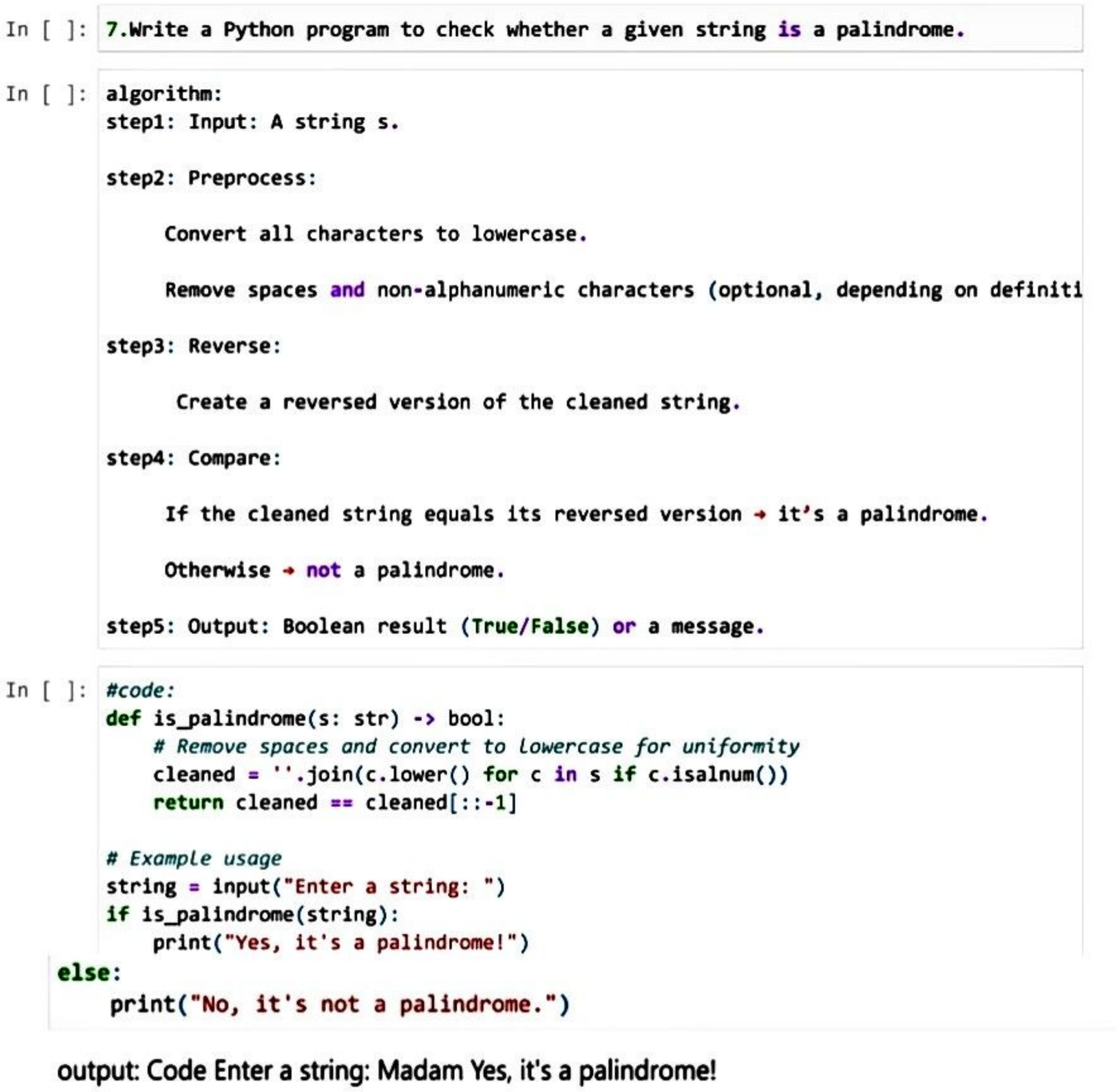
```
algorithm:
step1: Input the number

step2: Read the number n.

    Store the original number

    Keep a copy of n in a variable original.

    Reverse the number

step3: Initialize rev = 0.

step4:  While n > 0:

     Extract the last digit: digit = n % 10.

     Append it to rev: rev = rev * 10 + digit.

     Remove the last digit from n: n = n // 10.
```



```
     Compare reversed number with original

step5: If rev == original, then the number is a palindrome.

     Otherwise, it is not a palindrome.
```

```
In [ ]: #code:
        def is_palindrome(num):
            # Convert number to string
            str_num = str(num)

            # Check if string is equal to its reverse
            if str_num == str_num[::-1]:
                return True
            else:
                return False

        # Example usage
        number = int(input("Enter a number: "))
        if is_palindrome(number):
            print(f"{number} is a palindrome.")
        else:
            print(f"{number} is not a palindrome.")
```

output: Code Enter a number: 121 121 is a palindrome.



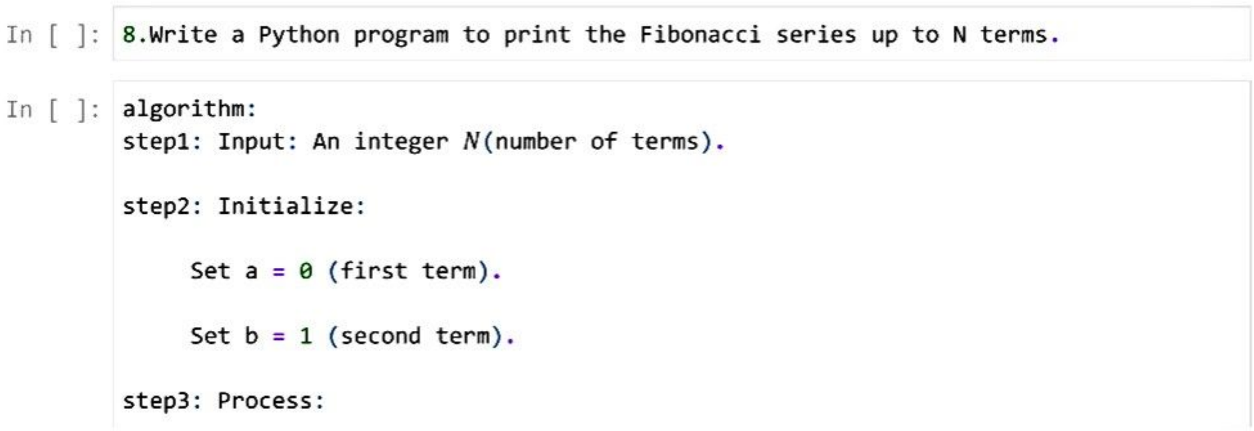
```
In [ ]: 7.Write a Python program to check whether a given string is a palindrome.
```

```
In [ ]: algorithm:
        step1: Input: A string s.

        step2: Preprocess:

             Convert all characters to lowercase.

             Remove spaces and non-alphanumeric characters (optional, depending on definiti

        step3: Reverse:

              Create a reversed version of the cleaned string.

        step4: Compare:

             If the cleaned string equals its reversed version → it’s a palindrome.

             Otherwise → not a palindrome.

        step5: Output: Boolean result (True/False) or a message.
```

```
In [ ]: #code:
        def is_palindrome(s: str) -> bool:
            # Remove spaces and convert to lowercase for uniformity
            cleaned = ''.join(c.lower() for c in s if c.isalnum())
            return cleaned == cleaned[::-1]

        # Example usage
        string = input("Enter a string: ")
        if is_palindrome(string):
            print("Yes, it's a palindrome!")
        else:
            print("No, it's not a palindrome.")
```

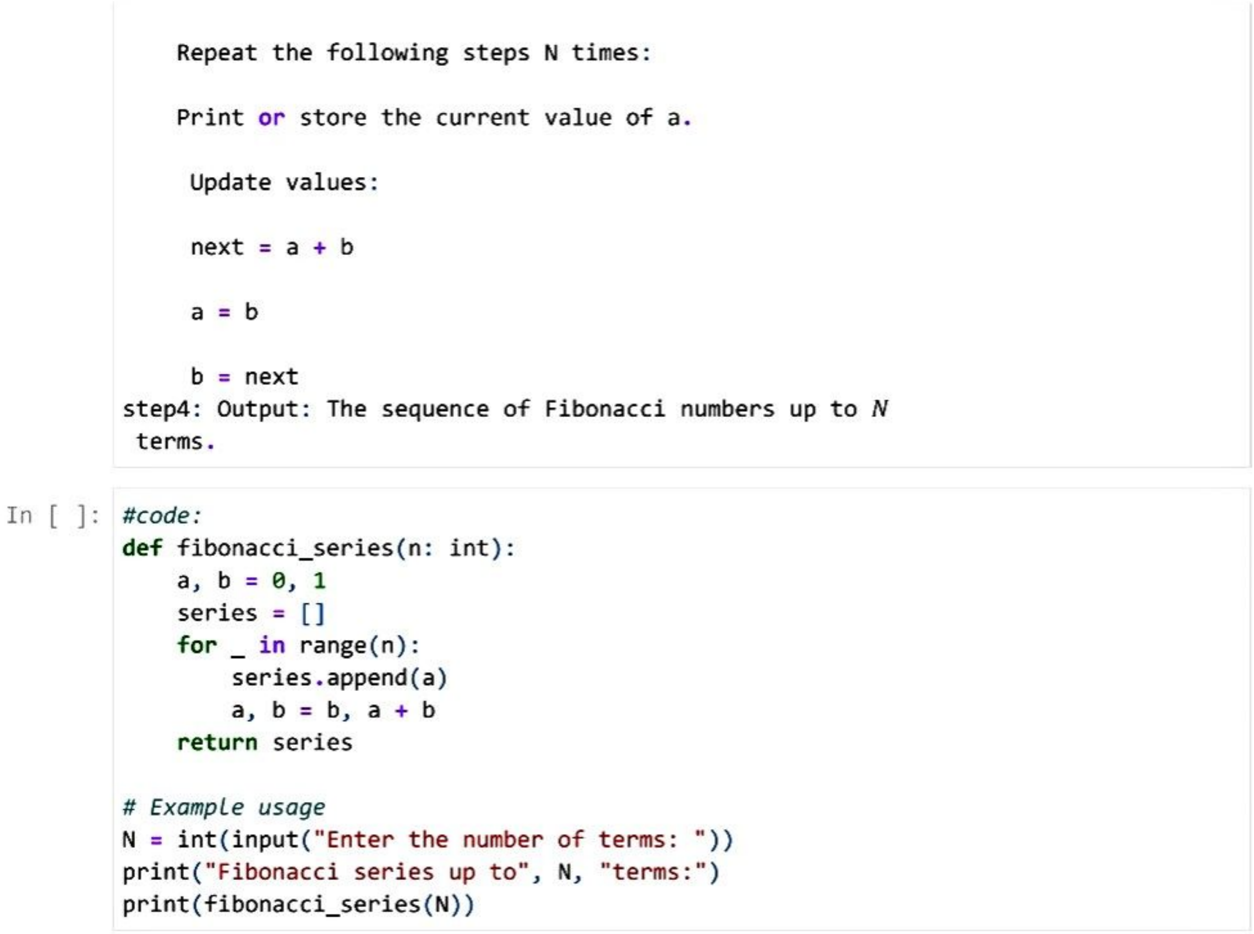
output: Code Enter a string: Madam Yes, it's a palindrome!

week-3



In [ ]: 8.Write a Python program to print the Fibonacci series up to N terms.

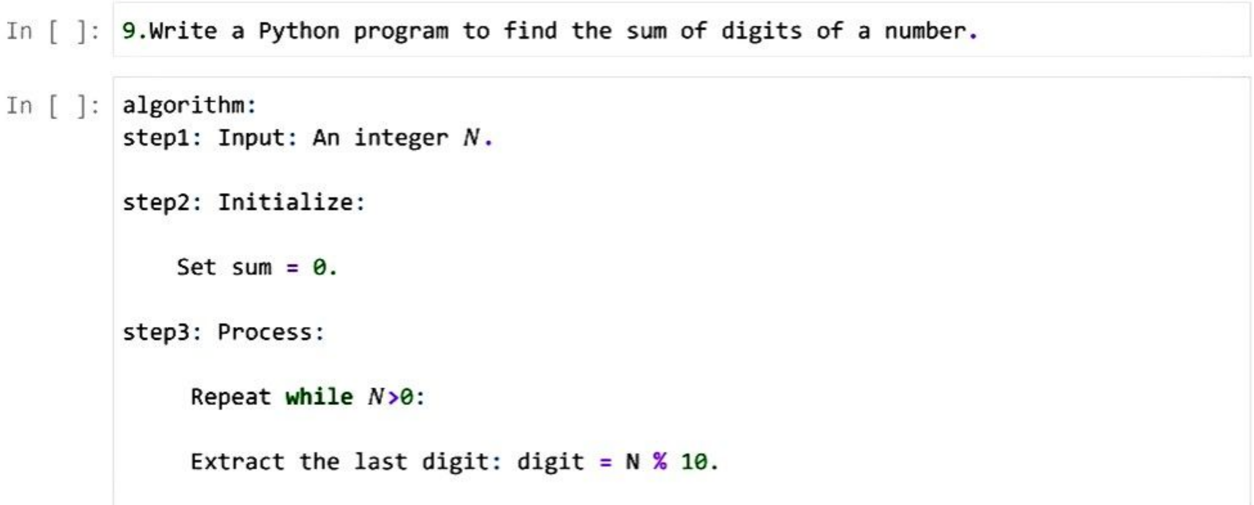
In [ ]:
```
algorithm:
step1: Input: An integer N(number of terms).

step2: Initialize:

     Set a = 0 (first term).

     Set b = 1 (second term).

step3: Process:
```



```
    Repeat the following steps N times:

    Print or store the current value of a.

     Update values:

     next = a + b

     a = b

     b = next
step4: Output: The sequence of Fibonacci numbers up to N
 terms.
```

In [ ]:
```python
#code:
def fibonacci_series(n: int):
    a, b = 0, 1
    series = []
    for _ in range(n):
        series.append(a)
        a, b = b, a + b
    return series

# Example usage
N = int(input("Enter the number of terms: "))
print("Fibonacci series up to", N, "terms:")
print(fibonacci_series(N))
```

output: Code Enter the number of terms: 5 Fibonacci series up to 5 terms:[0, 1, 1, 2, 3]



In [ ]: 9.Write a Python program to find the sum of digits of a number.

In [ ]:
```
algorithm:
step1: Input: An integer N.

step2: Initialize:

    Set sum = 0.

step3: Process:

     Repeat while N>0:

     Extract the last digit: digit = N % 10.
     Add it to the sum: sum = sum + digit.

     Remove the last digit: N = N // 10.

step4:  Output: The value of sum.
```

In [ ]:

```python
#code:
def sum_of_digits(n: int) -> int:
```

```python
    total = 0
    while n > 0:
        total += n % 10   # Extract last digit
        n //= 10          # Remove last digit
    return total

# Example usage
number = int(input("Enter a number: "))
print("Sum of digits:", sum_of_digits(number))
```

output: Code Enter a number: 123 Sum of digits: 6(Explanation: 1 + 2 + 3 = 6)

In [ ]:

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

In [ ]:

```
algorithm:
step1: Input: A string S.

step2: Initialize:

    vowel_count = 0

    consonant_count = 0

step3: Define vowels: vowels = {a, e, i, o, u} (both uppercase and lowercase).

      Process each character in the string:

     If the character is alphabetic:

     If the character is in vowels → increment vowel_count.

     Else → increment consonant_count.

     If the character is not alphabetic → ignore it.

step4: Output: Print vowel_count and consonant_count.
```

In [ ]:

```python
#code:
def count_vowels_consonants(s: str):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0

    for char in s:
        if char.isalpha():  # Check only alphabetic characters
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1

    return vowel_count, consonant_count
```

```
# Example usage
string = input("Enter a string: ")
vowels, consonants = count_vowels_consonants(string)
```

```
print("Number of vowels:", vowels)
print("Number of consonants:", consonants)
```

output: Code Enter a string: Hello World Number of vowels: 3 Number of consonants: 7(Explanation: vowels = e, o, o → 3; consonants = H, l, l, W, r, l, d → 7)

week-4

In [ ]:

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

In [ ]:

```
algorithm:
step1: Input: A string S.

step2: Initialize:

     Set reversed_str = "".

     Set index = length(S) - 1.

step3: Process:

    While index >= 0:

    Append S[index] to reversed_str.

    Decrease index by 1.

step3: Output: The value of reversed_str.
```

In [ ]:

```
#code:
def reverse_string(s: str) -> str:
    # Initialize an empty string to store the reversed result
    reversed_str = ""

    # Loop through the string from the end to the beginning
    index = len(s) - 1
    while index >= 0:
        reversed_str += s[index]   # Append each character
        index -= 1

    return reversed_str

# Example usage
string = input("Enter a string: ")
print("Reversed string:", reverse_string(string))
```

output: Code Enter a string: Hello Reversed string: olleH

In [ ]:

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

In [ ]:

```
algorithm:
step1: Input: A string S.
```

```
step2: Initialize:

     Create an empty dictionary (or map) called char_count.

step3: Process each character in the string:

     If the character already exists in char_count:

     Increment its value by 1.

Else:

     Add the character to char_count with value = 1.

step4:  Output: Display each character along with its count.
```

In [ ]:

```
#code:
def count_characters(s: str):
    char_count = {}
    for char in s:
        if char in char_count:
            char_count[char] += 1
        else:
            char_count[char] = 1
    return char_count

# Example usage
string = input("Enter a string: ")
result = count_characters(string)

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

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

In [ ]:

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

In [ ]:

```
algorithm:
step1: Input:

    Read first number (num1).

    Read operator (op).

    Read second number (num2).

step2: Initialize:

    Set result = 0.

step3: Process (using conditional checks):

     If op == '+' → result = num1 + num2.
    Else if op == '-' → result = num1 - num2.

    Else if op == '*' → result = num1 * num2.

    Else if op == '/':

    If num2 != 0 → result = num1 / num2.

    Else → print "Error! Division by zero".
    Else → print "Invalid operator".
```

In [ ]:

```
#code:
def calculator():
    print("Simple Calculator")
    print("Operations: +, -, *, /")

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

    # Conditional checks
    if operator == '+':
        result = num1 + num2
    elif operator == '-':
        result = num1 - num2
    elif operator == '*':
        result = num1 * num2
    elif operator == '/':
        if num2 != 0:
            result = num1 / num2
        else:
            return "Error! Division by zero."
    else:
        return "Invalid operator!"

    return f"Result: {result}"


# Example usage
print(calculator())
```

output: Code Simple Calculator Operations: +, -, *, / Enter first number: 10 Enter operator (+, -, *, /): + Enter second number: 5 Result: 15.0

week-5

In [ ]:

```
14.Write a Python program to implement a menu-driven calculator using a loop (repea
```

In [ ]:

```
algorithm:
step1: Start
```

```
step2: Display Menu:

Code
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
6.
step3: Input choicefrom user.

step4: Check choice:

     If choice = 5 → Print "Exit" and stop program.

     Else → Continue.

step5: Input two numbers: num1, num2.

step6: Perform operation based on choice:

     If choice = 1 → result = num1 + num2.
     If choice = 2 → result = num1 - num2.

     If choice = 3 → result = num1 * num2.

     If choice = 4 →

     If num2 ≠ 0 → result = num1 / num2.

    Else → Print "Error! Division by zero".

    Else → Print "Invalid choice".

step7: Display result.

step8: Repeat steps 2–7 until user selects Exit.
```

In [ ]:

```python
#code:
def menu_driven_calculator():
    while True:
        print("\n===== Menu-Driven Calculator =====")
        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 the calculator. Goodbye!")
            break

        # Take two numbers as input
```

```python
        try:
            num1 = float(input("Enter first number: "))
            num2 = float(input("Enter second number: "))
        except ValueError:
            print("Invalid input! Please enter numeric values.")
            continue
        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("Error! Division by zero.")
        else:
            print("Invalid choice! Please select from 1 to 5.")

# Run the calculator
menu_driven_calculator()
```

output: ===== Menu-Driven 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: 15.0

In [ ]:
```
15.Write a Python program to generate a multiplication table for a given number (lo
```

In [ ]:
```
algorithm:
step1: Start

step2: Repeat the following steps until the user chooses to stop:

      Input: Read a number N from the user.

Process:

     For i from 1 to 10:

    Compute product = N × i.

    Print N × i = product.

    Ask user: "Do you want to generate another table? (yes/no)"

    If the answer is "no" → Exit loop.



step3:End
```

In [ ]:
```
#code:
def multiplication_table():
    while True:
        try:
            # Ask the user for a number
            num = int(input("Enter a number to generate its multiplication table: "
            # Generate the multiplication table (1 to 10)
            print(f"\nMultiplication Table for {num}:")
            for i in range(1, 11):
                print(f"{num} x {i} = {num * i}")

            # Ask if the user wants to continue
            choice = input("\nDo you want to generate another table? (yes/no): ").s
            if choice != 'yes':
                print("Exiting program. Goodbye!")
                break
        except ValueError:
            print("Invalid input. Please enter a valid integer.\n")

# Run the program
multiplication_table()
```

output: Enter a number to generate its multiplication table: 5

```
Multiplication Table for 5:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

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

```
In [ ]: 16.Write a Python program to print different patterns using loop concepts (e.g., st
```

```
In [ ]: algorithm:
        step1: Input size (n)

               Decide how many rows the pattern should have (e.g., n = 5).

        step2: Loop through rows

              Use an outer loop (for i in range(1, n+1)) to control the number of rows.
```

```
        step3: Loop through columns

              Use an inner loop (for j in range(1, i+1) or similar) to control what gets pr

        step4: Decide what to print

               Stars (*), numbers (j or i), or spaces (" ") depending on the pattern type.

        step5: Print row output

               After finishing the inner loop, move to the next line (print()).
```

In [ ]:

```
#code:
# Compact Pattern Printing Examples

n = 5

# 1. Right-Angled Triangle
for i in range(1, n+1):
    print("*" * i)

print()

# 2. Inverted Right-Angled Triangle
for i in range(n, 0, -1):
    print("*" * i)

print()

# 3. Pyramid
for i in range(1, n+1):
    print(" " * (n-i) + "*" * (2*i-1))

print()

# 4. Number Triangle
for i in range(1, n+1):
    print(" ".join(str(j) for j in range(1, i+1)))

print()

# 5. Floyd’s Triangle
num = 1
for i in range(1, n+1):
    print(" ".join(str(num+j) for j in range(i)))
    num += i

print()

# 6. Diamond
for i in range(1, n+1):
    print(" " * (n-i) + "*" * (2*i-1))
for i in range(n-1, 0, -1):
    print(" " * (n-i) + "*" * (2*i-1))
```

In [ ]:

```
output:
  Right-Angled Triangle Code
*
**
***
****
*****
  Inverted Right-Angled Triangle Code
*****
****
***
**
*
  Pyramid Code
    *
   ***
  *****
 *******
*********
```

```
   Number Triangle Code
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

FUNCTION-BASED QUESTIONS

week-6

In [ ]:

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

In [ ]:

```
algorithm:
step1: Start

step2: Input name

     Accept the user’s name as a parameter to the function.

step3: Process

     Construct a greeting message using the given name.

      Example: "Hello, <name>! Welcome to Python programming."

step4: Output

     Print the greeting message on the screen.

step5: End
```

In [ ]:

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

# Example usage
greet_user("Alice")
greet_user("Rahul")
```

output: Hello, Alice! Welcome to Python programming. Hello, Rahul! Welcome to Python programming.

In [ ]:

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

In [ ]:

```
algorithm:
step1:Start

step2:Input two numbers

Accept two values as parameters (say a and b).

step3:Process

Compute the sum using the formula: sum = a + b.
```

```
step4:Output

Return the computed sum to the caller.

step5:End
```

In [ ]:

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

# Example usage:
result = add_numbers(5, 7)
print("The sum is:", result)
```

output: The sum is: 12 ✅ Explanation:

The function add_numbers(5, 7) returns 5 + 7 = 12.

The print statement displays the result in the format: "The sum is: 12".

week-7

In [ ]:

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

In [ ]:

```
algorithm:
step1: Start with an integer n.

step2: Check base case:

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

step3: Recursive case:

     Otherwise, return n * factorial(n - 1).

    Repeat until the base case is reached.

    End with the final result.
```

In [ ]:

```python
#code:
def factorial(n: int) -> int:
    """Return the factorial of n using recursion."""
    if n == 0 or n == 1:   # base case
        return 1
    else:
        return n * factorial(n - 1)  # recursive case

# Example usage:
print(factorial(5))  # Output: 120
```

output: print(factorial(0)) # 1 print(factorial(1)) # 1 print(factorial(4)) # 24 print(factorial(6)) # 720

In [ ]:

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

In [ ]:

```
algorithm:
step1: Start with an integer n.

step2: Check base case:

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

step3: Recursive case:

      Otherwise, compute n * factorial(n - 1).

step4: Repeat step 2-3 until the base case is reached.

step5: Return the final result.
```

In [ ]:

```
#code:
def factorial(n):
    # Base case: factorial of 0 or 1 is 1
    if n == 0 or n == 1:
        return 1
    else:
        # Recursive case: n! = n * (n-1)!
        return n * factorial(n - 1)


# Example usage:
print(factorial(5))  # Output: 120
```

output: Code 120 ✅ Explanation:

The function factorial(5) works recursively:

5!=5×4!

4!=4×3!

3!=3×2!

2!=2×1!

1!=1(base case)

So, 5!=5×4×3×2×1=120.

In [ ]:

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

In [ ]:

```
algorithm:
step1: Start

    Prompt the user to enter a number.

step2: Input Validation

    Check if the input is an integer.

    If not, display an error message and stop.

    Check if the integer is non-negative.

     If negative, display an error message and stop.

step3: Recursive Function Definition

    Define a function factorial(n):

    Base case: If n == 0 or n == 1, return 1.

step4: Recursive case:Otherwise, return n * factorial(n - 1).

step5: Computation

      Call the recursive function with the validated input.

step6: Output

      Display the result of the factorial calculation.

step7: End
```

In [ ]:

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

def main():
    try:
        # Take input from user
        num = int(input("Enter a non-negative integer: "))

        # Validate input
        if num < 0:
            print("Error: Factorial is not defined for negative numbers.")
        else:
            result = factorial(num)
            print(f"Factorial of {num} is {result}")

    except ValueError:
        print("Error: Please enter a valid integer.")
```

```python
# Run the program
if __name__ == "__main__":
    main()
```

In [ ]:

```
output:
Example 1:  Valid input
Code
  Enter a non-negative integer: 5
  Factorial of 5 is 120

Example 2: Input = 0
Code
  Enter a non-negative integer: 0
  Factorial of 0 is 1

Example 3: Negative input
Code
  Enter a non-negative integer: -3
  Error: Factorial is not defined for negative numbers.

Example 3: Negative input
Code
  Enter a non-negative integer: -3
  Error: Factorial is not defined for negative numbers.
   ✅ Explanation:

   The program uses recursion to compute factorial.

   It validates input to ensure only non-negative integers are accepted.

   Errors are handled gracefully with clear messages.
```

PROJECT / ADVANCED QUESTIONS

WEEK – 8

In [ ]:

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

In [ ]:

```
algorithm:
step1: Start

step2: Initialize → create empty list library.

step3: Functions

      add_book(title, author) → add dict {title, author, available=True} to library

      display_books() → if empty print "No books", else loop and show details.

      search_book(title) → loop, if match print details, else "Not found".
```

```
     borrow_book(title) → if found and available → set available=False; else show s

     return_book(title) → if found and not available → set available=True; else sho

step4: Menu

     Show options: Add, Display, Search, Borrow, Return, Exit.

     Take user choice → call function.
```

```
    Repeat until Exit.

step5: End
```

In [ ]:

```python
#code:
# Library Book Management System using Functions

library = []  # List to store books

def add_book(title, author):
    """Add a new book to the library"""
    book = {"title": title, "author": author, "available": True}
    library.append(book)
    print(f'Book "{title}" by {author} added successfully!')

def display_books():
    """Display all books in the library"""
    if not library:
        print("No books in the library yet.")
        return
    print("\nLibrary Books:")
    for idx, book in enumerate(library, start=1):
        status = "Available" if book["available"] else "Borrowed"
        print(f'{idx}. {book["title"]} by {book["author"]} - {status}')

def search_book(title):
    """Search for a book by title"""
    for book in library:
        if book["title"].lower() == title.lower():
            status = "Available" if book["available"] else "Borrowed"
            print(f'Found: "{book["title"]}" by {book["author"]} - {status}')
            return
    print(f'Book "{title}" not found in the library.')

def borrow_book(title):
    """Borrow a book if available"""
    for book in library:
        if book["title"].lower() == title.lower():
            if book["available"]:
                book["available"] = False
                print(f'You borrowed "{book["title"]}".')
            else:
                print(f'Sorry, "{book["title"]}" is already borrowed.')
            return
    print(f'Book "{title}" not found in the library.')
```

```python
def return_book(title):
    """Return a borrowed book"""
    for book in library:
        if book["title"].lower() == title.lower():
            if not book["available"]:
                book["available"] = True
                print(f'You returned "{book["title"]}".')
            else:
                print(f'"{book["title"]}" was not borrowed.')
            return
    print(f'Book "{title}" not found in the library.')

# Menu-driven program
def menu():
    while True:
        print("\n--- Library Menu ---")
        print("1. Add Book")
        print("2. Display Books")
        print("3. Search Book")
        print("4. Borrow Book")
        print("5. Return Book")
        print("6. Exit")

        choice = input("Enter your choice (1-6): ")

        if choice == "1":
            title = input("Enter book title: ")
            author = input("Enter book author: ")
            add_book(title, author)
        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 borrow: ")
            borrow_book(title)
        elif choice == "5":
            title = input("Enter book title to return: ")
            return_book(title)
        elif choice == "6":
            print("Exiting Library System. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")

# Run the program
menu()
```

output: --- Library Menu ---

1. Add Book
2. Display Books
3. Search Book
4. Borrow Book
5. Return Book
6. Exit Enter your choice (1-6): 1 Enter book title: Python Basics Enter book author: John Smith Book "Python Basics" by John Smith added successfully!

--- Library Menu --- Enter your choice (1-6): 2

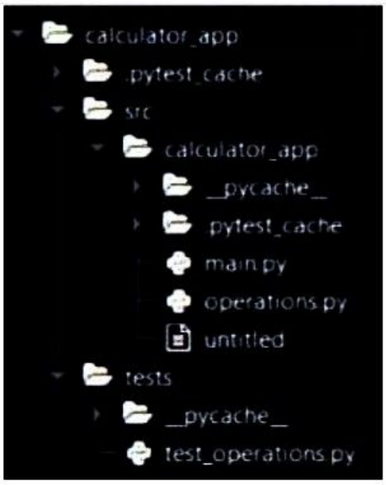
Library Books:

1. Python Basics by John Smith - Available

--- Library Menu --- Enter your choice (1-6): 6 Exiting Library System. Goodbye!

week-9

```
In [ ]: 23. Write a Python project to build a Calculator using modular programming (separat
```



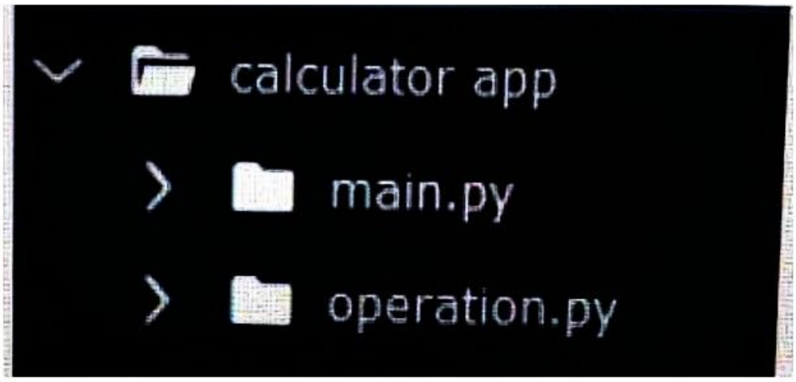
```
calculator_app
    .pytest_cache
    src
        calculator_app
            __pycache__
            .pytest_cache
            main.py
            operations.py
            untitled
    tests
        __pycache__
        test_operations.py
```

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")
```

```python
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(__test__), "..", ".
    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)
```



calculator app

main.py

operation.py

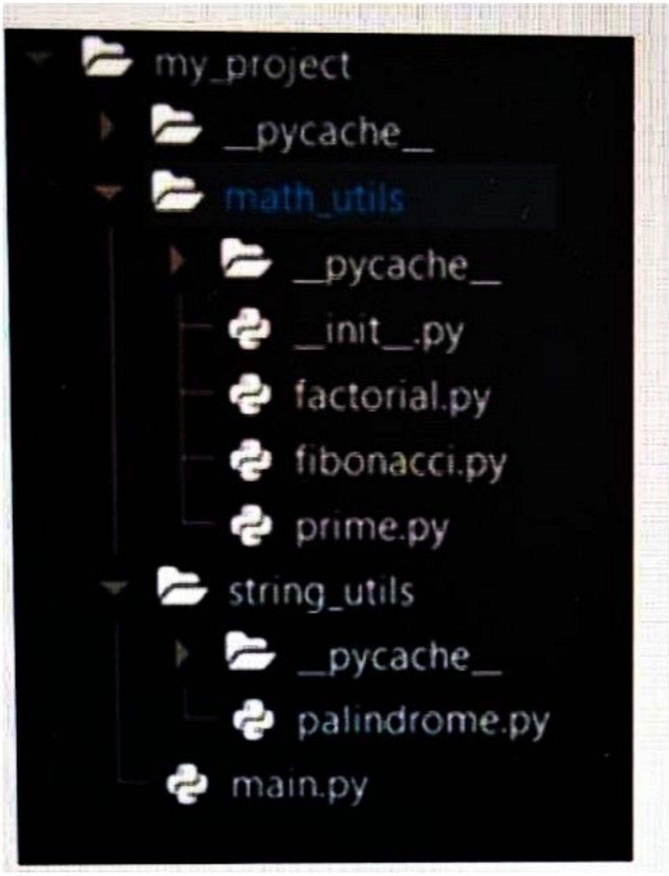
In [ ]:

```
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        return "Error: Division by zero"
    return a / b


from src.calculator_app.operations import add, subtract, multiply, divide

print("Simple Calculator")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")

choice = int(input("Enter your choice (1-4): "))
a = float(input("Enter first number: "))
b = float(input("Enter second number: "))

if choice == 1:
    print("Result:", operations.add(a, b))
elif choice == 2:
    print("Result:", operations.subtract(a, b))
elif choice == 3:
```

```
    print("Result:", operations.multiply(a, b))
elif choice == 4:
    print("Result:", operations.divide(a, b))
else:
    print("Invalid choice")
```

week-10

In [ ]: 24. Write a Python program that applies modular programming principles and define



```
my_project
    __pycache__
    math_utils
        __pycache__
        __init__.py
        factorial.py
        fibonacci.py
        prime.py
    string_utils
        __pycache__
        palindrome.py
    main.py
```

In [ ]:

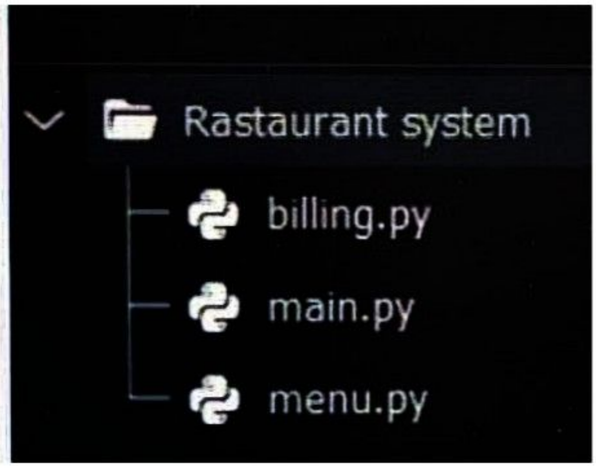
```
 # -*- coding: utf-8 -*
    # math_utils/factorial.py
def factorial(n):
     """Return factorial of a number using recursion."""
     if n == 0 or n == 1:
      return 1
      return n * factorial(n - 1)
      In [ ]:

# -*- 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]
```

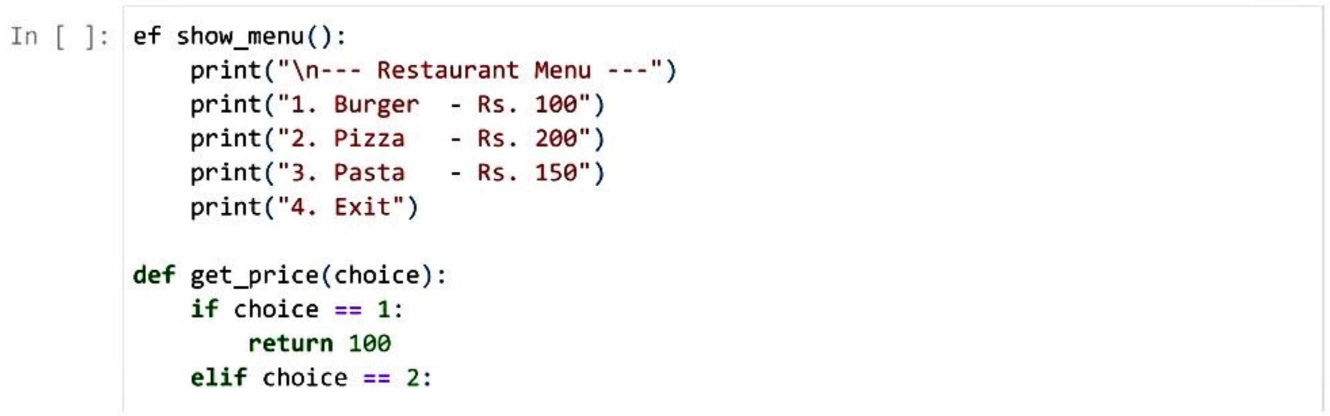
week-11

In [ ]:

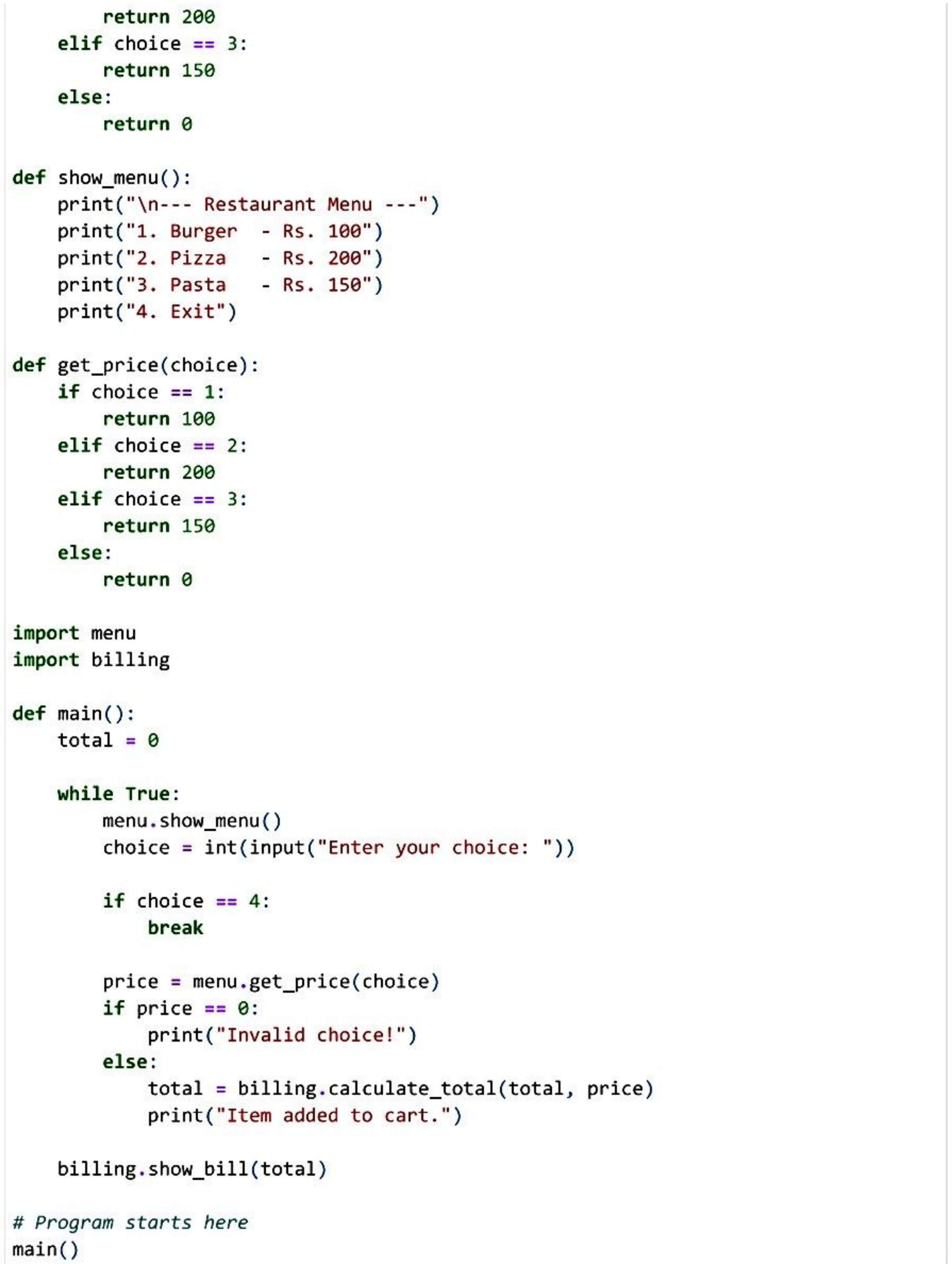
```
25.Write a Python program using modular programming principles and demonstrate:
Input validation
Testing (minimum 3 test cases)
Debugging practice with comments
```



Rastaurant system
billing.py
main.py
menu.py



```
In [ ]: ef show_menu():
            print("\n--- Restaurant Menu ---")
            print("1. Burger  - Rs. 100")
            print("2. Pizza   - Rs. 200")
            print("3. Pasta   - Rs. 150")
            print("4. Exit")

        def get_price(choice):
            if choice == 1:
                return 100
            elif choice == 2:
```

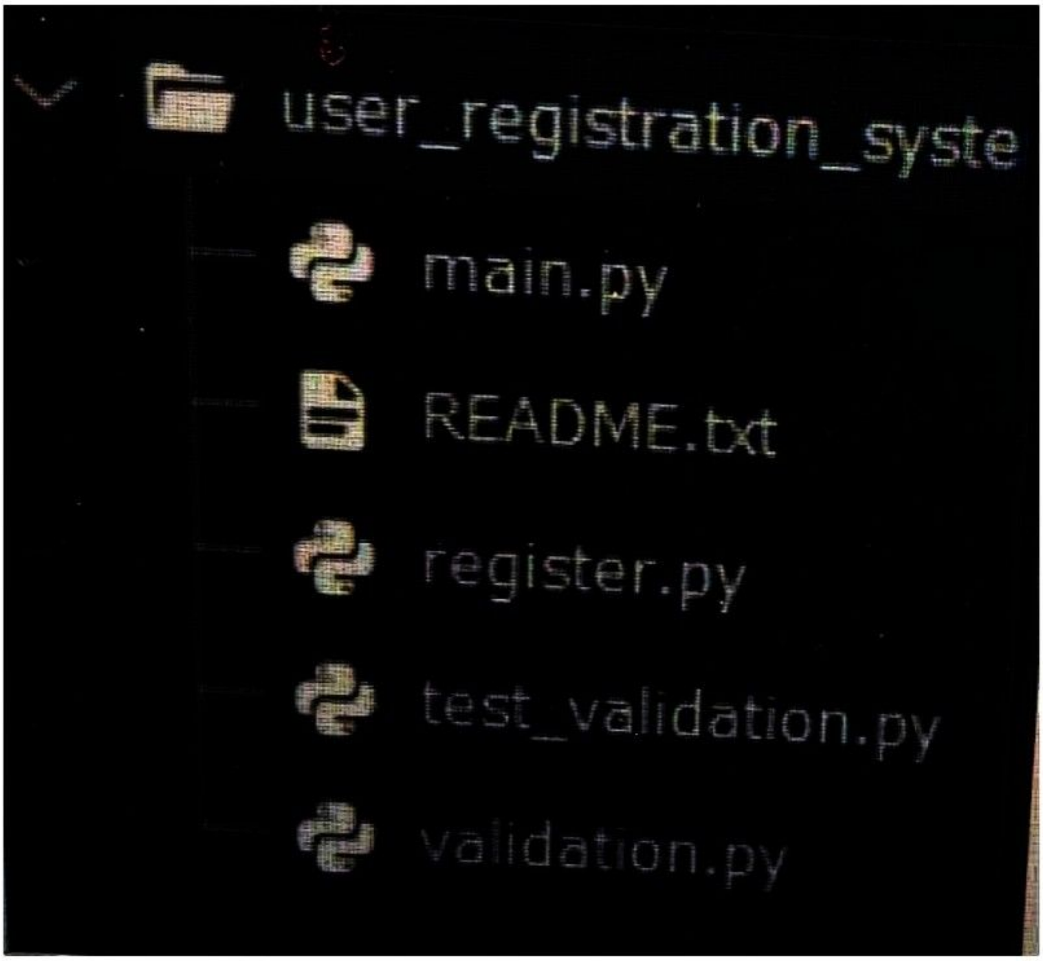


```
        return 200
    elif choice == 3:
        return 150
    else:
        return 0

def show_menu():
    print("\n--- Restaurant Menu ---")
    print("1. Burger  - Rs. 100")
    print("2. Pizza   - Rs. 200")
    print("3. Pasta   - Rs. 150")
    print("4. Exit")

def get_price(choice):
    if choice == 1:
        return 100
    elif choice == 2:
        return 200
    elif choice == 3:
        return 150
    else:
        return 0

import menu
import billing

def main():
    total = 0

    while True:
        menu.show_menu()
        choice = int(input("Enter your choice: "))

        if choice == 4:
            break

        price = menu.get_price(choice)
        if price == 0:
            print("Invalid choice!")
        else:
            total = billing.calculate_total(total, price)
            print("Item added to cart.")

    billing.show_bill(total)

# Program starts here
main()
```

week-12

In [ ]:

```
26.Write a Python project for a User Registration System with input validation, tes
```



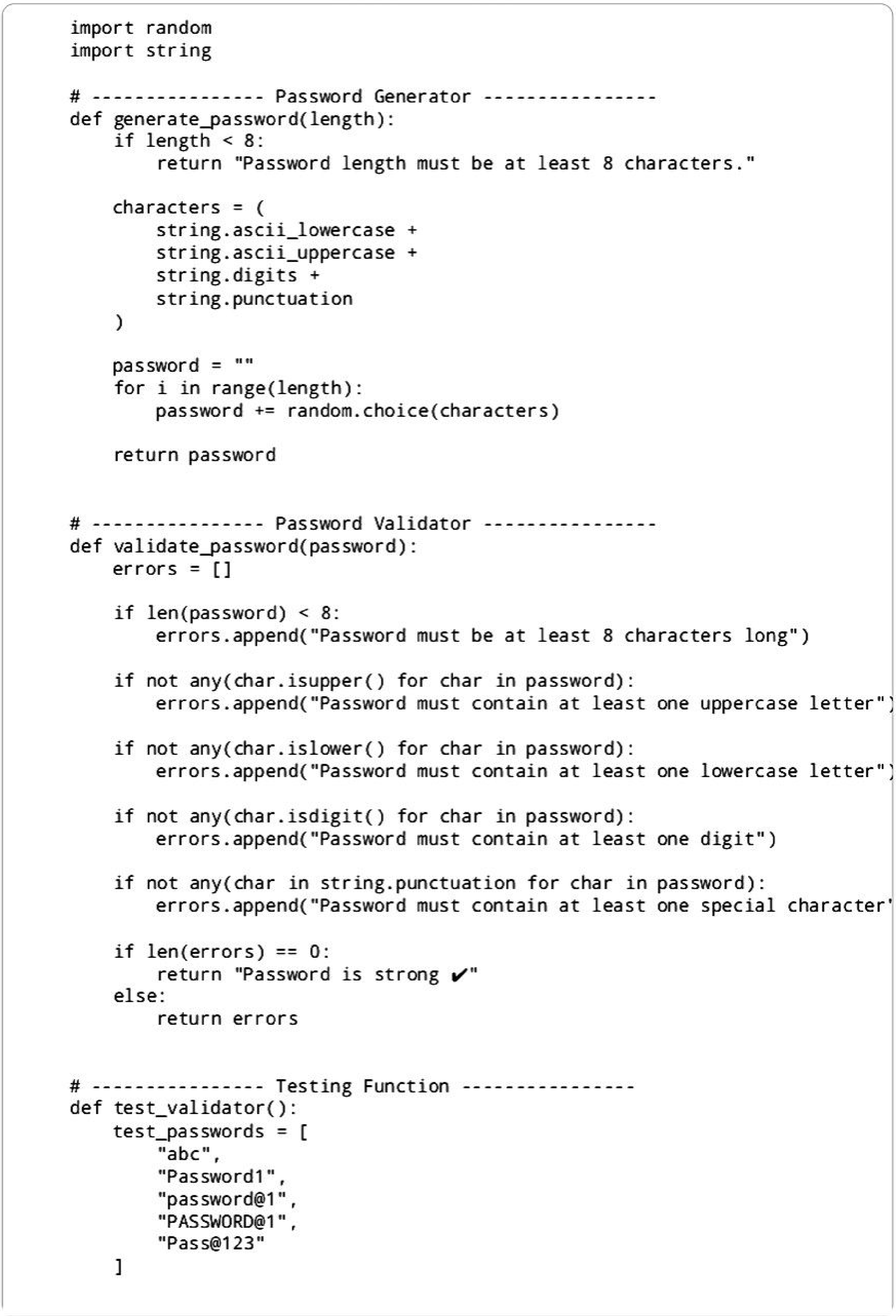
```
user_registration_syste
    main.py
    README.txt
    register.py
    test_validation.py
    validation.py
```

In [ ]:

```python
def validate_username(username):
    if len(username) < 3:
        return False
    return True

def validate_password(password):
    if len(password) < 6:
        return False
    return True

def validate_age(age):
    if age < 18:
        return False
    return True


import validation

def register_user(username, password, age):
    if not validation.validate_username(username):
        return "Invalid username (min 3 characters)"

    if not validation.validate_password(password):
        return "Invalid password (min 6 characters)"
    if not validation.validate_age(age):
        return "Age must be 18 or above"

    return "User registered successfully!"


    import register

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

    username = input("Enter username: ")
    password = input("Enter password: ")
    age = int(input("Enter age: "))

    result = register.register_user(username, password, age)
    print(result)

# Program starts here
main()


import validation

def test_validation():
    print("Testing Validation Functions")

    print(validation.validate_username("ab"))     # False
    print(validation.validate_username("user"))   # True

    print(validation.validate_password("123"))    # False
    print(validation.validate_password("123456")) # True

    print(validation.validate_age(16))               # False
    print(validation.validate_age(20))               # True

test_validation()


user registration system documention

1. Input validation:
 - username must be at least 3 characters
 - password must be at least 6 characters
 - age must be 18 or above

2. Testing:
 - test_validation.py is used to test validation functions
 - each functions is tested with valid and invalid inputs

3. Debugging:
 - errors like invalid inputs are handled using validation functions
 - modular design help identify bugs easily
 - each module can be tested independently

4. Conclusion:
```

```
 - the system users modular programming
 - code is reusable , readable , and easy to maintain
```

27.write a mini project in python programming where is programming concepts (loops, function,list, modules, validation, testing)



```
import random
import string

# ---------------- Password Generator ----------------
def generate_password(length):
    if length < 8:
        return "Password length must be at least 8 characters."

    characters = (
        string.ascii_lowercase +
        string.ascii_uppercase +
        string.digits +
        string.punctuation
    )

    password = ""
    for i in range(length):
        password += random.choice(characters)

    return password


# ---------------- Password Validator ----------------
def validate_password(password):
    errors = []

    if len(password) < 8:
        errors.append("Password must be at least 8 characters long")

    if not any(char.isupper() for char in password):
        errors.append("Password must contain at least one uppercase letter")

    if not any(char.islower() for char in password):
        errors.append("Password must contain at least one lowercase letter")

    if not any(char.isdigit() for char in password):
        errors.append("Password must contain at least one digit")

    if not any(char in string.punctuation for char in password):
        errors.append("Password must contain at least one special character"

    if len(errors) == 0:
        return "Password is strong ✔"
    else:
        return errors


# ---------------- Testing Function ----------------
def test_validator():
    test_passwords = [
        "abc",
        "Password1",
        "password@1",
        "PASSWORD@1",
        "Pass@123"
    ]
    for pwd in test_passwords:
        print(f"Testing Password: {pwd}")
        print(validate_password(pwd))
        print("-" * 40)


# ---------------- Main Program ----------------
def main():
    while True:
        print("\n----- Password Generator & Validator -----")
        print("1. Generate Password")
        print("2. Validate Password")
        print("3. Test Validator")
        print("4. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            length = int(input("Enter password length: "))
            password = generate_password(length)
            print("Generated Password:", password)

        elif choice == "2":
            pwd = input("Enter password to validate: ")
            result = validate_password(pwd)
            print(result)

        elif choice == "3":
            test_validator()

        elif choice == "4":
            print("Exiting program...")
            break

        else:
            print("Invalid choice. Try again.")


# Program Execution
main()
```

```
----- Password Generator & Validator -----
1. Generate Password
2. Validate Password
3. Test Validator
4. Exit
Generated Password: xp*#eZ=};

----- Password Generator & Validator -----
1. Generate Password
2. Validate Password
3. Test Validator
4. Exit
Enter your choice: 
```