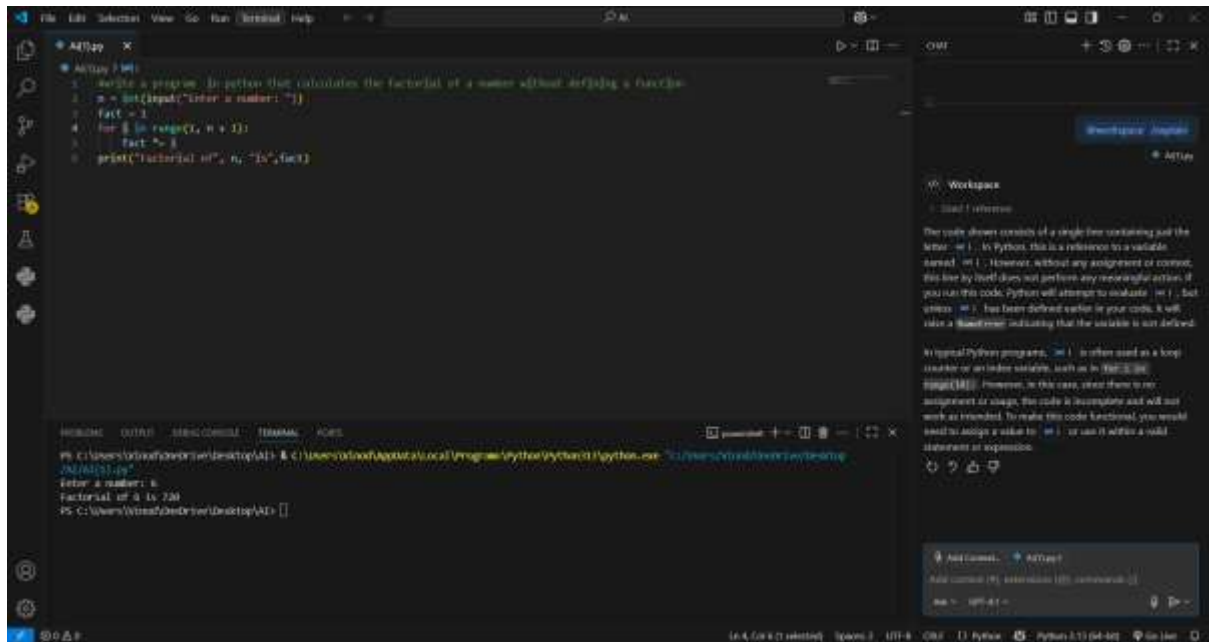


Assignment -1

Name: K.Vinod

2403A52132

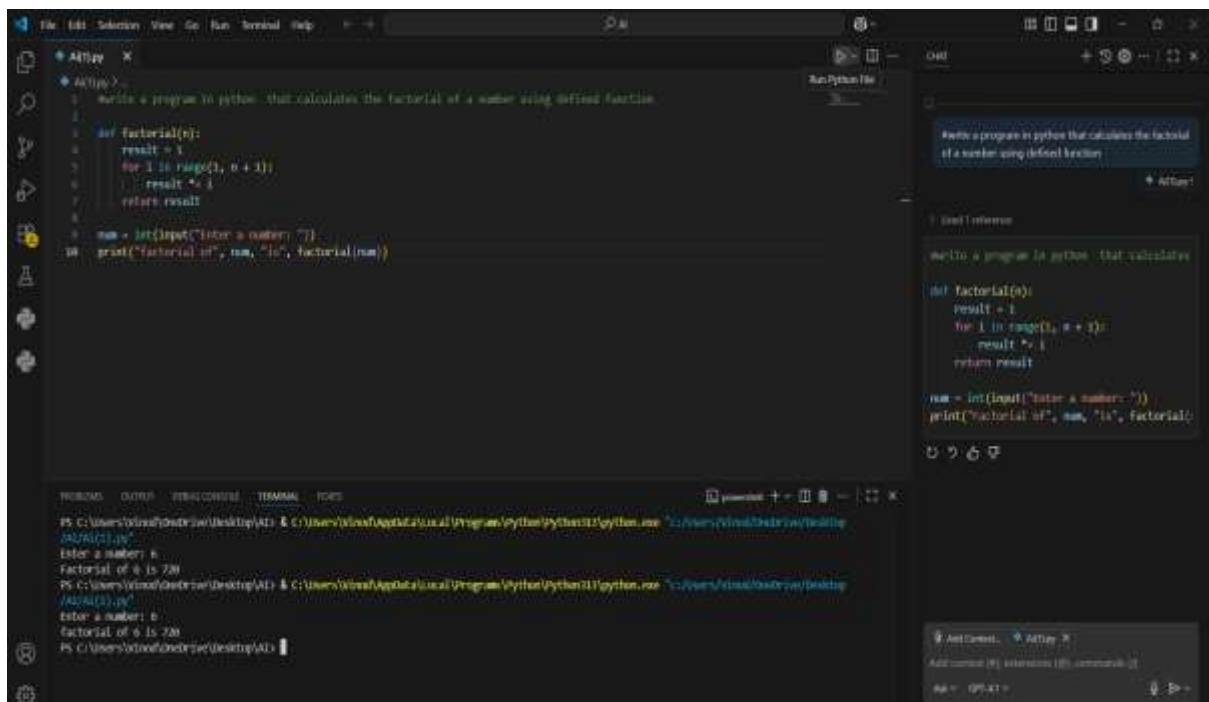
Task 1:



```
File Edit Selection View Go Run Terminal Help
JupyterLab
Python 3
Write a program in python that calculates the factorial of a number without defining a function.
1 n = int(input("Enter a number: "))
2 fact = 1
3 for i in range(1, n + 1):
4     fact *= i
5 print("Factorial of", n, "is", fact)

REPL PREVIEW TERMINAL FILES
PS C:\Users\Vinod\OneDrive\Desktop> python3 /C:/Users/Vinod/AppData/Local/Programs/Python/Python311/python.exe /C:/Users/Vinod/OneDrive/Desktop/A2/A2(1).py
Enter a number: 6
Factorial of 6 is 720
PS C:\Users\Vinod\OneDrive\Desktop>
```

Task 3:



```
File Edit Selection View Go Run Terminal Help
JupyterLab
Python 3
Write a program in python that calculates the factorial of a number using defined function.
1 def factorial(n):
2     result = 1
3     for i in range(1, n + 1):
4         result *= i
5     return result
6
7 num = int(input("Enter a number: "))
8 print("Factorial of", num, "is", factorial(num))

REPL PREVIEW TERMINAL FILES
PS C:\Users\Vinod\OneDrive\Desktop> python3 /C:/Users/Vinod/AppData/Local/Programs/Python/Python311/python.exe /C:/Users/Vinod/OneDrive/Desktop/A2/A2(3).py
Enter a number: 6
Factorial of 6 is 720
PS C:\Users\Vinod\OneDrive\Desktop> python3 /C:/Users/Vinod/AppData/Local/Programs/Python/Python311/python.exe /C:/Users/Vinod/OneDrive/Desktop/A2/A2(3).py
Enter a number: 6
factorial of 6 is 720
PS C:\Users\Vinod\OneDrive\Desktop>
```

Task 4:

#differentiate between the copilot generated factorial function with functions and without functions in the term of logic ,reusability and execution

AI(1).py:1

> Used 1 reference

Difference between factorial with function and without function:

1. Logic

- **With Function:** The logic is encapsulated inside a function (`factorial_with_function`). You pass a value, and it returns the factorial.
- **Without Function:** The logic is written directly in the main code block. No encapsulation; it runs for a specific value only.

2. Reusability

- **With Function:** Highly reusable. You can call the function with any value, multiple times, or use it in other programs.
- **Without Function:** Not reusable. You must rewrite or copy the code each time you need to calculate a factorial for a different value.

3. Execution

- **With Function:** The function is defined once and can be executed many times by calling it with different arguments.
- **Without Function:** The code executes only once for the hardcoded value. To use it again, you must modify and rerun the code.

Summary:

Using functions improves code organization, reusability, and flexibility. Without functions, code is less maintainable and only suitable for single-

The screenshot shows a VS Code editor with a Python file named `AI(1).py`. The code defines a function `factorial_with_function(n)` and a main block that calculates the factorial of 5 using both the function and a direct loop. The terminal shows the output of the script, and the Copilot chat window provides explanations for the code.

```
1 #differentiate between the copilot generated factorial function with functions and without functions
2 def factorial_with_function(n):
3     result = 1
4     for i in range(1, n + 1):
5         result *= i
6     return result
7
8 num = 5
9 print("Factorial of", num, "is", factorial_with_function(num))
10
11 # without function
12 num = 5
13 factorial = 1
14 for i in range(1, num + 1):
15     factorial *= i
16 print("Factorial of", num, "is", factorial)
```

Terminal Output:

```
PS C:\Users\Vinod\OneDrive\Desktop\AI> & C:\Users\Vinod\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\Vinod\OneDrive\Desktop\AI\AI(1).py"
Enter a number: 5
Factorial of 5 is 120
PS C:\Users\Vinod\OneDrive\Desktop\AI> & C:\Users\Vinod\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\Vinod\OneDrive\Desktop\AI\AI(1).py"
Enter a number: 5
Factorial of 5 is 120
PS C:\Users\Vinod\OneDrive\Desktop\AI>
```

Copilot Chat:

The code consists of a single line containing just the variable `num = 5`. In Python, writing `num` by itself is an expression that evaluates to the value of the variable `num`. However, if `num` has not been previously defined or assigned a value in the current scope, running this code will result in a `NameError` because Python does not know what `num` refers to.

This line does not perform any computation, output, or side effect; it simply asks Python to evaluate and return the value of `num`. In interactive environments like the Python REPL, entering `num` would display its value if it exists. In a script or file, unless `num` is assigned earlier, this line is not useful and will likely cause an error. Typically, `num` is used as a loop variable or index, but on its own, it does not accomplish anything meaningful.

Task 5:

```
1 # Write a program in python to generate the iterative and recursive of the factorial function
2
3 # Iterative factorial function
4 def factorial_iterative(n):
5     result = 1
6     for i in range(1, n + 1):
7         result *= i
8     return result
9
10 # Recursive factorial function
11 def factorial_recursive(n):
12     if n == 0 or n == 1:
13         return 1
14     else:
15         return n * factorial_recursive(n - 1)
16
17 num = int(input("Enter a number: "))
18 print("Iterative factorial of", num, "is", factorial_iterative(num))
19 print("Recursive factorial of", num, "is", factorial_recursive(num))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Vinod\OneDrive\Desktop\AI1> & C:\Users\Vinod\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\Vinod\OneDrive\Desktop\AI1\AI1.py"
Enter a number: 6
Iterative factorial of 6 is 720
Recursive factorial of 6 is 720
PS C:\Users\Vinod\OneDrive\Desktop\AI1>
```

Workspace

Used 1 reference

This Python program demonstrates two different approaches to calculating the factorial of a number: iterative and recursive. The factorial of a non-negative integer n (denoted as $n!$) is the product of all positive integers less than or equal to n . For example, $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$.

The first function, `factorial_iterative()`, uses an iterative approach. It initializes a variable `result` to 1, then uses a `for` loop to multiply `result` by each integer from 1 up to and including `n`. This method is straightforward and efficient, especially for larger values of `n`, as it avoids the overhead of recursive function calls.

The second function, `factorial_recursive()`, uses recursion. It checks if `n` is 0 or 1, in which case it returns 1 (since both 0! and 1! are defined as 1). Otherwise, it returns `n` multiplied by the factorial of `n - 1`, effectively breaking down the problem into smaller subproblems until the base case is reached. While elegant and closer to the mathematical definition, recursion can be less efficient for large `n` due to call stack limitations.

At the end of the program, the user is prompted to enter a number. The program then prints the factorial of that number calculated both iteratively and recursively. This allows users to see that both methods produce the same result, illustrating two fundamental programming techniques for solving problems.

Add Content... AI1.py X

Add content (H), references (R), notebooks (J)

Ask - GPT-4.1