

Lab Assignment 1.3

A.Nikhil (2303A51934)

This screenshot shows the Visual Studio Code interface with the 'Python' extension by Microsoft selected in the Extensions Marketplace. The left sidebar lists various extensions, including Code Runner, Python Debugger, Python, Pylance, Python Environments, Python Indent, and Python for VSCode. The main panel displays the details for the 'Python' extension, which has 198,510,521 installations and a 5-star rating. It includes an 'Installation' section with details like identifier (ms-python.python), version (2025.20.1), and size (29.61MB). The 'Marketplace' section shows it was published 9 years ago and last updated 23 hours ago. The 'Categories' section is also visible. The right sidebar shows a chat window with a script for summing two numbers and a function for adding two numbers.

Python
Microsoft | microsoft.com | 198,510,521 | ★★★★★

Python language support with extension access points for Inte...

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).

Support for [vscode.dev](#)

The Python extension does offer some support when running on [vscode.dev](#) (which includes [authdev](#)). This

Installation

| Property | Value |
|--------------|------------------|
| Identifier | ms-python.python |
| Version | 2025.20.1 |
| Last Updated | 2 weeks ago |
| Size | 29.61MB |
| Cache | 5.24KB |

Marketplace

| Property | Value |
|---------------|--------------|
| Published | 9 years ago |
| Last Released | 23 hours ago |

Categories

1) Script (reads two numbers from the user)

```
# sum_script.py
a = float(input("Enter first numl
b = float(input("Enter second nur
print("Sum:", a + b)
```

2) Function (reusable)

```
def add(a, b):
    return a + b

# usage
print(add(3, 4)) # prints 7
```

Note: use `int()` instead of `float()` if you only want integers.

This screenshot shows the Visual Studio Code interface with the 'GitHub Copilot' extension by GitHub selected in the Extensions Marketplace. The left sidebar lists various extensions, including GitLens, GitHub Copilot, GitHub Copilot Chat, GitHub Pull Requests, Git Graph, Git History, Git Blame, and Git Extension Pack. The main panel displays the details for the 'GitHub Copilot' extension, which has 66,427,573 installations and a 5-star rating. It includes an 'Installation' section with details like identifier (github.copilot), version (1.388.0), and size (73.27MB). The 'Marketplace' section shows it was published 4 years ago and last updated 2 months ago. The 'Categories' section is also visible. The right sidebar shows a chat window with a script for summing two numbers and a function for adding two numbers.

GitHub Copilot
GitHub | github.com | 66,427,573 | ★★★★★ (1035)

Your AI pair programmer

Enable (Workspace) | Uninstall | Switch to Pre-Release Version

All GitHub Copilot functionality is now being served from the GitHub Copilot Chat extension. To temporarily opt out of this extension unification, toggle the `chat.extensionUnification.enabled` setting.

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.

Sign up for [GitHub Copilot Free!](#)

Installation

| Property | Value |
|--------------|----------------|
| Identifier | github.copilot |
| Version | 1.388.0 |
| Last Updated | 37 minutes ago |
| Size | 73.27MB |

Marketplace

| Property | Value |
|---------------|--------------|
| Published | 4 years ago |
| Last Released | 2 months ago |

1) Script (reads two numbers from the user)

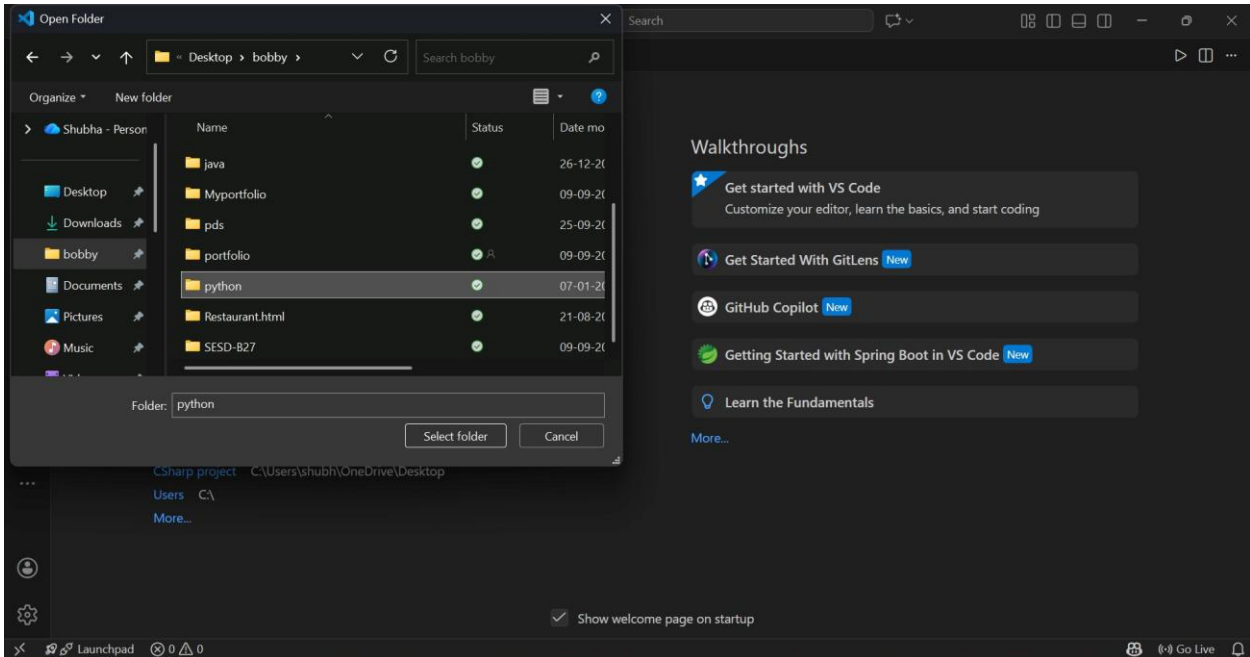
```
# sum_script.py
a = float(input("Enter first numl
b = float(input("Enter second nur
print("Sum:", a + b)
```

2) Function (reusable)

```
def add(a, b):
    return a + b

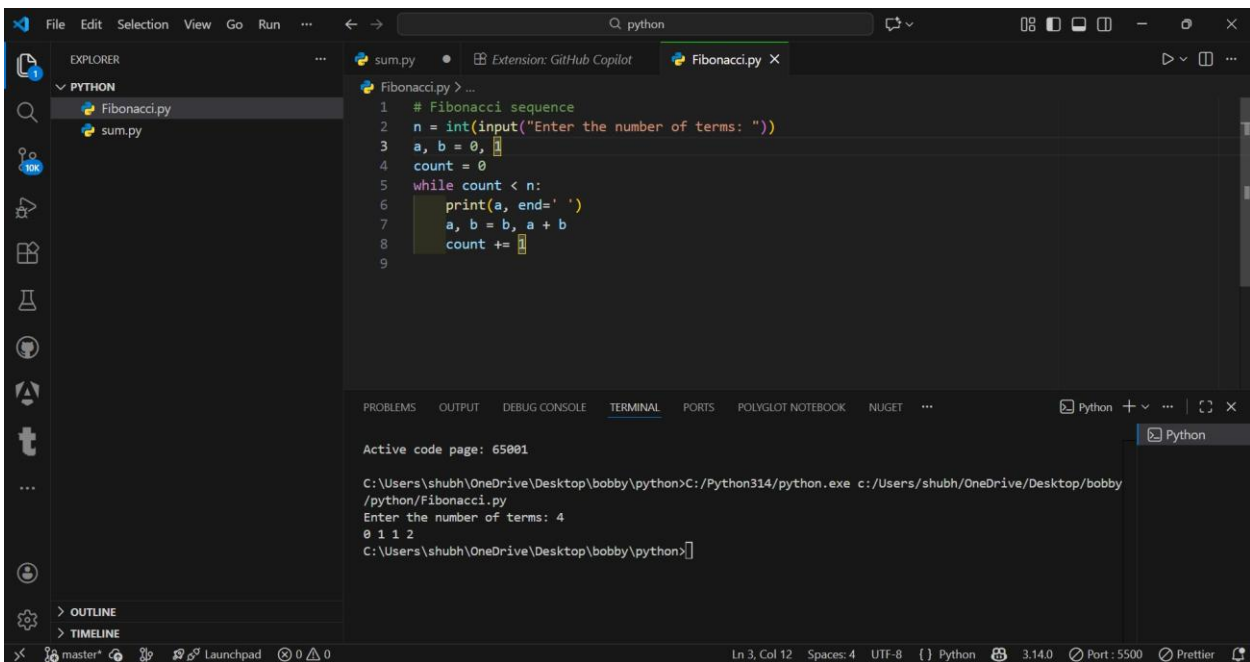
# usage
print(add(3, 4)) # prints 7
```

Note: use `int()` instead of `float()` if you only want integers.



Task 1: AI-Generated Logic Without Modularization (Procedural Fibonacci) :

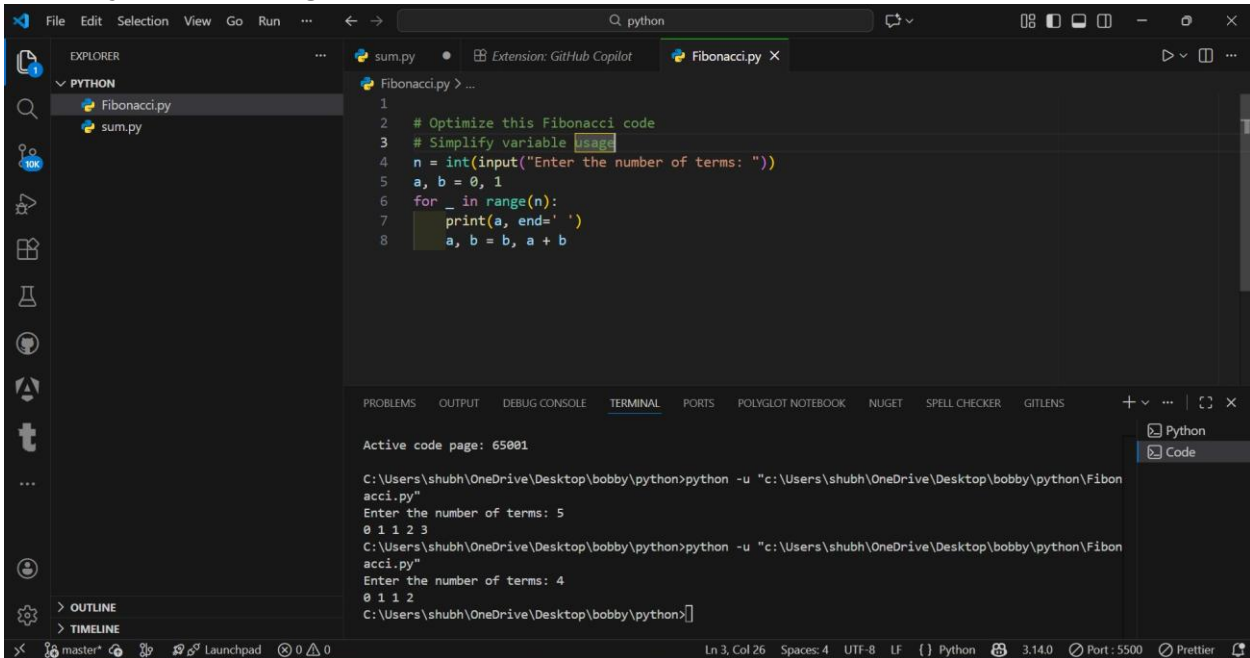
Fibonacci sequence



Task 2: AI Code Optimization & Cleanup :

Optimize this Fibonacci code

Simplify variable usage



The screenshot shows the Visual Studio Code editor with a Python file named `Fibonacci.py`. The code is as follows:

```
1
2 # Optimize this Fibonacci code
3 # Simplify variable usage
4 n = int(input("Enter the number of terms: "))
5 a, b = 0, 1
6 for _ in range(n):
7     print(a, end=' ')
8     a, b = b, a + b
```

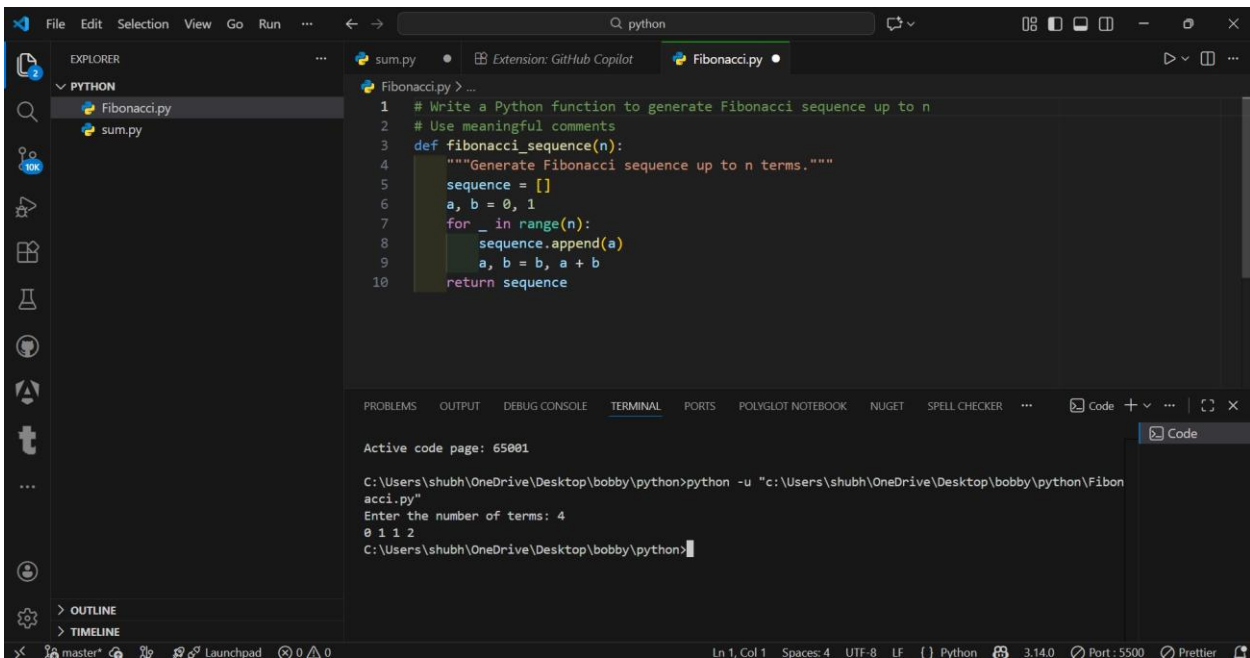
The terminal output shows the execution of the script:

```
C:\Users\shubh\OneDrive\Desktop\bobby\python>python -u "c:\Users\shubh\OneDrive\Desktop\bobby\python\Fibonacci.py"
Enter the number of terms: 5
0 1 1 2 3
C:\Users\shubh\OneDrive\Desktop\bobby\python>python -u "c:\Users\shubh\OneDrive\Desktop\bobby\python\Fibonacci.py"
Enter the number of terms: 4
0 1 1 2
C:\Users\shubh\OneDrive\Desktop\bobby\python>
```

Task 3: Modular Design Using AI Assistance (Function-Based Fibonacci) :

Write a Python function to generate Fibonacci sequence up to n

Use meaningful comments



The screenshot shows the Visual Studio Code editor with a Python file named `Fibonacci.py`. The code is as follows:

```
1 # Write a Python function to generate Fibonacci sequence up to n
2 # Use meaningful comments
3 def fibonacci_sequence(n):
4     """Generate Fibonacci sequence up to n terms."""
5     sequence = []
6     a, b = 0, 1
7     for _ in range(n):
8         sequence.append(a)
9         a, b = b, a + b
10    return sequence
```

The terminal output shows the execution of the script:

```
C:\Users\shubh\OneDrive\Desktop\bobby\python>python -u "c:\Users\shubh\OneDrive\Desktop\bobby\python\Fibonacci.py"
Enter the number of terms: 4
0 1 1 2
C:\Users\shubh\OneDrive\Desktop\bobby\python>
```

Task 4: Comparative Analysis – Procedural vs Modular Code

| Criteria | Without Functions | With Functions |
|----------------------------|-------------------|----------------|
| Code Clarity | Lower | Higher |
| Reusability | No | Yes |
| Debugging | Harder | Easier |
| Scalability | Poor | Excellent |
| Suitable for Large Systems | No | Yes |

Task 5: Iterative vs Recursive Fibonacci (AI-Generated):

Generate Fibonacci using iterative approach

```

1 # Generate Fibonacci using iterative approach
2 # Generate Fibonacci using recursive approach
3 def fibonacci_iterative(n):
4     a, b = 0, 1
5     for _ in range(n):
6         print(a, end=' ')
7         a, b = b, a + b
8     print()

```

Active code page: 65001

```

C:\Users\shubh\OneDrive\Desktop\bobby\python>python -u "c:\Users\shubh\OneDrive\Desktop\bobby\python\Fibonacci.py"
Enter the number of terms: 4
0 1 1 2
C:\Users\shubh\OneDrive\Desktop\bobby\python>

```

Generate Fibonacci using recursive approach

