AI Assisted Coding

ASSIGNMENT-01

Name : G. Abhiram

Hallticket:2303A51087

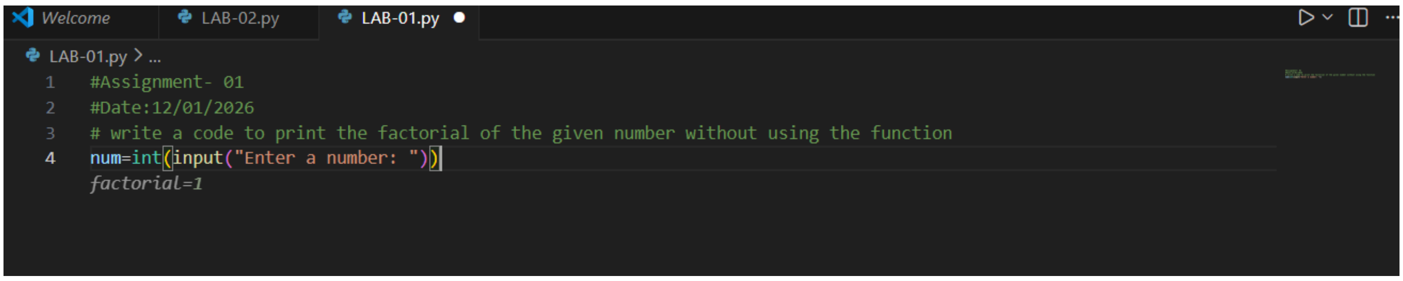
Batch-02

Task:01

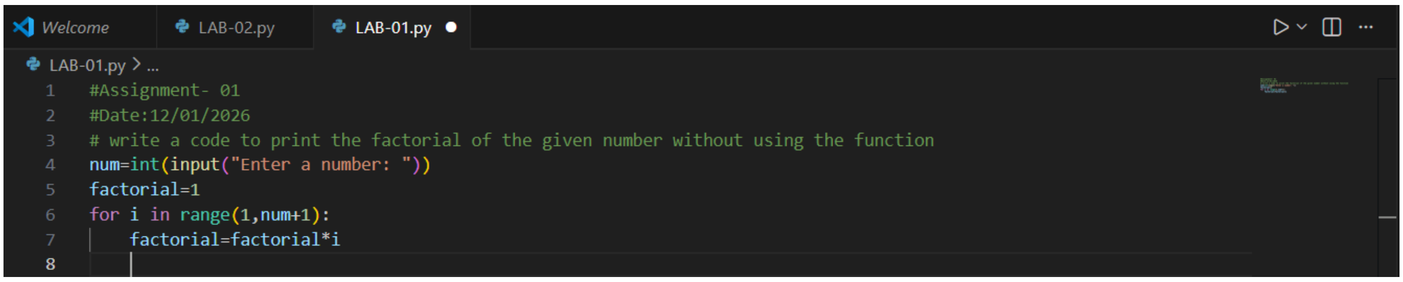
- **A working Python program generated with Copilot assistance**

```
❖ #Assingment-01
❖ #Date:12/01/2026
❖ #write a code to print the factorial of the given number without using the function
❖ num = int(input("Enter a number: "))
❖ factorial = 1
❖ for i in range(1, num + 1):
❖     factorial *= i
❖ print(f"The factorial of {num} is {factorial}")
```

- **Screenshot(s) showing:**

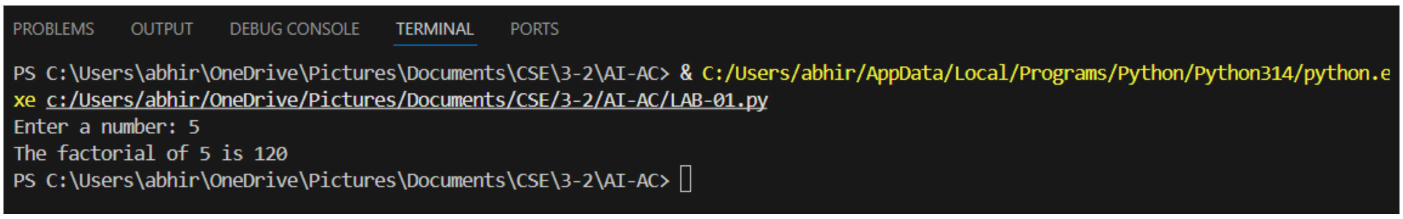


```
#Assignment- 01
#Date:12/01/2026
# write a code to print the factorial of the given number without using the function
num=int(input("Enter a number: "))
factorial=1
```

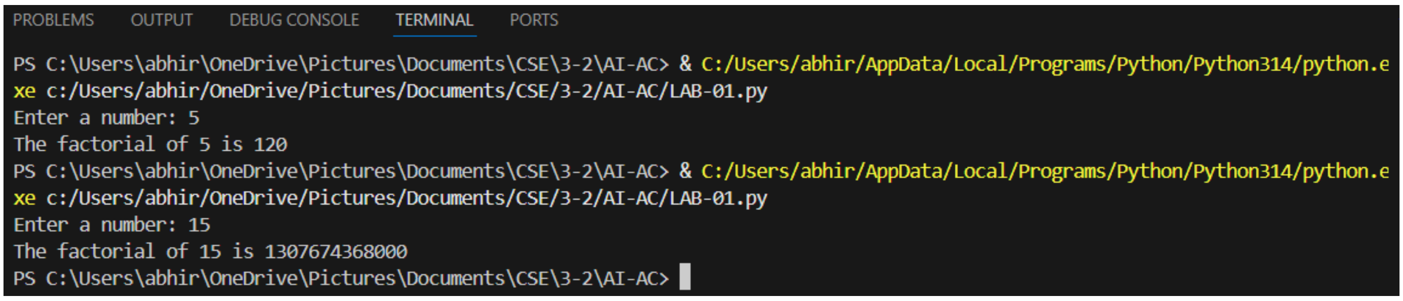


```
#Assignment- 01
#Date:12/01/2026
# write a code to print the factorial of the given number without using the function
num=int(input("Enter a number: "))
factorial=1
for i in range(1,num+1):
    factorial=factorial*i

```

- **Sample input/output screenshots**



```
PS C:\Users\abhir\OneDrive\Pictures\Documents\CSE\3-2\AI-AC> & C:/Users/abhir/AppData/Local/Programs/Python/Python314/python.e
xe c:/Users/abhir/OneDrive/Pictures/Documents/CSE/3-2/AI-AC/LAB-01.py
Enter a number: 5
The factorial of 5 is 120
PS C:\Users\abhir\OneDrive\Pictures\Documents\CSE\3-2\AI-AC>
```



```
PS C:\Users\abhir\OneDrive\Pictures\Documents\CSE\3-2\AI-AC> & C:/Users/abhir/AppData/Local/Programs/Python/Python314/python.exe c:/Users/abhir/OneDrive/Pictures/Documents/CSE/3-2/AI-AC/LAB-01.py
Enter a number: 5
The factorial of 5 is 120
PS C:\Users\abhir\OneDrive\Pictures\Documents\CSE\3-2\AI-AC> & C:/Users/abhir/AppData/Local/Programs/Python/Python314/python.exe c:/Users/abhir/OneDrive/Pictures/Documents/CSE/3-2/AI-AC/LAB-01.py
Enter a number: 15
The factorial of 15 is 1307674368000
PS C:\Users\abhir\OneDrive\Pictures\Documents\CSE\3-2\AI-AC>
```

- **Brief reflection (5–6 lines):**

  ->GitHub Copilot was used to generate the logic quickly without putting much thought into the program structure. It advocated a fresh and clean approach to code and made it readable without having functions in use. Following its suggestions, I reduced the need for me to memorize exact syntax, freeing my mind to focus on the core logic. Copilot also helped me write simple and efficient code in a neat way. Overall, it was faster to develop, less stressful and better in terms of clarity.

- **How helpful was Copilot for a beginner?**

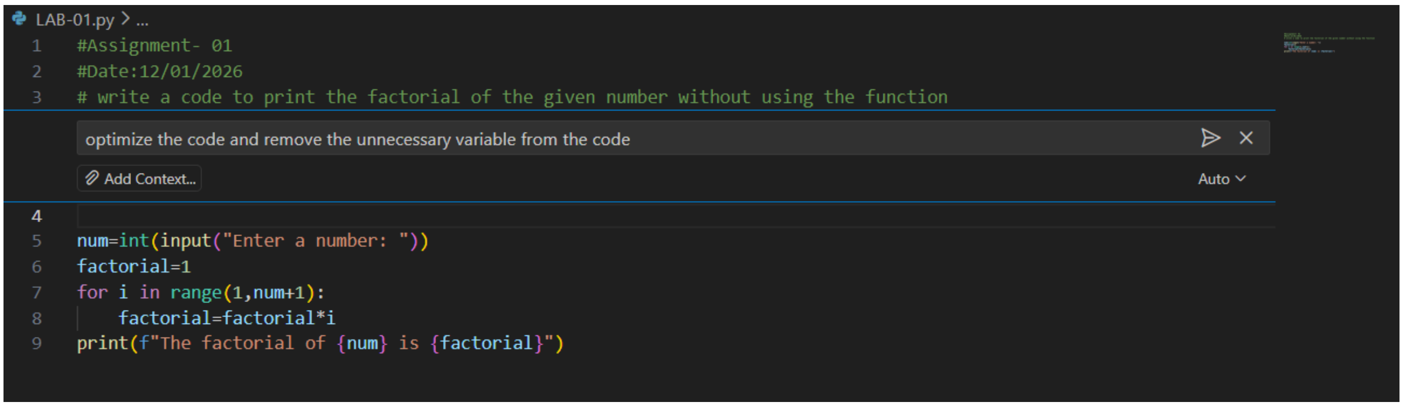
  ->copilot is very useful for beginners because it provides instant code suggestion when the user is stuck in middle. It will help beginner by suggesting the code so that user can understand the code and memorized and it makes learning programming easier and more confident

- **Did it follow best practices automatically?**

  ->Yes it follows best practices automatically for user by writing the code itself copilot will guess the next lines by that user can also more confident and user also get to know what next should be .It automatically best practices

Task-02

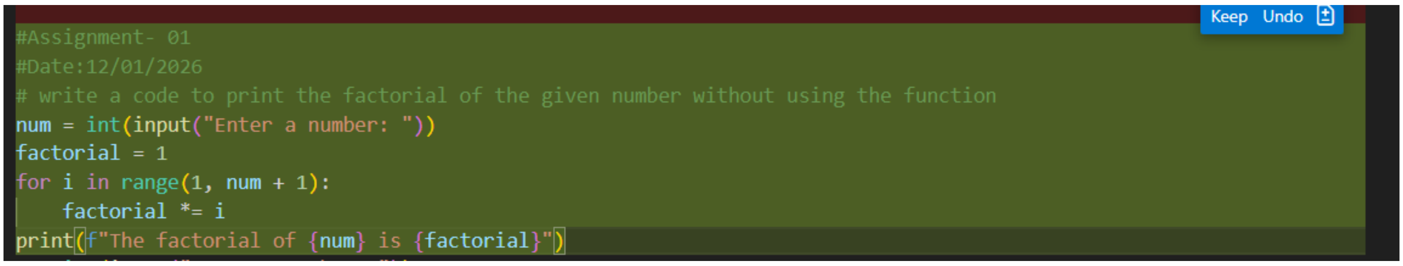
- **Screenshot(s) showing:**



```
#Assignment- 01
#Date:12/01/2026
# write a code to print the factorial of the given number without using the function
```

optimize the code and remove the unnecessary variable from the code

```

num=int(input("Enter a number: "))
factorial=1
for i in range(1,num+1):
    factorial=factorial*i
print(f"The factorial of {num} is {factorial}")
```



```
#Assignment- 01
#Date:12/01/2026
# write a code to print the factorial of the given number without using the function
num = int(input("Enter a number: "))
factorial = 1
for i in range(1, num + 1):
    factorial *= i
print(f"The factorial of {num} is {factorial}")
```

code

```
#Assingment-01
#Date:12/01/2026
#write a code to print the factorial of the given number without using the
function
num = int(input("Enter a number"))
factorial = 1
for i in range(1, num + 1):
    factorial *= i
print(f"The factorial of {num} is {factorial}")
```

side by side comparition

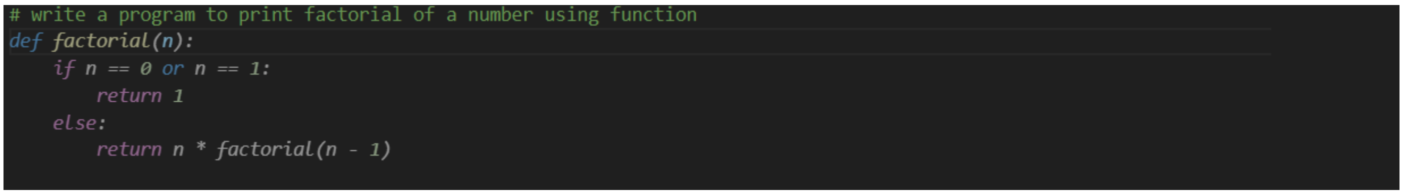
```
n=int(input("Enter a number: "))
factorial = 1
for i in range(1, n + 1):
     factorial = factorial * i
print("Factorial of", n, "is", factorial)
```

optimal code

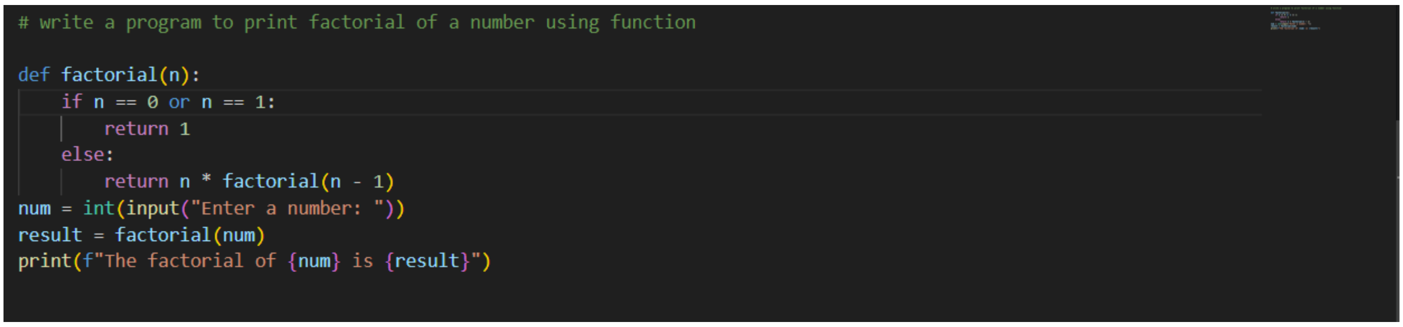
```
n = int(input("Enter a number: "))
factorial = 1
for i in range(2, n + 1):
    factorial *= i
print(f"Factorial of {n} is {factorial}")
```

task-03

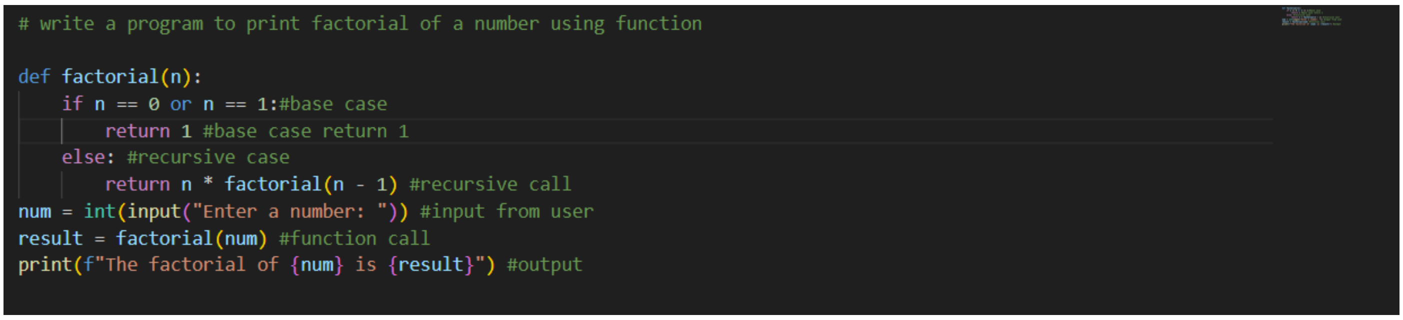
Modular Design Using AI Assistance (Factorial with Functions)



```python
# write a program to print factorial of a number using function
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
```



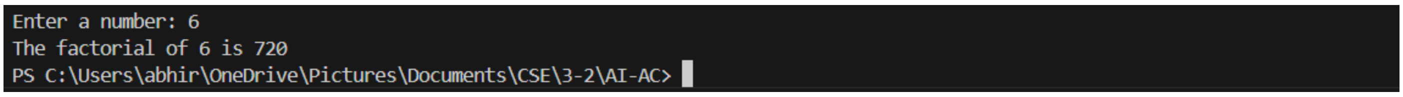
```python
# write a program to print factorial of a number using function

def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
num = int(input("Enter a number: "))
result = factorial(num)
print(f"The factorial of {num} is {result}")
```



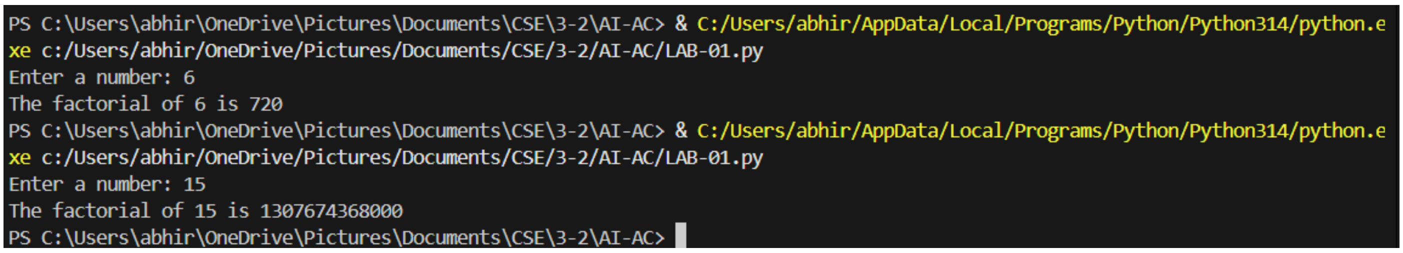
```python
# write a program to print factorial of a number using function

def factorial(n):
    if n == 0 or n == 1:#base case
        return 1 #base case return 1
    else: #recursive case
        return n * factorial(n - 1) #recursive call
num = int(input("Enter a number: ")) #input from user
result = factorial(num) #function call
print(f"The factorial of {num} is {result}") #output
```

```python
#write a program to print factorial of the given number by using the fuction
def factorial(n):
    if n == 0 or n == 1: #base case
        return 1  #base case return 1
    else: #recursive case
        return n*factorial(n-1) #recursive call
number = int(input("Enter a number: "))  #input from user
result=factorial(num) #function call
print(f"The factorial of {num} is {result}") #output
```

- **Sample input/output screenshots**



```
Enter a number: 6
The factorial of 6 is 720
PS C:\Users\abhir\OneDrive\Pictures\Documents\CSE\3-2\AI-AC>
```


```
PS C:\Users\abhir\OneDrive\Pictures\Documents\CSE\3-2\AI-AC> & C:/Users/abhir/AppData/Local/Programs/Python/Python314/python.e
xe c:/Users/abhir/OneDrive/Pictures/Documents/CSE/3-2/AI-AC/LAB-01.py
Enter a number: 6
The factorial of 6 is 720
PS C:\Users\abhir\OneDrive\Pictures\Documents\CSE\3-2\AI-AC> & C:/Users/abhir/AppData/Local/Programs/Python/Python314/python.e
xe c:/Users/abhir/OneDrive/Pictures/Documents/CSE/3-2/AI-AC/LAB-01.py
Enter a number: 15
The factorial of 15 is 1307674368000
PS C:\Users\abhir\OneDrive\Pictures\Documents\CSE\3-2\AI-AC>
```

Short note:

How modularity improves reusability.

->Modularity improves reusability by breaking a program into independent, self-contained parts such as functions or modules. Each module can be reused in different programs or in multiple places within the same program without rewriting code. This reduces duplication and saves development time. Modular code is also easier to test, debug, and maintain because changes in one module do not affect the entire program. Overall, modularity makes software more flexible and scalable..

Task 4:

Comparative Analysis – Procedural vs Modular AI Code (With vs Without Functions

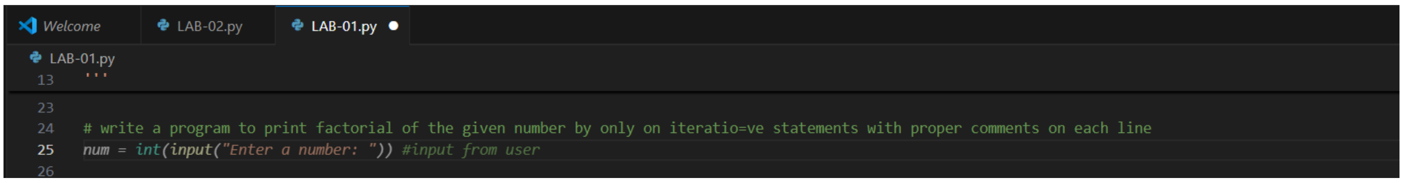
· **Comparative Analysis: Procedural vs Modular AI Code**

Criteria	**Procedural Code (Without Functions)**	**Modular Code (With Functions)**
Logic Clarity	The logic is written in a single block, which is easy to understand for small programs but becomes difficult to follow as the code grows.	The logic is organized inside a function, making the program more structured and easier to understand.
Reusability	The code cannot be reused easily and must be rewritten if the same logic is needed again.	The function can be reused multiple times across the program or in other programs.
Debugging Ease	Debugging is harder because the entire code must be checked for errors.	Debugging is easier since errors are confined within the function.
Suitability for Large Projects	This approach is not suitable for large projects due to poor structure and maintainability.	This approach is suitable for large projects because modular design improves organization and scalability.
AI Dependency Risk	Since the logic is simple, dependency on AI is minimal.	Over-dependence on AI-generated functions without

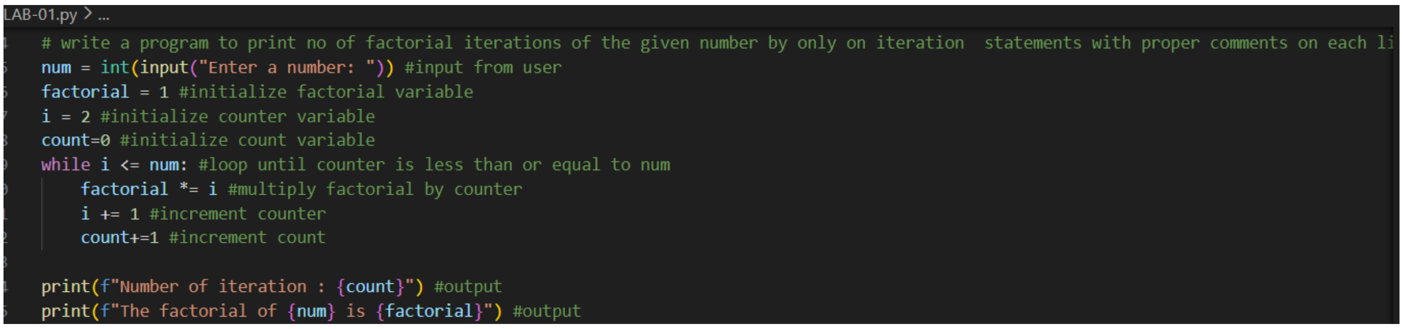
		understanding the logic can introduce risks.

Task 5:

AI-Generated Iterative vs Recursive Thinking



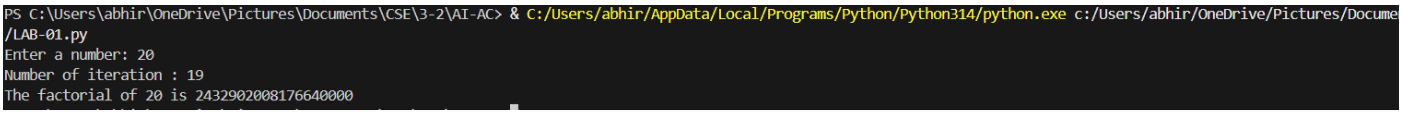
```
LAB-01.py
    '''

# write a program to print factorial of the given number by only on iteratio=ve statements with proper comments on each line
num = int(input("Enter a number: ")) #input from user
```



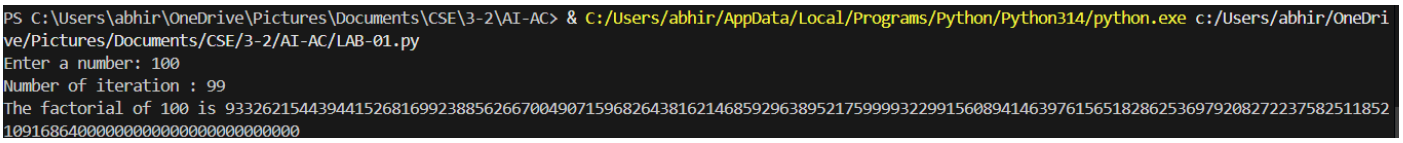
```
LAB-01.py > ...
# write a program to print no of factorial iterations of the given number by only on iteration  statements with proper comments on each li
num = int(input("Enter a number: ")) #input from user
factorial = 1 #initialize factorial variable
i = 2 #initialize counter variable
count=0 #initialize count variable
while i <= num: #loop until counter is less than or equal to num
    factorial *= i #multiply factorial by counter
    i += 1 #increment counter
    count+=1 #increment count

print(f"Number of iteration : {count}") #output
print(f"The factorial of {num} is {factorial}") #output
```

```
#write a program to print factorial of the given number by only on
iteratio=ve statements with proper comments on each line
num = int(input("Enter a number: ")) #input from user
factorial = 1 #initialize factorial variable
i=2 # initialize counter variable
count=0  # initialize count variable
while i <=num: #loop until counter is less than or equal to num
    factorial *= i # multiply factorial by i
     i+=1 # increment by 1
     count+=1 #increment count
print(f"Number of iterations: {count}") # print iteration count
print(f” factorial of {num} is {factorial}”) #output
```

outputs

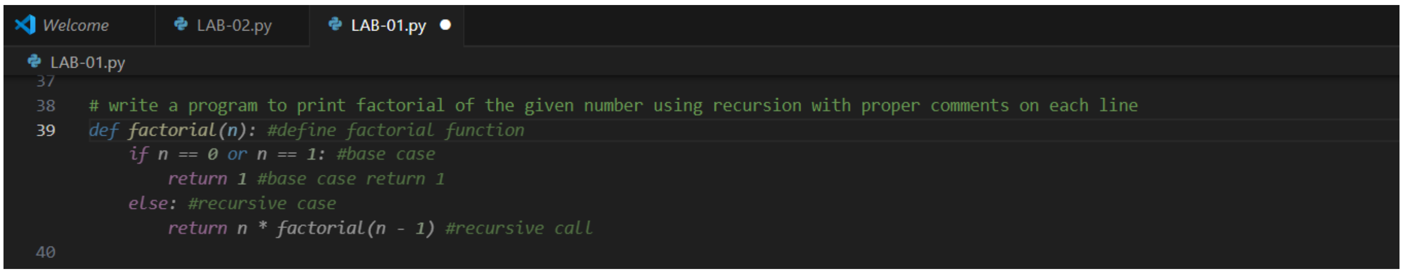


```
PS C:\Users\abhir\OneDrive\Pictures\Documents\CSE\3-2\AI-AC> & C:/Users/abhir/AppData/Local/Programs/Python/Python314/python.exe c:/Users/abhir/OneDrive/Pictures/Docume
/LAB-01.py
Enter a number: 20
Number of iteration : 19
The factorial of 20 is 2432902008176640000
```

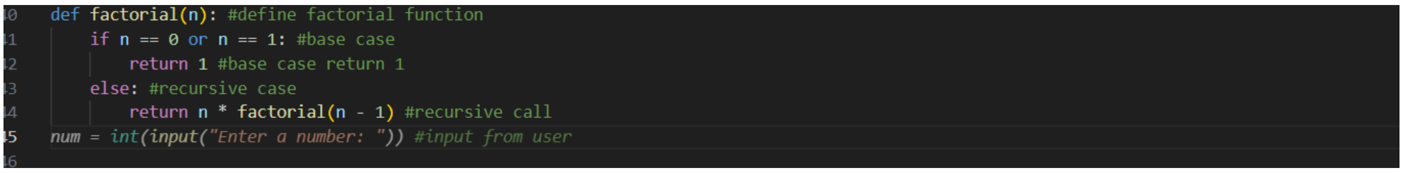


```
PS C:\Users\abhir\OneDrive\Pictures\Documents\CSE\3-2\AI-AC> & C:/Users/abhir/AppData/Local/Programs/Python/Python314/python.exe c:/Users/abhir/OneDri
ve/Pictures/Documents/CSE/3-2/AI-AC/LAB-01.py
Enter a number: 100
Number of iteration : 99
The factorial of 100 is 9332621544394415268169923885626670049071596826438162146859296389521759999322991560894146397615651828625369792082722375825118521
109168640000000000000000000000000
```

Using Recursion:



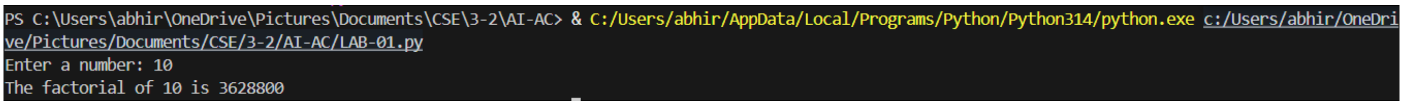
```
# write a program to print factorial of the given number using recursion with proper comments on each line
def factorial(n): #define factorial function
    if n == 0 or n == 1: #base case
        return 1 #base case return 1
    else: #recursive case
        return n * factorial(n - 1) #recursive call
```



```
def factorial(n): #define factorial function
    if n == 0 or n == 1: #base case
        return 1 #base case return 1
    else: #recursive case
        return n * factorial(n - 1) #recursive call
num = int(input("Enter a number: ")) #input from user
```

```
#write a program to print factorail of the given number by using recursion
with proper comments on each line
def factorial(n): # define factorial function
    if n == 0 or n == 1: # base case for recursion
        return 1
    else:
        return n * factorial(n – 1) # recursive case
number = int(input("Enter a Number ")) # take input from user
result=factorial(num) # function call
print(f"The factorial of {num} is {result}”) #output
```

output:



```
PS C:\Users\abhir\OneDrive\Pictures\Documents\CSE\3-2\AI-AC> & C:/Users/abhir/AppData/Local/Programs/Python/Python314/python.exe c:/Users/abhir/OneDri
ve/Pictures/Documents/CSE/3-2/AI-AC/LAB-01.py
Enter a number: 10
The factorial of 10 is 3628800
```

Comparison: Iterative vs Recursive Approach

Criteria	**Iterative Approach**	**Recursive Approach**
Readability	Easy to understand for beginners, logic is straightforward.	Code is shorter and elegant but may be confusing for beginners.
Stack Usage	Does not use call stack, memory usage is minimal.	Uses call stack for every function call, increases memory usage.
Performance Implications	Faster and more efficient for large inputs.	Slower due to repeated function calls and stack overhead.
When Recursion Is Not Recommended	Not applicable (safe for large inputs).	Not recommended for large inputs due to stack overflow risk and recursion limits.