SR UNIVERSITY

Lab Assignment # 1

Name of Student : B. Sai Rithvik
Enrollment No. : 2303A51536
Batch No. : 22

Task 1: AI-Generated Logic Without Modularization (Factorial without Functions)

Prompt-

Write a Python program that calculates the factorial of a given positive integer using an iterative loop only. Do not use functions or recursion. The entire logic should be written in a single block of code.

Code-

session 1.2.py > ...

```
#Write a simple Python program to find the factorial of a number using loops, without using any functions.
import math
num = 5
factorial = 1
for i in range(1, num + 1):
    factorial *= i
print("Factorial of", num, "is", factorial)

```

Output-

```
PS C:\Users\Dell\Desktop\PYTHON> & C:/Users/Dell/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Dell/Desktop/PYTHON/session 1.2.p
y"
Factorial of 5 is 120
```

Justification-

This code solves the problem in a very straightforward way by writing everything in one place. It is easy for beginners to understand and good for small tasks. However, since it does not use functions, the code cannot be reused easily in bigger programs.

SR UNIVERSITY

Task 2: AI Code Optimization & Cleanup (Improving Efficiency)

Prompt-

Optimize the factorial program created in Task 1 by simplifying the logic, removing redundant variables or statements, and enhancing code readability while still avoiding the use of functions and recursion.

Code-

```
task2.py > ...
1   #Improve the above factorial program by making the code simpler, removing extra variables, and making it easier to read.
2   import math
3   num = 8
4   factorial = 1
5   for i in range(1, num + 1):
6       factorial *= i
7   print(f"Factorial of {num} is {factorial}")
8
```

Output-

```
PS C:\Users\Dell\Desktop\PYTHON> & C:/Users/Dell/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Dell/Desktop/PYTHON/task2.py
Factorial of 8 is 40320
PS C:\Users\Dell\Desktop\PYTHON>
```

Justification-

This improved version looks cleaner and more organized. It removes unnecessary parts and focuses only on what is needed. Such clean code is easier to understand, debug, and maintain, especially when working with others.

SR UNIVERSITY

Task 3: Modular Design Using AI Assistance (Factorial with Functions)

Prompt-

Develop a Python program to compute the factorial of a number by defining and using a user-defined function. The function should take the number as input and return the factorial value.

Code-

```python
#Write a Python program to calculate the factorial of a number using a function.
import math
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
num = 6
print(f"Factorial of {num} is {factorial(num)}")
```

Output-

```
PS C:\Users\Dell\Desktop\PYTHON> & C:/Users/Dell/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Dell/Desktop/PYTHON/task2.py
Factorial of 8 is 40320
PS C:\Users\Dell\Desktop\PYTHON> & C:/Users/Dell/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Dell/Desktop/PYTHON/task3.py
Factorial of 6 is 720
PS C:\Users\Dell\Desktop\PYTHON>
```

Justification-

Using a function makes the program more structured and professional. The factorial logic is separated from the main code, which makes it reusable and easy to modify later. This is a better approach for larger programs.

SR UNIVERSITY

Task 4: Comparative Analysis – Procedural vs Modular AI Code (With vs Without Functions)

Prompt-

Create a Python program that calculates factorial in two ways: (1) without using functions and (2) using a user-defined function. Present a comparison of both approaches in a table based on readability, reusability, ease of debugging, and suitability for large-scale applications.

Code-

```python
#Write a program that compares factorial code written without functions and with functions
import math
print("Factorial of 5 without function:")
num = 5
factorial = 1
for i in range(1, num + 1):
    factorial *= i
print("Factorial of", num, "is", factorial)
print("\nFactorial of 6 with function:")
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
num = 6
print(f"Factorial of {num} is {factorial(num)}")
```

Output-

```
PS C:\Users\Dell\Desktop\PYTHON> & C:/Users/Dell/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Dell/Desktop/PYTHON/task4.py
Factorial of 5 without function:
Factorial of 5 is 120

Factorial of 6 with function:
Factorial of 6 is 720
```

Justification-

This comparison helps us clearly see the advantages of using functions. Code with functions is easier to understand and debug and can be reused in many places. It is more suitable for large and real-world applications than single-block code.

SR UNIVERSITY

Task 5: AI-Generated Iterative vs Recursive Thinking

Prompt-

Implement a Python program that calculates the factorial of a number using both iterative and recursive approaches. Explain how each method works and compare them with respect to readability, memory usage, performance, and practical limitations.

Code-

task5.py > ...

```python
#Write a program that calculates factorial using both iterative and recursive methods
import math
def iterative_factorial(n):
    """Calculate factorial using an iterative approach."""
    factorial = 1
    for i in range(1, n + 1):
        factorial *= i
    return factorial
def recursive_factorial(n):
    """Calculate factorial using a recursive approach."""
    if n == 0 or n == 1:
        return 1
    else:
        return n * recursive_factorial(n - 1)
num = 7
print(f"Iterative Factorial of {num} is {iterative_factorial(num)}")
print(f"Recursive Factorial of {num} is {recursive_factorial(num)}")

```

Output-

```
PS C:\Users\Dell\Desktop\PYTHON> & C:/Users/Dell/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Dell/Desktop/PYTHON/task5.py
Iterative Factorial of 7 is 5040
Recursive Factorial of 7 is 5040
```

Justification-

This task shows two different ways of solving the same problem. The iterative method is simple and safe for large numbers, while the recursive method is easier to understand logically but can cause memory issues for big inputs. Knowing both methods helps in choosing the right solution.