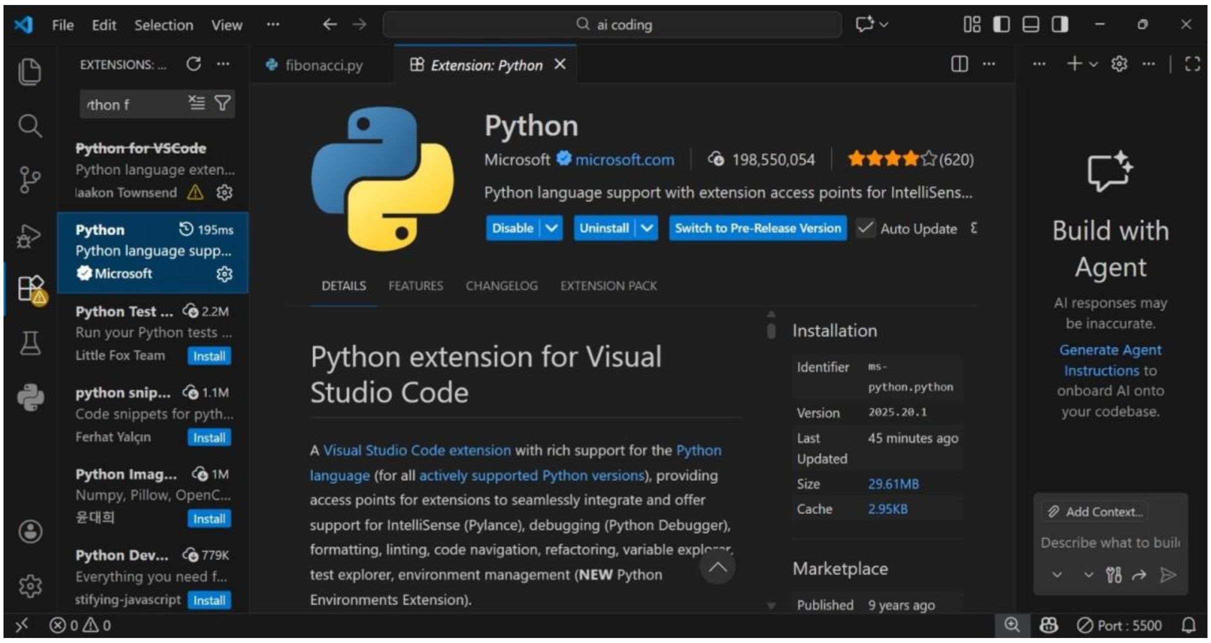
Course Title: AI-Assisted Coding

Batch – 06

Hall no. – 2303A510C3

Question: Lab 1: Environment Setup – GitHub Copilot and VS Code Integration + Understanding AI-assisted Coding Workflow

Task 0 ● Install and configure GitHub Copilot in VS Code. Take screenshots of each step.



Python for VSCode
Python language exten...
Iaakon Townsend

Python 195ms
Python language supp...
Microsoft

Python Test ... 2.2M
Run your Python tests ...
Little Fox Team Install

python snip... 1.1M
Code snippets for pyth...
Ferhat Yalçın Install

Python Imag... 1M
Numpy, Pillow, OpenC...
윤대희 Install

Python Dev... 779K
Everything you need f...
stifying-javascript Install

Python
Microsoft microsoft.com 198,550,054 ★★★★☆(620)
Python language support with extension access points for IntelliSens...

Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the Python language (for all actively supported Python versions), providing access points for extensions to seamlessly integrate and offer support for IntelliSense (Pylance), debugging (Python Debugger), formatting, linting, code navigation, refactoring, variable explorer, test explorer, environment management (**NEW** Python Environments Extension).

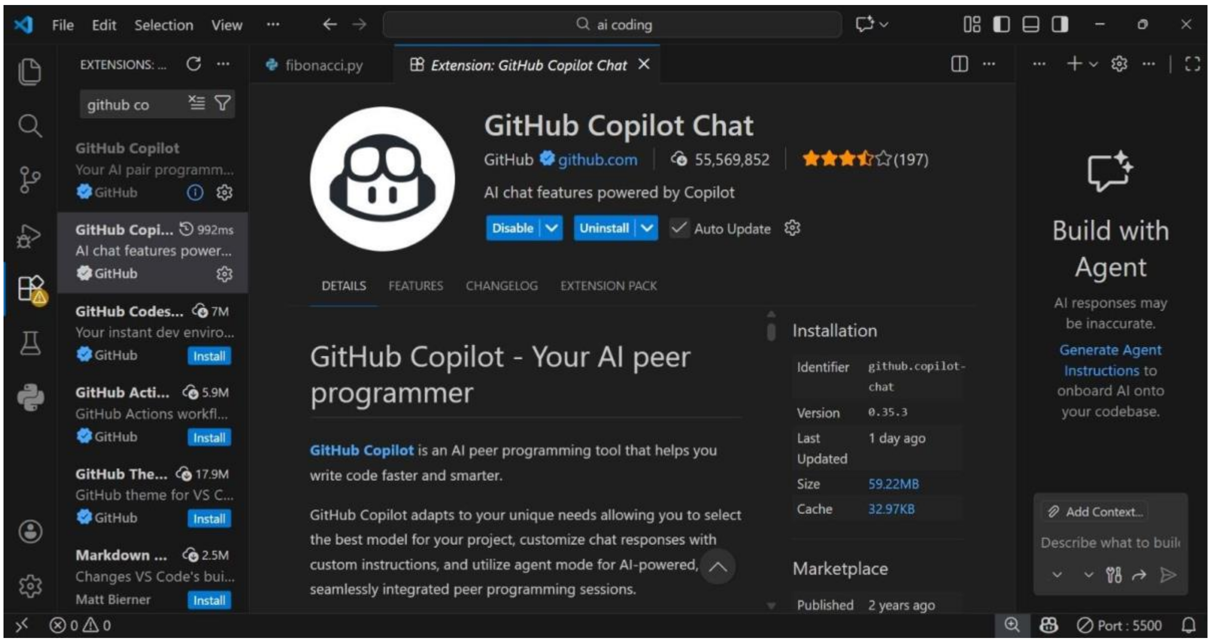
Installation

Identifier	ms-python.python
Version	2025.20.1
Last Updated	45 minutes ago
Size	29.61MB
Cache	2.95KB

Marketplace

Published	9 years ago

AI responses may be inaccurate.



github co

GitHub Copilot
Your AI pair programm...
GitHub

GitHub Copi... 992ms
AI chat features power...
GitHub

GitHub Codes... 7M
Your instant dev enviro...
GitHub Install

GitHub Acti... 5.9M
GitHub Actions workfl...
GitHub Install

GitHub The... 17.9M
GitHub theme for VS C...
GitHub Install

Markdown ... 2.5M
Changes VS Code's bui...
Matt Bierner Install

GitHub Copilot Chat
GitHub github.com 55,569,852 ★★★½☆(197)
AI chat features powered by Copilot

GitHub Copilot - Your AI peer programmer

GitHub Copilot is an AI peer programming tool that helps you write code faster and smarter.

GitHub Copilot adapts to your unique needs allowing you to select the best model for your project, customize chat responses with custom instructions, and utilize agent mode for AI-powered, seamlessly integrated peer programming sessions.

Installation

Identifier	github.copilot-chat
Version	0.35.3
Last Updated	1 day ago
Size	59.22MB
Cache	32.97KB

Marketplace

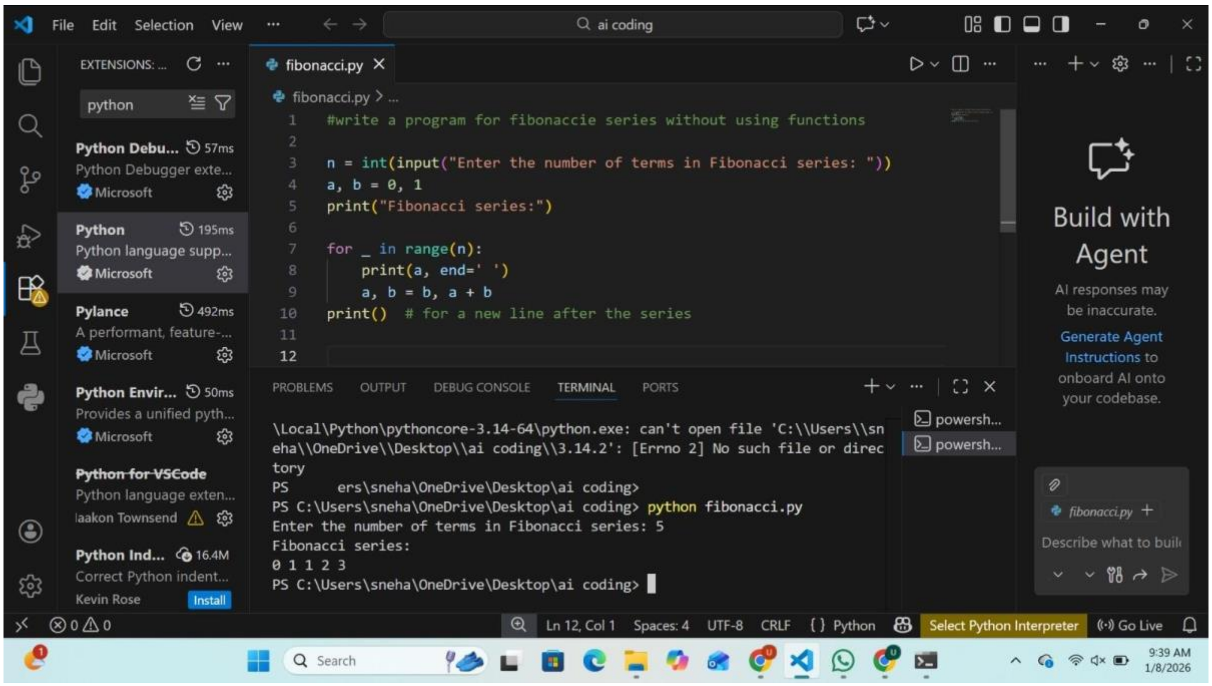
Published	2 years ago

AI responses may be inaccurate.

Explanation: I installed GitHub Copilot in VS Code using the Extensions option. Then I signed in with my GitHub account and allowed permissions. Copilot started giving code suggestions while typing, which made coding easier.

Task 1: AI-Generated Logic Without Modularisation (Fibonacci Sequence Without Functions)

Input :



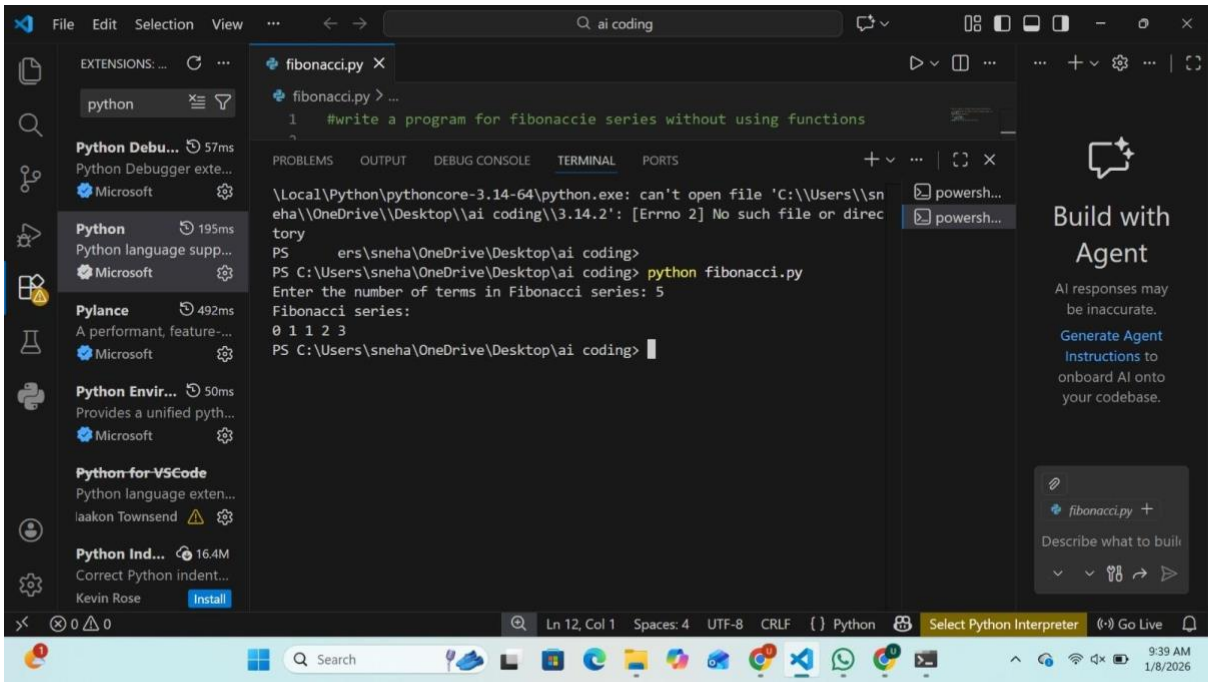
```python
#write a program for fibonaccie series without using functions

n = int(input("Enter the number of terms in Fibonacci series: "))
a, b = 0, 1
print("Fibonacci series:")

for _ in range(n):
    print(a, end=' ')
    a, b = b, a + b
print()  # for a new line after the series
```

```
\Local\Python\pythoncore-3.14-64\python.exe: can't open file 'C:\\Users\\sn
eha\\OneDrive\\Desktop\\ai coding\\3.14.2': [Errno 2] No such file or direc
tory
PS      ers\sneha\OneDrive\Desktop\ai coding>
PS C:\Users\sneha\OneDrive\Desktop\ai coding> python fibonacci.py
Enter the number of terms in Fibonacci series: 5
Fibonacci series:
0 1 1 2 3
PS C:\Users\sneha\OneDrive\Desktop\ai coding>
```

Output :



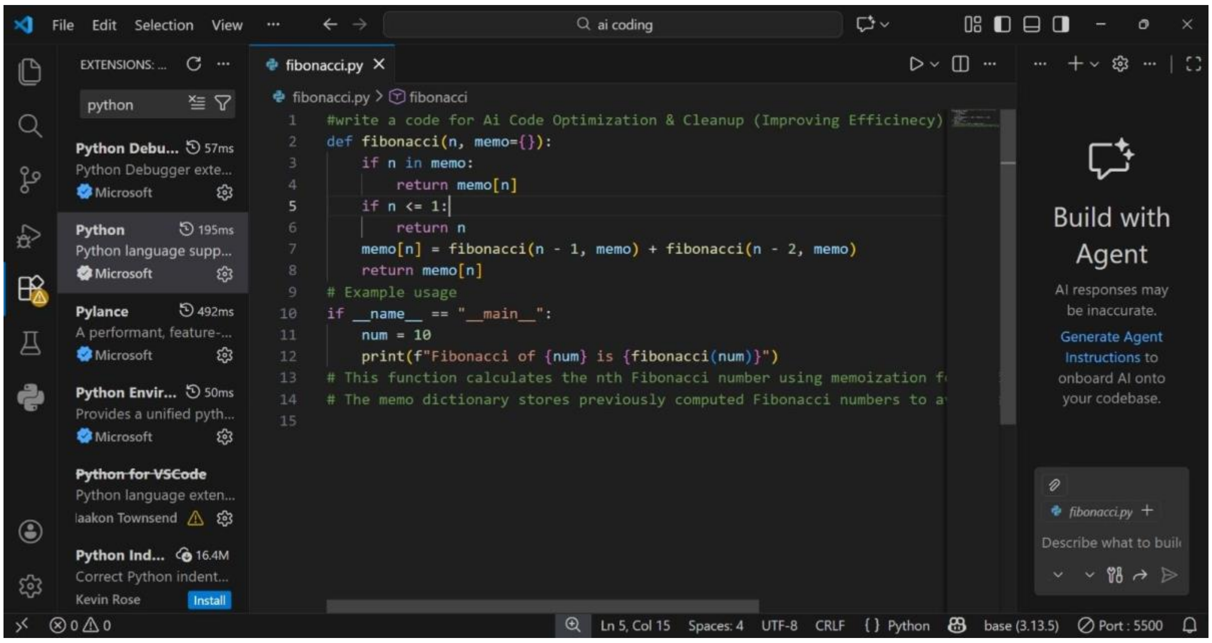
```python
#write a program for fibonaccie series without using functions
```

```
\Local\Python\pythoncore-3.14-64\python.exe: can't open file 'C:\\Users\\sn
eha\\OneDrive\\Desktop\\ai coding\\3.14.2': [Errno 2] No such file or direc
tory
PS      ers\sneha\OneDrive\Desktop\ai coding>
PS C:\Users\sneha\OneDrive\Desktop\ai coding> python fibonacci.py
Enter the number of terms in Fibonacci series: 5
Fibonacci series:
0 1 1 2 3
PS C:\Users\sneha\OneDrive\Desktop\ai coding>
```

Explanation: The Fibonacci code is written in one place. No functions are used in this program. The code works, but it looks messy.

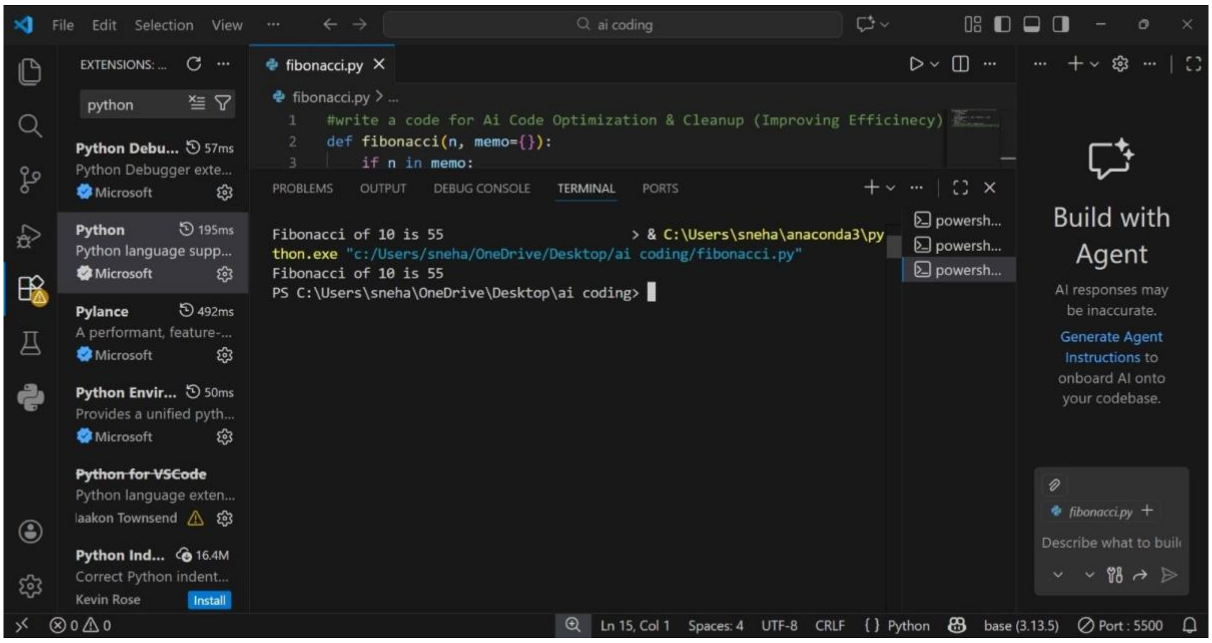
Task 2: AI-Code Optimisation & Cleanup (Improving Efficiency)

Input :



```python
#write a code for Ai Code Optimization & Cleanup (Improving Efficinecy)
def fibonacci(n, memo={}):
    if n in memo:
        return memo[n]
    if n <= 1:
        return n
    memo[n] = fibonacci(n - 1, memo) + fibonacci(n - 2, memo)
    return memo[n]
# Example usage
if __name__ == "__main__":
    num = 10
    print(f"Fibonacci of {num} is {fibonacci(num)}")
# This function calculates the nth Fibonacci number using memoization f
# The memo dictionary stores previously computed Fibonacci numbers to a
```

Output :



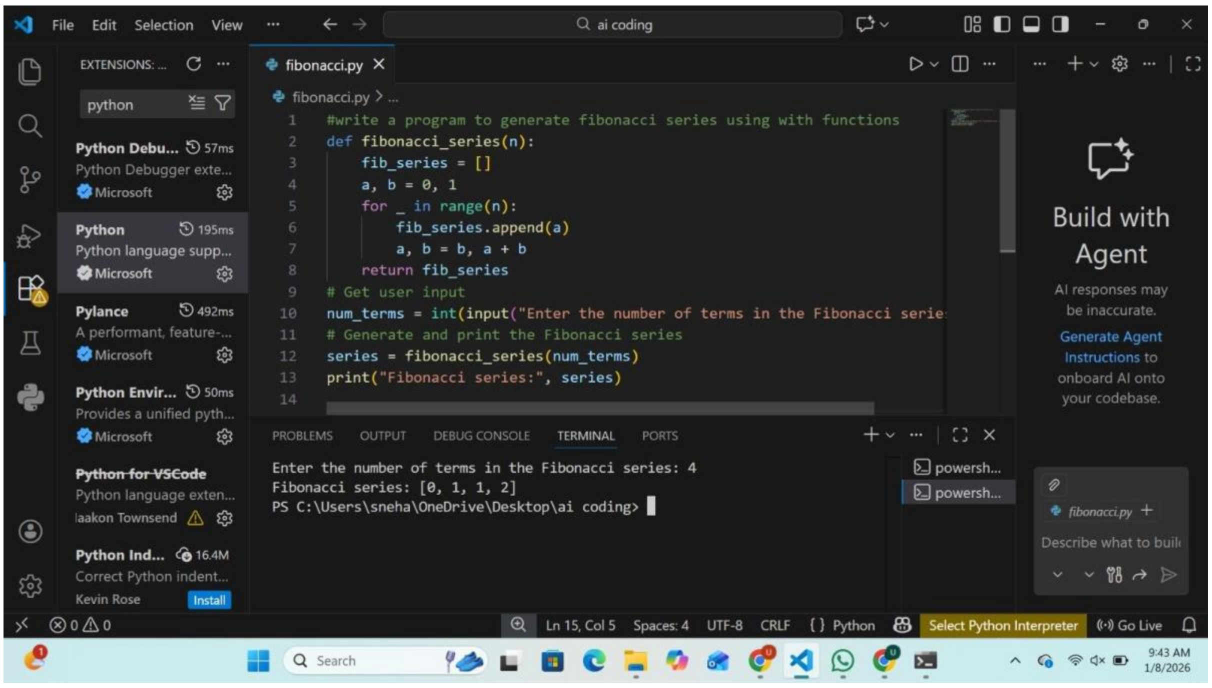
```
Fibonacci of 10 is 55                    > & C:\Users\sneha\anaconda3\py
thon.exe "c:/Users/sneha/OneDrive/Desktop/ai coding/fibonacci.py"
Fibonacci of 10 is 55
PS C:\Users\sneha\OneDrive\Desktop\ai coding>
```

Explanation : AI removed extra and useless code.The program became short and clean.

Now it is easy to understand.

Task 3: Modular Design Using AI Assistance (Fibonacci Using Functions)

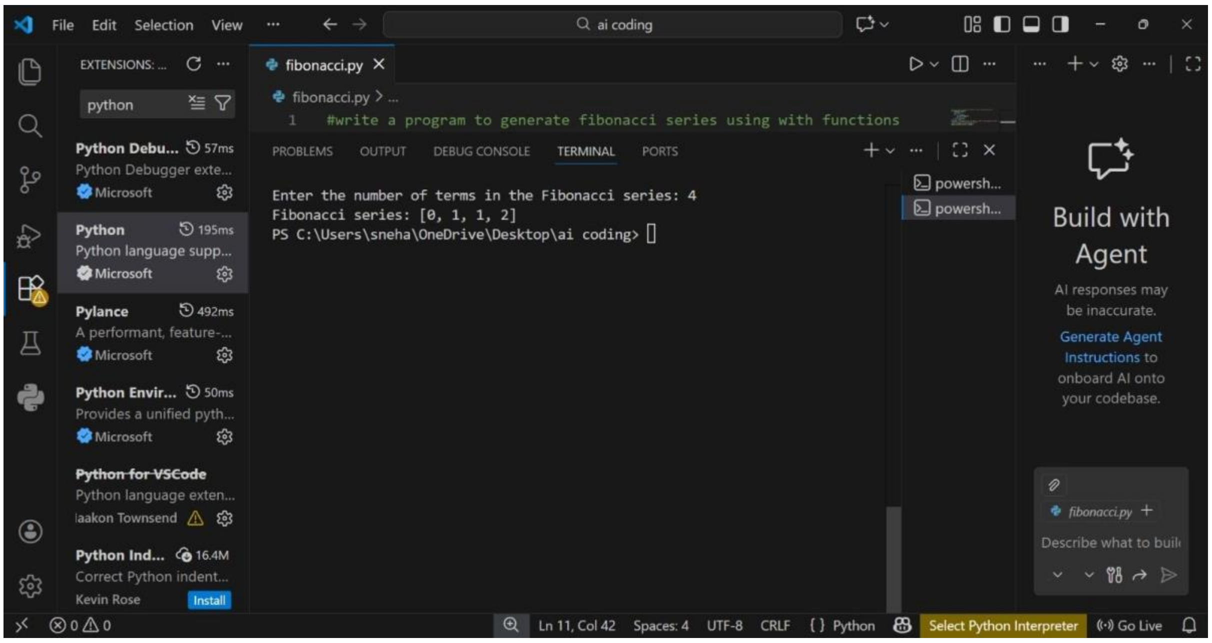
Input :


fibonacci.py
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Select Python Interpreter

```python
#write a program to generate fibonacci series using with functions
def fibonacci_series(n):
    fib_series = []
    a, b = 0, 1
    for _ in range(n):
        fib_series.append(a)
        a, b = b, a + b
    return fib_series
# Get user input
num_terms = int(input("Enter the number of terms in the Fibonacci serie
# Generate and print the Fibonacci series
series = fibonacci_series(num_terms)
print("Fibonacci series:", series)
```

```
Enter the number of terms in the Fibonacci series: 4
Fibonacci series: [0, 1, 1, 2]
PS C:\Users\sneha\OneDrive\Desktop\ai coding>
```

Output :


fibonacci.py
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Select Python Interpreter

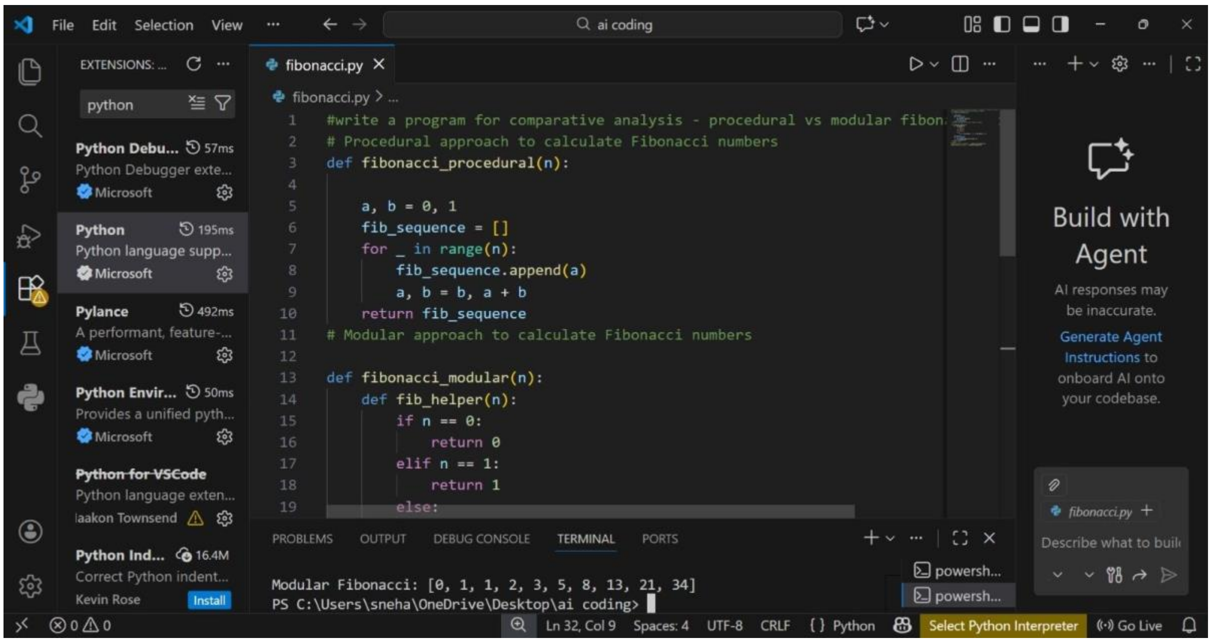
```
Enter the number of terms in the Fibonacci series: 4
Fibonacci series: [0, 1, 1, 2]
PS C:\Users\sneha\OneDrive\Desktop\ai coding>
```

Explanation : The code is written using a function.This makes the program neat.

The function can be reused.

Task 4: Comparative Analysis – Procedural vs Modular Fibonacci Code

Input :



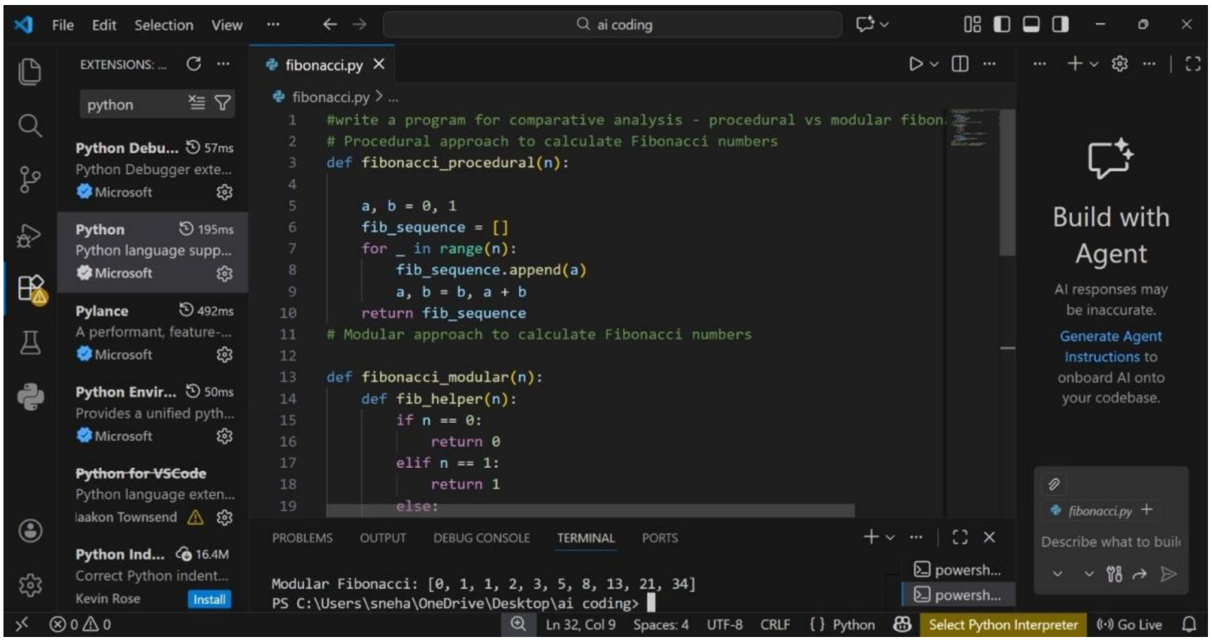
```python
#write a program for comparative analysis - procedural vs modular fibon
# Procedural approach to calculate Fibonacci numbers
def fibonacci_procedural(n):

    a, b = 0, 1
    fib_sequence = []
    for _ in range(n):
        fib_sequence.append(a)
        a, b = b, a + b
    return fib_sequence
# Modular approach to calculate Fibonacci numbers

def fibonacci_modular(n):
    def fib_helper(n):
        if n == 0:
            return 0
        elif n == 1:
            return 1
        else:
```

```
Modular Fibonacci: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
PS C:\Users\sneha\OneDrive\Desktop\ai coding>
```

fibonacci.py
EXTENSIONS
python
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Ln 32, Col 9 Spaces: 4 UTF-8 CRLF Python
Select Python Interpreter
Go Live



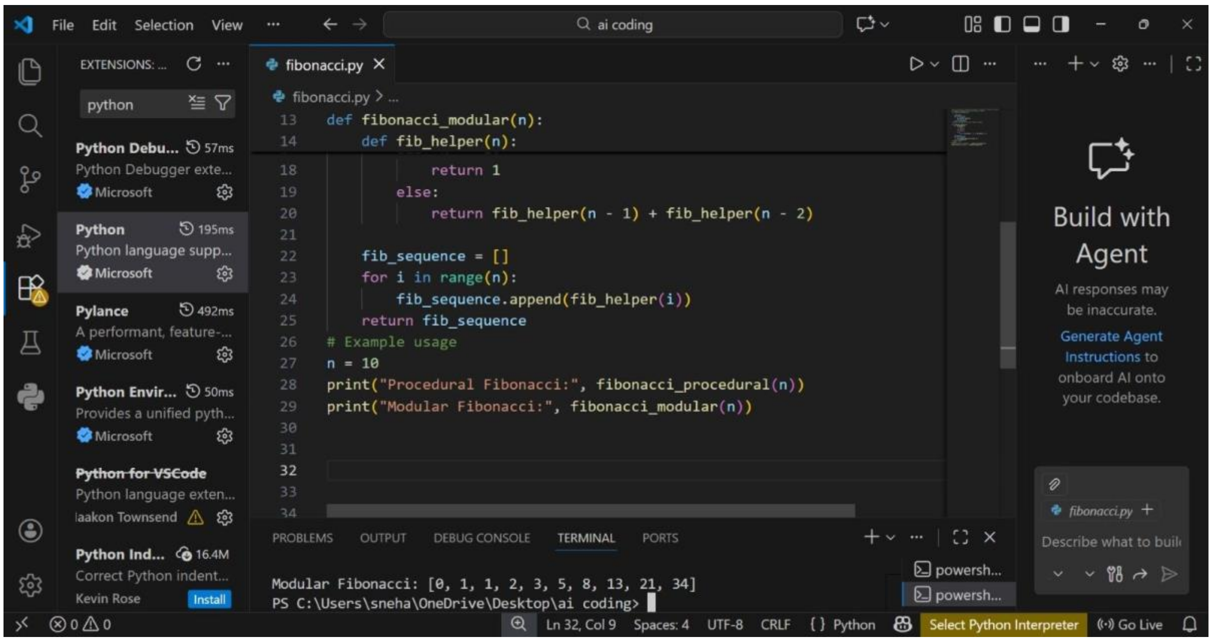
```python
#write a program for comparative analysis - procedural vs modular fibon
# Procedural approach to calculate Fibonacci numbers
def fibonacci_procedural(n):

    a, b = 0, 1
    fib_sequence = []
    for _ in range(n):
        fib_sequence.append(a)
        a, b = b, a + b
    return fib_sequence
# Modular approach to calculate Fibonacci numbers

def fibonacci_modular(n):
    def fib_helper(n):
        if n == 0:
            return 0
        elif n == 1:
            return 1
        else:
```

```
Modular Fibonacci: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
PS C:\Users\sneha\OneDrive\Desktop\ai coding>
```

Output :



```python
def fibonacci_modular(n):
    def fib_helper(n):
            return 1
        else:
            return fib_helper(n - 1) + fib_helper(n - 2)

    fib_sequence = []
    for i in range(n):
        fib_sequence.append(fib_helper(i))
    return fib_sequence
# Example usage
n = 10
print("Procedural Fibonacci:", fibonacci_procedural(n))
print("Modular Fibonacci:", fibonacci_modular(n))
```

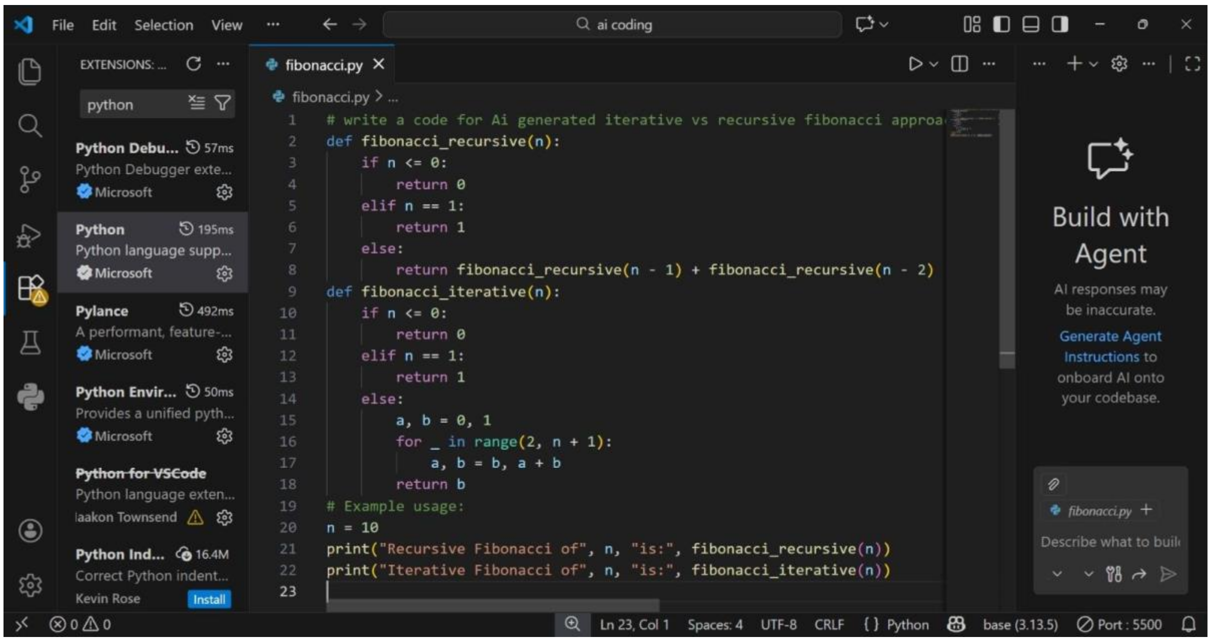
```
Modular Fibonacci: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
PS C:\Users\sneha\OneDrive\Desktop\ai coding>
```

Explantion ; Procedural code is written in one block.Modular code uses functions.

Modular code is better and clearer.

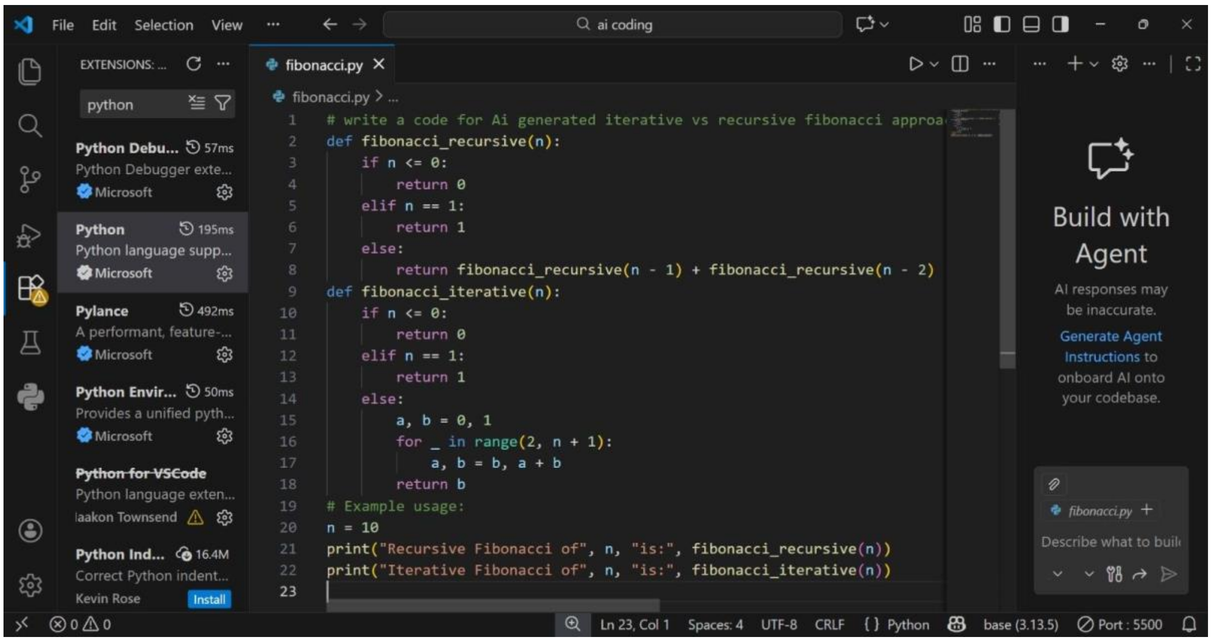
Task 5: AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches for Fibonacci Series)

Input :



```python
# write a code for Ai generated iterative vs recursive fibonacci approa
def fibonacci_recursive(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2)
def fibonacci_iterative(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        a, b = 0, 1
        for _ in range(2, n + 1):
            a, b = b, a + b
        return b
# Example usage:
n = 10
print("Recursive Fibonacci of", n, "is:", fibonacci_recursive(n))
print("Iterative Fibonacci of", n, "is:", fibonacci_iterative(n))
```

Output :



```python
# write a code for Ai generated iterative vs recursive fibonacci approa
def fibonacci_recursive(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2)
def fibonacci_iterative(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        a, b = 0, 1
        for _ in range(2, n + 1):
            a, b = b, a + b
        return b
# Example usage:
n = 10
print("Recursive Fibonacci of", n, "is:", fibonacci_recursive(n))
print("Iterative Fibonacci of", n, "is:", fibonacci_iterative(n))
```

Build with Agent

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

Explanation :

Iterative method uses a loop. Recursive method calls itself. The loop method is faster.