

Name: Narige Rohith
Enroll no : 2403A54101

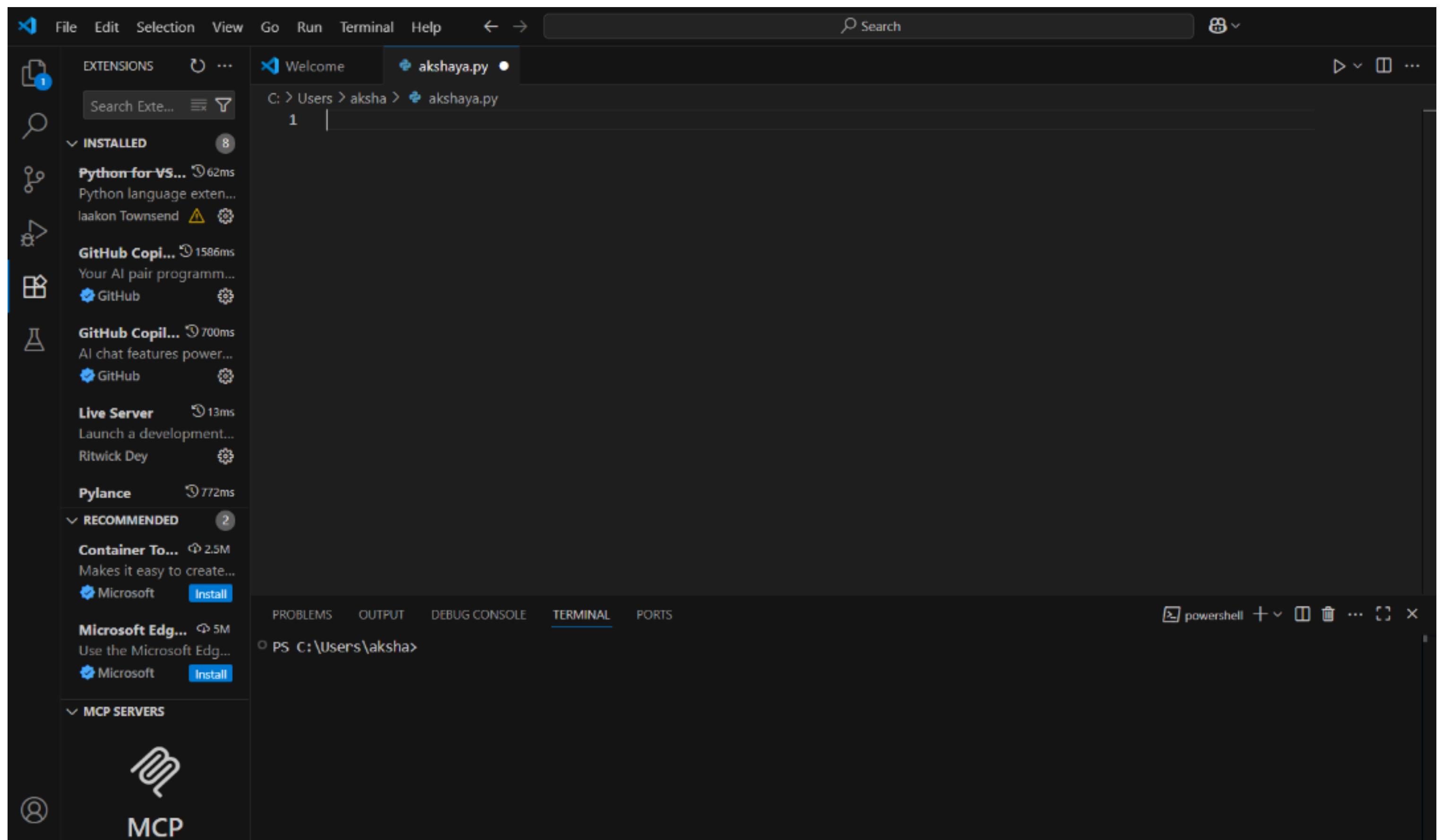
Batch : (DS)batch-4

Task 0

- Install and configure GitHub Copilot in VS Code. Take screenshots of each step.

Expected Output

- Install and configure GitHub Copilot in VS Code. Take screenshots of each step.



Task 1: Factorial without Functions

- Description:
Use GitHub Copilot to generate a Python program that calculates the factorial of a number without defining any functions (using loops directly in the main code).
- Expected Output:
 - A working program that correctly calculates the factorial for user-provided input.
 - Screenshots of the code generation process.

prompt : generate a python program which takes a number from the user and give the factorial as the output ,without using any functions

The screenshot shows the VS Code interface with the following details:

- Editor:** The file `akshaya.py` is open, displaying the following Python code:

```
C:\> Users\aksha> akshaya.py ...
1 num = int(input("Enter a number: "))
2 factorial = 1
3
4 for i in range(1, num + 1):
5     factorial *= i
6
7 print(f"Factorial of {num} is{factorial}")
```
- Code Generation Panel:** A floating panel on the right side provides instructions for GitHub Copilot:
 - correctly calculates the factorial
 - for user-provided input
 - Screenshots of the code generation process.
- Terminal:** The terminal shows the execution of the script and its output:

```
PS C:\Users\aksha> & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher' '63883' '--' 'c:\Users\aksha\akshaya.py'
● Enter a number: 2
Factorial of 2 is2
○ PS C:\Users\aksha>
```
- Bottom Right:** A small sidebar with options like "Add Context...", "Current File", "Ask Copilot", "Ask", "GPT-4.1", and "Copilot".

Task 2: Improving Efficiency

- Description:
Examine the Copilot-generated code from Task 1 and demonstrate how its efficiency can be improved (e.g., removing unnecessary variables, optimizing loops).
- Expected Output:
 - Original and improved versions of the code.
 - Explanation of how the improvements enhance performance.

prompt : improve the efficiency from the previous code by removing unnecessary variables, optimizing loops etc...

The screenshot shows the Visual Studio Code interface. On the left, the 'akshaya.py' file is open in the editor, displaying the following code:

```

1 num = int(input("Enter a number: "))
2 result = 1
3
4 for i in range(2, num + 1):
5     result *= i
6
7 print(f"Factorial of {num} is {result}")

```

In the center, the Python Debug Console shows the execution of the script:

```

PS C:\Users\aksha> & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\.vscode\extensions\ms-python.python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63883' '--' 'c:\Users\aksha\akshaya.py'
● Enter a number: 2
Factorial of 2 is 2
● PS C:\Users\aksha> c;; cd 'c:\Users\aksha'; & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\.vscode\extensions\ms-python.python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63916' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 5
Factorial of 5 is 120
○ PS C:\Users\aksha>

```

To the right, there is a panel titled 'Copilot Suggestion' with the following content:

Original Code:

```

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

print(f"Factorial of {num}

```

Improved Code:

```

for i in range(2, num + 1)
    factorial *= i

print(f"Factorial of {num} is {result}")

```

Explanation:

- The loop now starts from 2 instead of 1, since multiplying by 1 is redundant.
- The variable name `result` is more descriptive.
- This reduces one unnecessary multiplication per run, slightly improving efficiency, especially for large numbers.

At the bottom right of the interface, there is a 'Ask Copilot' button.

Task 3: Factorial with Functions

- Description:
Use GitHub Copilot to generate a Python program that calculates the factorial of a number using a user-defined function.
- Expected Output:
 - Correctly working factorial function with sample outputs.
 - Documentation of the steps Copilot followed to generate the function.

Prompt :generate a python program which take the input from the user and give factorial of it by taking an user defined function

The screenshot shows the VS Code interface with a Python file named 'akshaya.py' open. The code defines a function 'factorial(n)' that calculates the factorial of a number using a loop. It then prompts the user for input and prints the result. A Copilot-generated documentation box is visible, detailing the steps taken by the AI. Below the editor, the terminal window shows three executions of the program, demonstrating its functionality with inputs 5 and 45.

```
C:\> Users > aksha > akshaya.py ...
1 def factorial(n):
2     result = 1
3     for i in range(2, n + 1):
4         result *= i
5     return result
6
7 num = int(input("Enter a number: "))
8 print(f"Factorial of {num} is {factorial(num)}")

def factorial(n):
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

num = int(input("Enter a n
print(F"Factorial of {num}

Documentation of Copilot Steps:
1. Defined a function
factorial(n) that calculates
the factorial using a loop.
2. Initialized result to 1 and
multiplied it by each integer
from 2 to n.
3. Returned the computed result.
4. Took user input, called the
function, and printed the output.

PS C:\Users\aksha> c;; cd 'c:\Users\aksha'; & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63916' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 5
Factorial of 5 is 120
PS C:\Users\aksha> c;; cd 'c:\Users\aksha'; & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63918' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 45
Factorial of 45 is 11962222886548019456196316149565771586438373176000000000000
PS C:\Users\aksha> c;; cd 'c:\Users\aksha'; & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63953' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 5
Factorial of 5 is 120
PS C:\Users\aksha>
```

Task 4: Comparative Analysis – With vs Without Functions

- Description:

Differentiate between the Copilot-generated factorial program with functions and without functions in terms of logic, reusability, and execution.

- Expected Output:

A comparison table or short report explaining the differences

Prompt:

- Differentiate between program with functions and without functions in terms of logic, reusability, and execution.

The screenshot shows a code editor interface with a sidebar containing icons for file operations like Open, Save, Find, and Run. The main window title is "CHW" and the status bar shows "assignment-1-1.py". The code content is as follows:

```
Here's a comparison between the factorial program with and without functions:

Logic:
• Without functions: The logic is written directly in the main code block. The loop and calculation are performed inline, making the code straightforward but less organized.
• With functions: The logic is encapsulated in a function (factorial(n)), separating the calculation from input/output. This improves code structure and clarity.

Reusability:
• Without functions: The factorial calculation cannot be reused elsewhere in the program without copying the code.
• With functions: The Factorial function can be called multiple times with different arguments, making it reusable in other parts of the program or in other programs.

Execution:
• Without functions: The code executes sequentially from top to bottom. Only one factorial calculation is possible per run.
• With functions: The function can be called as needed, allowing for multiple calculations or integration into larger programs. It also makes testing and debugging easier.

In summary, using functions improves organization, reusability, and flexibility, while direct code is simpler but less versatile.
```

The bottom status bar includes icons for GitHub, Python, and Go Live, along with the current file path "assignment-1-1.py" and line/column information "Ln 0, Col 55".

Task 5: Iterative vs Recursive Factorial

- Description:
Prompt GitHub Copilot to generate both iterative and recursive versions of the factorial function.
- Expected Output:
 - Two correct implementations.
 - A documented comparison of logic, performance, and execution flow between iterative and recursive approaches.

- Prompt : generate both iterative and recursive versions of the factorial function

```

C:\> Users > aksha > akshaya.py ...
1 # Iterative factorial function
2 def factorial_iterative(n):
3     result = 1
4     for i in range(2, n + 1):
5         result *= i
6     return result
7
8 # Recursive factorial function
9 def factorial_recursive(n):
10    if n == 0 or n == 1:
11        return 1
12    else:
13        return n * factorial_recursive(n - 1)
14
15 num = int(input("Enter a number: "))
16 print(f"Iterative: Factorial of {num} is {factorial_iterative(num)}")
17 print(f"Recursive: Factorial of {num} is {factorial_recursive(num)}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python Debug Console

```

-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63948' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 45
Factorial of 45 is 119622220865488019456196316149565771506438373376000000000000
● PS C:\Users\aksha> cd 'c:\Users\aksha'; & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\.vscode\extensions\ms-python.python-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '63953' '--' 'C:\Users\aksha\akshaya.py'
Enter a number: 5
Factorial of 5 is 120
● PS C:\Users\aksha> cd 'c:\Users\aksha'; & 'c:\Users\aksha\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\aksha\.vscode\extensions\ms-python.python-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '64004' '--' 'c:\Users\aksha\akshaya.py'
Enter a number: 4
Iterative: Factorial of 4 is 24
Recursive: Factorial of 4 is 24
○ PS C:\Users\aksha>

```

In 17, Col 70 Spaces: 4 Diff: 0 CRLF: 0 Python: 0m 3.13s 98% (60.18%)

Two correct implementations.
A documented comparison of logic, performance, and execution flow between iterative and recursive approaches.

Used 1 reference

Iterative factorial func
def factorial_iterative(n)
 result = 1
 for i in range(2, n +
 result *= i
 return result

Recursive factorial func
def factorial_recursive(n)
 if n == 0 or n == 1:
 return 1

Snipping Tool

Screenshot copied to clipboard
Automatically saved to screenshots folder.

Mark-up and share