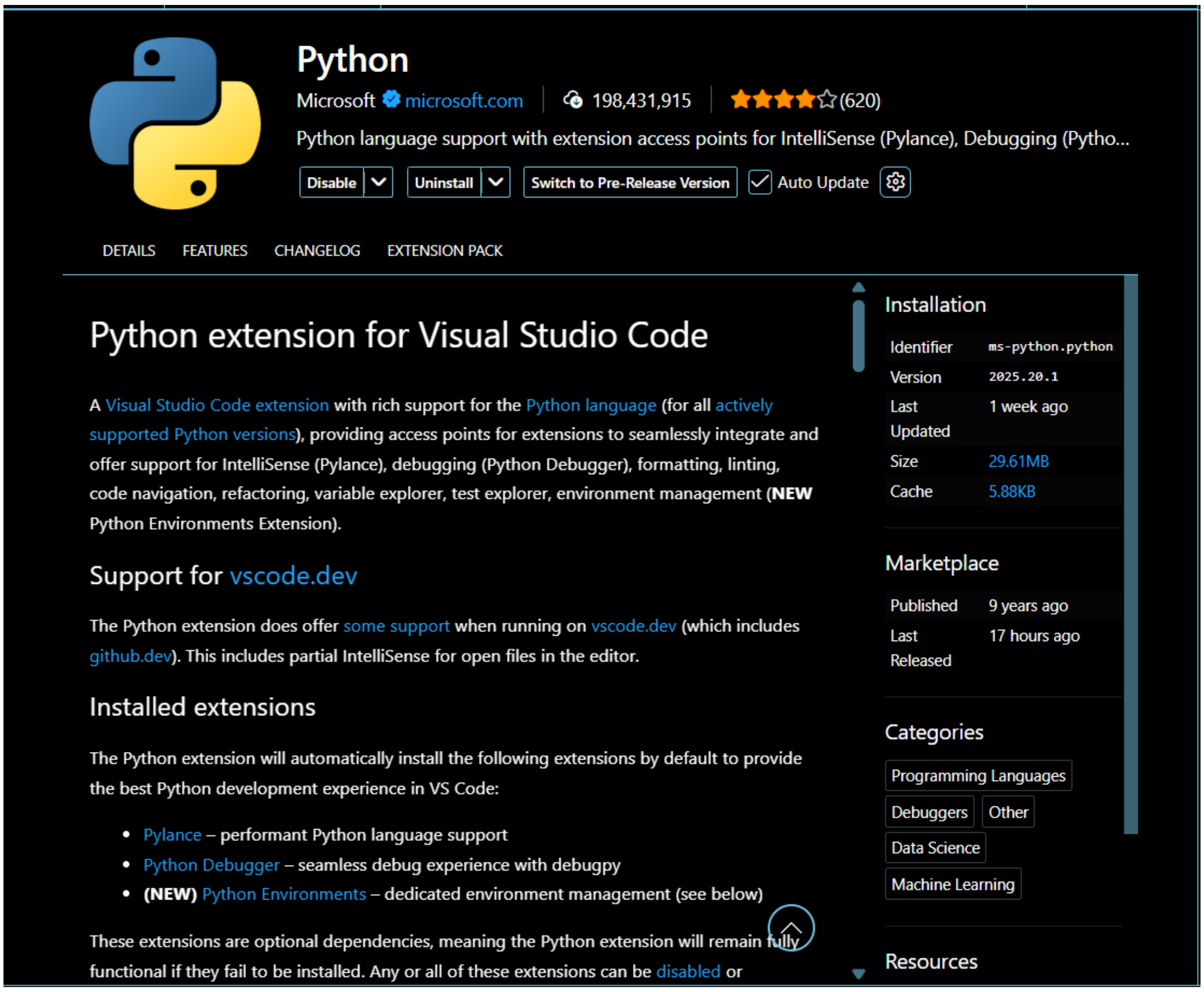
Assignment-1

Ai Assisstant coding

Name : Pritty Biswas

Hall-ticket : 2303A510F3


Python
Microsoft microsoft.com | 198,431,915 | (620)
Disable
Uninstall
Switch to Pre-Release Version
Auto Update
DETAILS
FEATURES
CHANGELOG
EXTENSION PACK

Python language support with extension access points for IntelliSense (Pylance), Debugging (Pytho...

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 github.dev). This includes partial IntelliSense for open files in the editor.

Installed extensions

The Python extension will automatically install the following extensions by default to provide the best Python development experience in VS Code:

- Pylance – performant Python language support
- Python Debugger – seamless debug experience with debugpy
- **(NEW)** Python Environments – dedicated environment management (see below)

These extensions are optional dependencies, meaning the Python extension will remain fully functional if they fail to be installed. Any or all of these extensions can be disabled or

Installation

Identifier	ms-python.python
Version	2025.20.1
Last Updated	1 week ago
Size	29.61MB
Cache	5.88KB

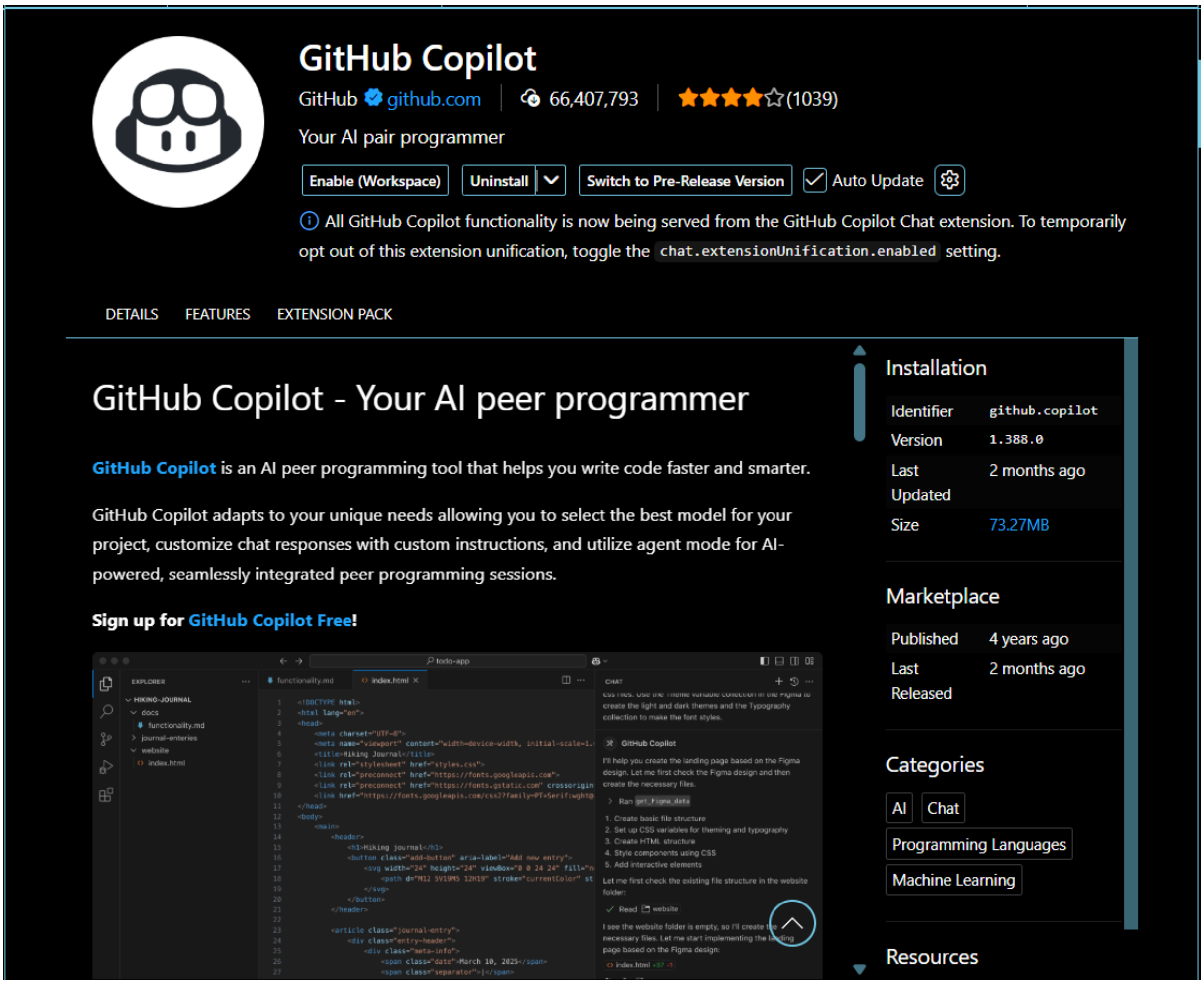
Marketplace

Published	9 years ago
Last Released	17 hours ago

Categories

Programming Languages
Debuggers
Other
Data Science
Machine Learning

Resources



GitHub Copilot

GitHub github.com | 66,407,793 | ★★★★☆ (1039)

Your AI pair programmer

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

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.

DETAILS FEATURES EXTENSION PACK

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

Identifier	`github.copilot`
Version	`1.388.0`
Last Updated	2 months ago
Size	73.27MB

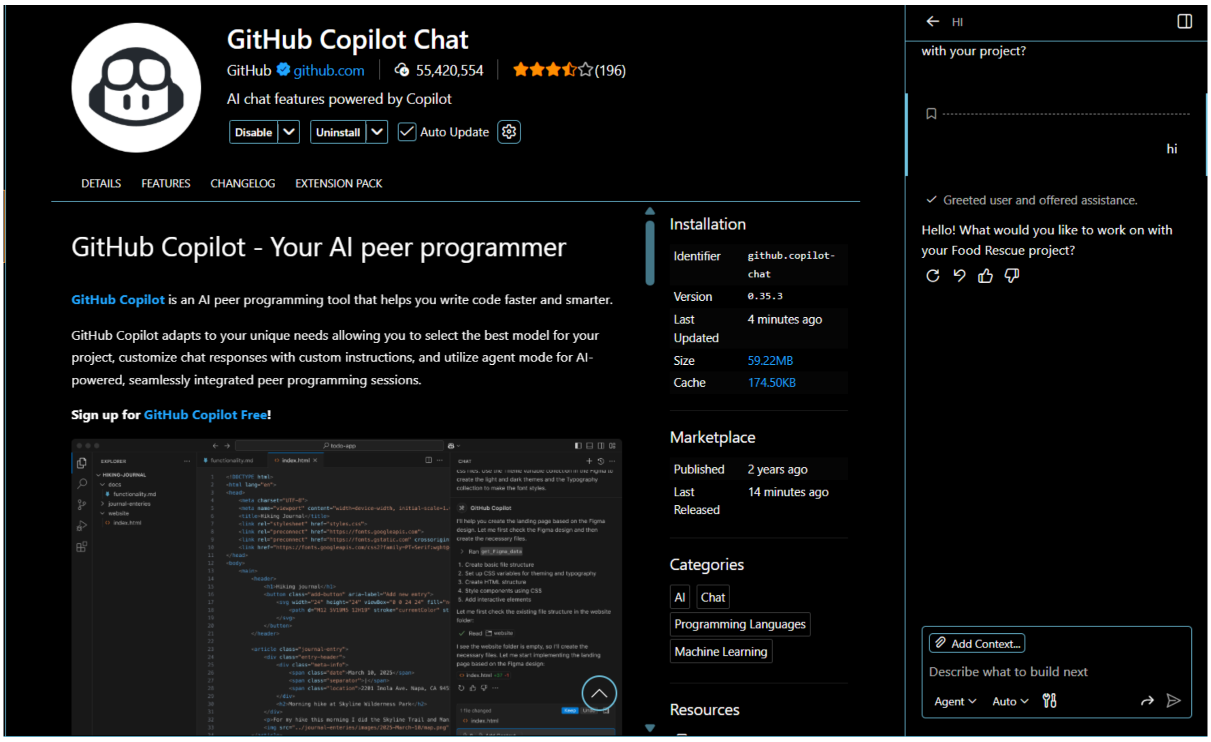
Marketplace

Published	4 years ago
Last Released	2 months ago

Categories

AI | Chat | Programming Languages | Machine Learning

Resources



GitHub Copilot Chat

GitHub github.com | 55,420,554 | ★★★★☆ (196)

AI chat features powered by Copilot

Disable | Uninstall | ✓ Auto Update

DETAILS FEATURES CHANGELOG EXTENSION PACK

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

Identifier	`github.copilot-chat`
Version	`0.35.3`
Last Updated	4 minutes ago
Size	59.22MB
Cache	174.50KB

Marketplace

Published	2 years ago
Last Released	14 minutes ago

Categories

AI | Chat | Programming Languages | Machine Learning

Resources

HI

with your project?

hi

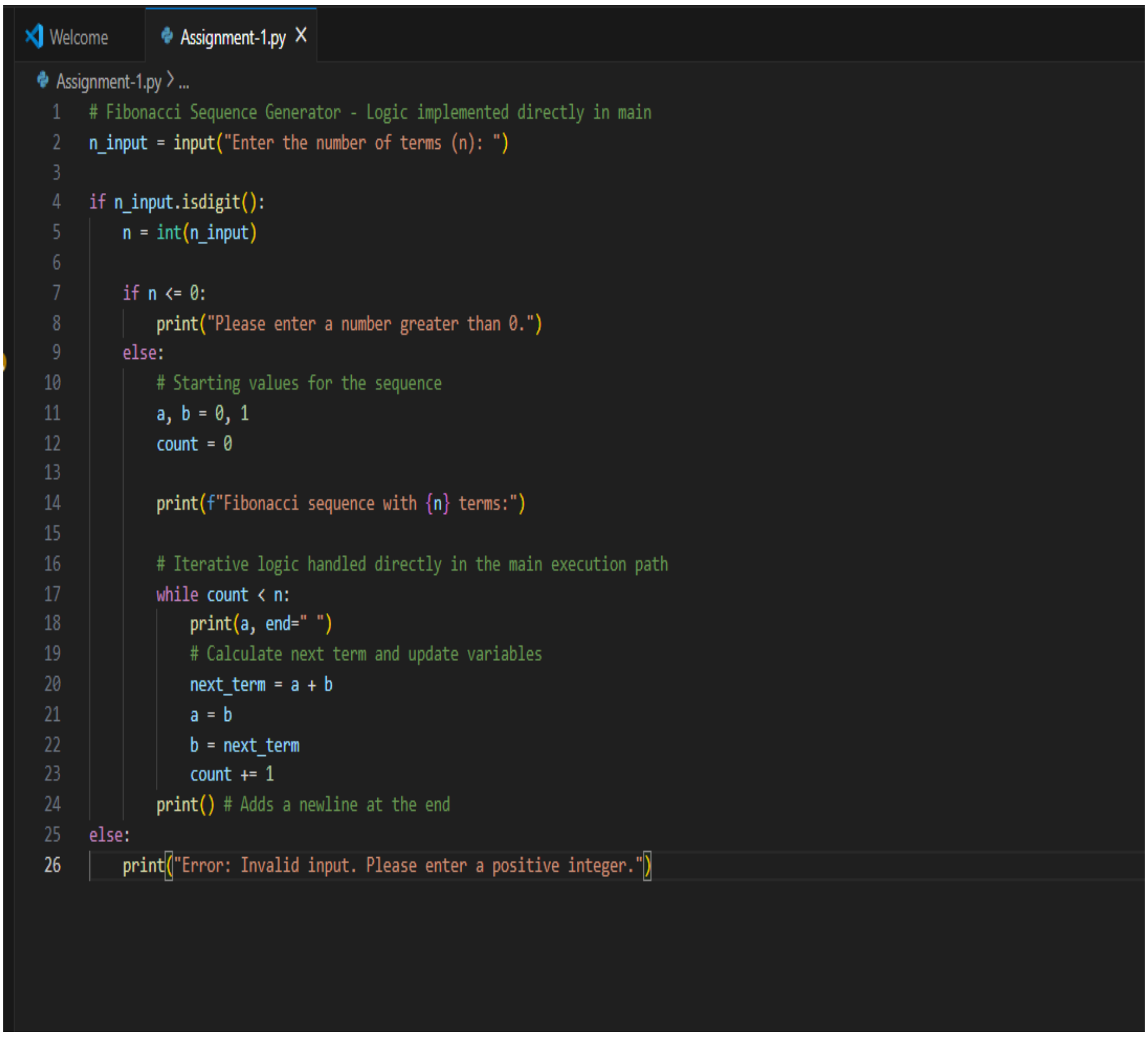
✓ Greeted user and offered assistance.

Hello! What would you like to work on with your Food Rescue project?

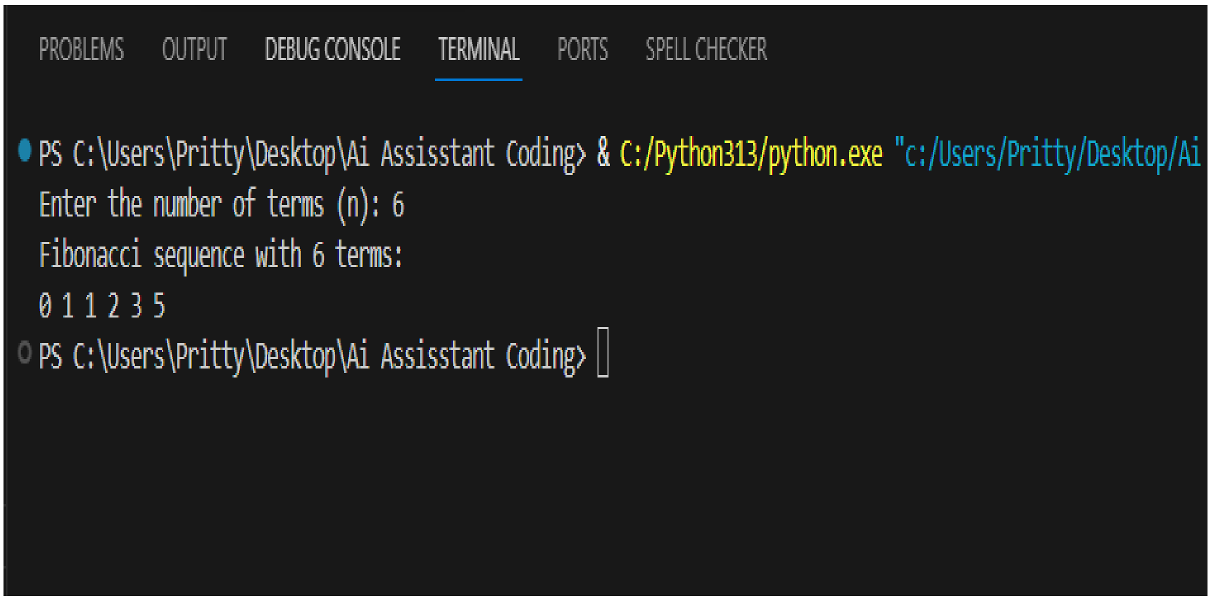
Add Context...

Describe what to build next

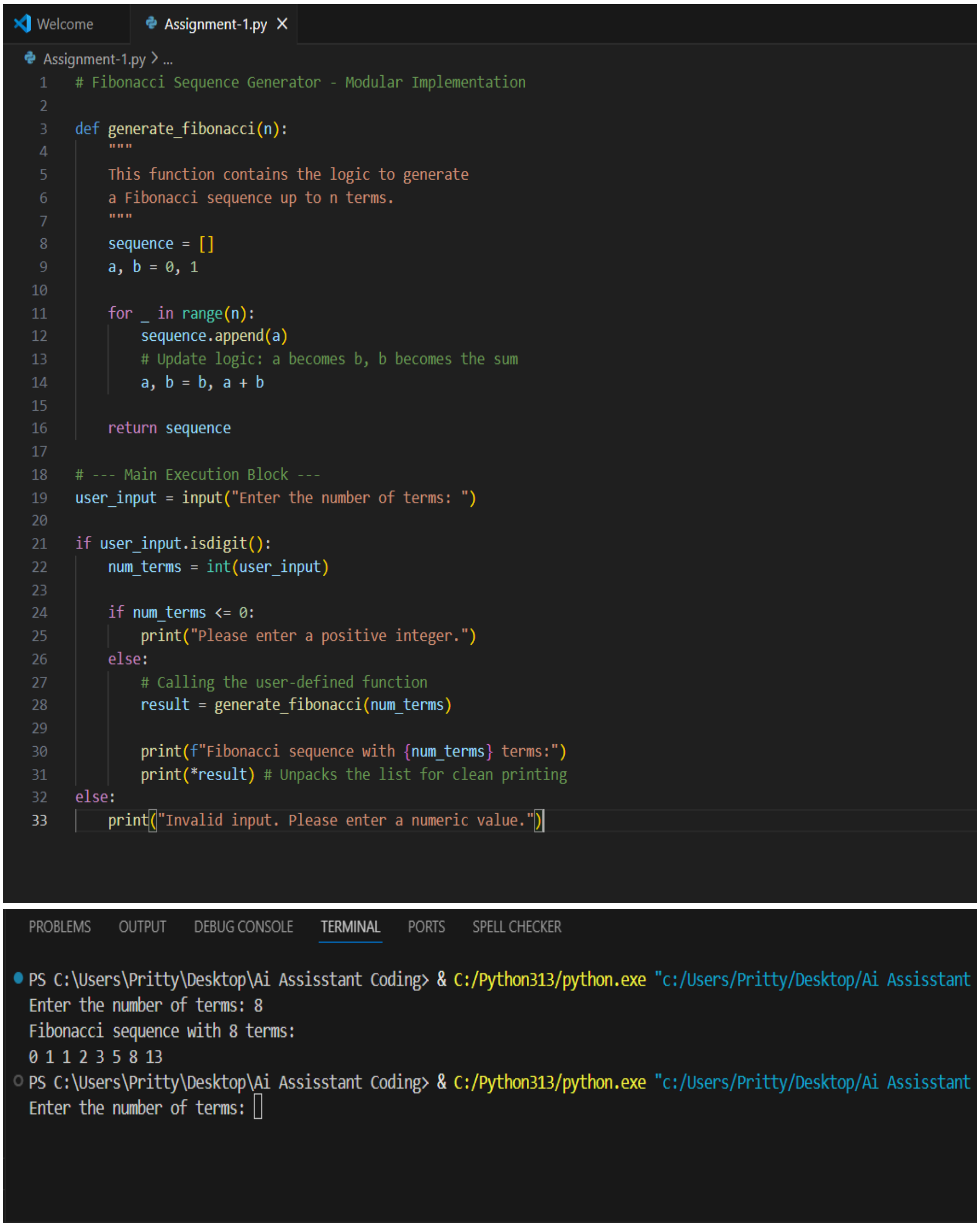
Agent Auto



```python
# Fibonacci Sequence Generator - Logic implemented directly in main
n_input = input("Enter the number of terms (n): ")

if n_input.isdigit():
    n = int(n_input)

    if n <= 0:
        print("Please enter a number greater than 0.")
    else:
        # Starting values for the sequence
        a, b = 0, 1
        count = 0

        print(f"Fibonacci sequence with {n} terms:")

        # Iterative logic handled directly in the main execution path
        while count < n:
            print(a, end=" ")
            # Calculate next term and update variables
            next_term = a + b
            a = b
            b = next_term
            count += 1
        print() # Adds a newline at the end
else:
    print("Error: Invalid input. Please enter a positive integer.")
```

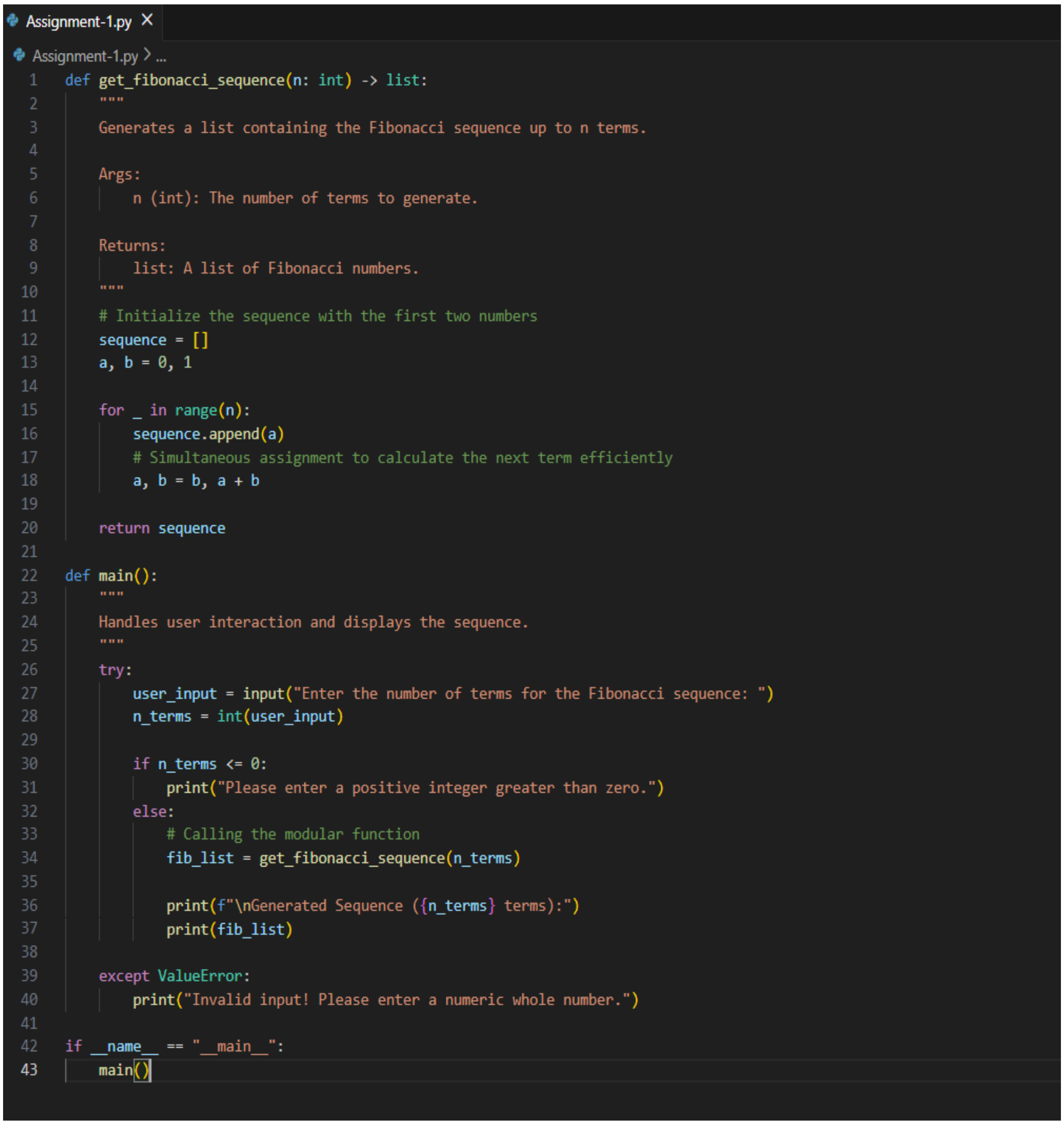


```
PS C:\Users\Pritty\Desktop\Ai Assisstant Coding> & C:/Python313/python.exe "c:/Users/Pritty/Desktop/Ai
Enter the number of terms (n): 6
Fibonacci sequence with 6 terms:
0 1 1 2 3 5
PS C:\Users\Pritty\Desktop\Ai Assisstant Coding>
```



```python
# Fibonacci Sequence Generator - Modular Implementation

def generate_fibonacci(n):
    """
    This function contains the logic to generate
    a Fibonacci sequence up to n terms.
    """
    sequence = []
    a, b = 0, 1

    for _ in range(n):
        sequence.append(a)
        # Update logic: a becomes b, b becomes the sum
        a, b = b, a + b

    return sequence

# --- Main Execution Block ---
user_input = input("Enter the number of terms: ")

if user_input.isdigit():
    num_terms = int(user_input)

    if num_terms <= 0:
        print("Please enter a positive integer.")
    else:
        # Calling the user-defined function
        result = generate_fibonacci(num_terms)

        print(f"Fibonacci sequence with {num_terms} terms:")
        print(*result) # Unpacks the list for clean printing
else:
    print("Invalid input. Please enter a numeric value.")
```

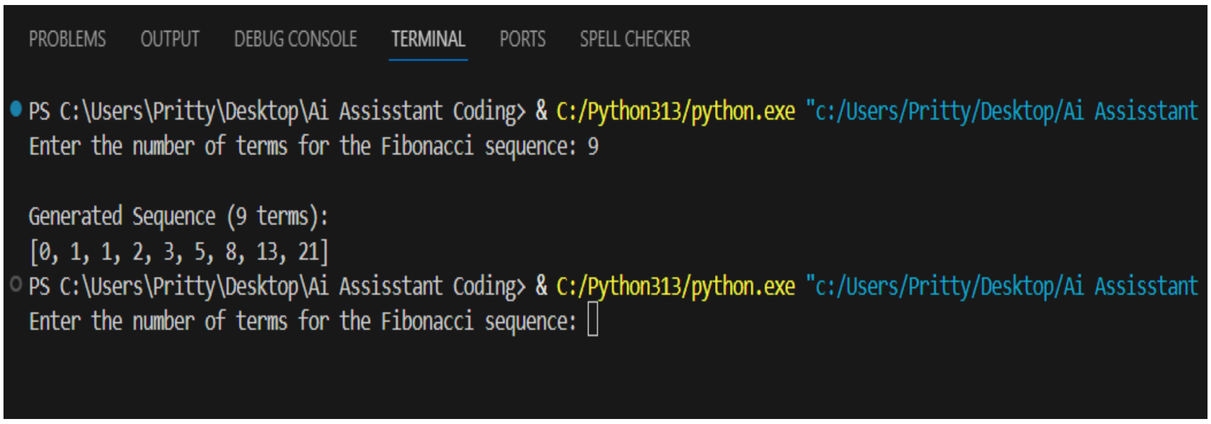
```
PS C:\Users\Pritty\Desktop\Ai Assisstant Coding> & C:/Python313/python.exe "c:/Users/Pritty/Desktop/Ai Assisstant
Enter the number of terms: 8
Fibonacci sequence with 8 terms:
0 1 1 2 3 5 8 13
PS C:\Users\Pritty\Desktop\Ai Assisstant Coding> & C:/Python313/python.exe "c:/Users/Pritty/Desktop/Ai Assisstant
Enter the number of terms:
```



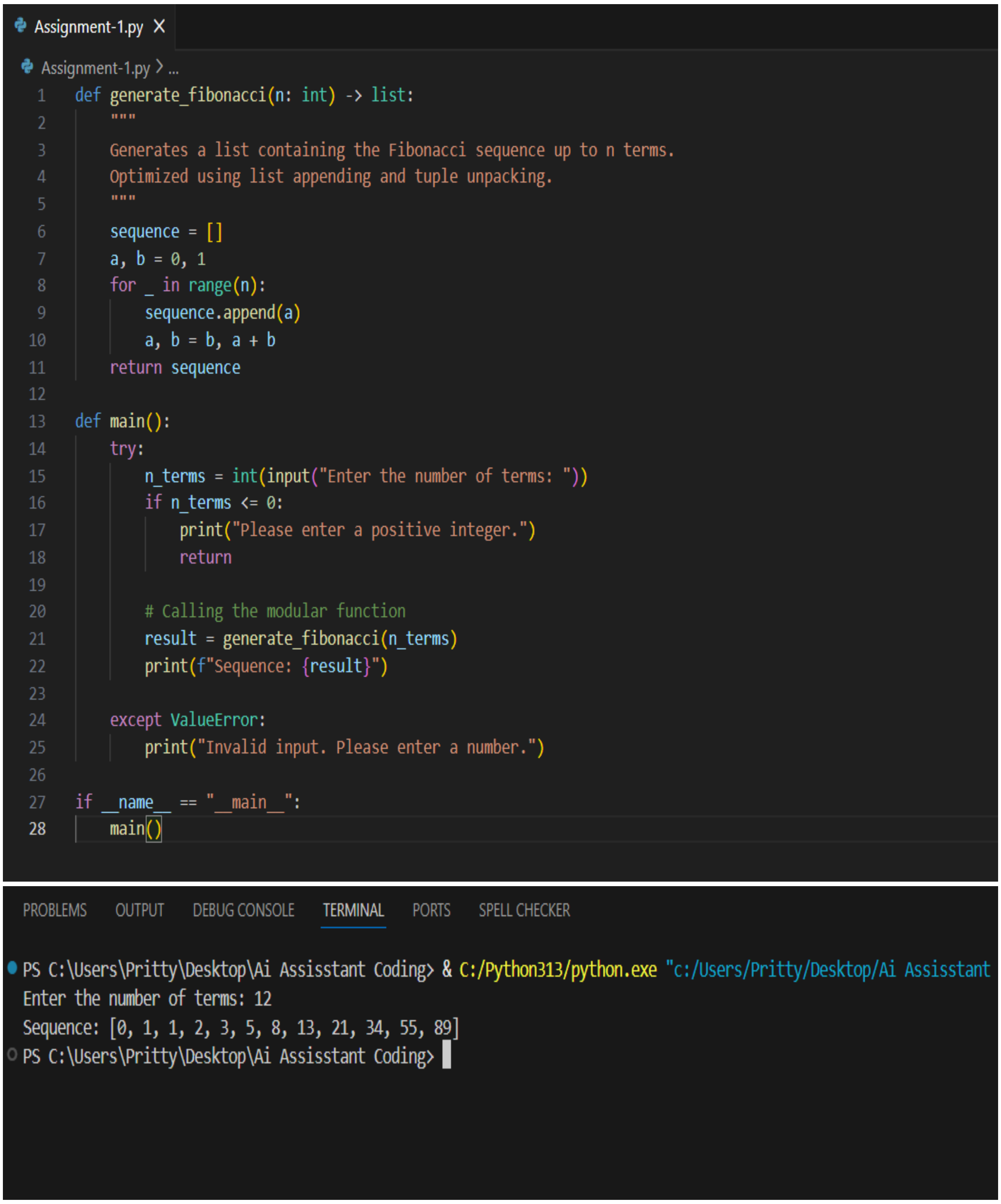
Assignment-1.py ×

Assignment-1.py > ...

```python
def get_fibonacci_sequence(n: int) -> list:
    """
    Generates a list containing the Fibonacci sequence up to n terms.

    Args:
        n (int): The number of terms to generate.

    Returns:
        list: A list of Fibonacci numbers.
    """
    # Initialize the sequence with the first two numbers
    sequence = []
    a, b = 0, 1

    for _ in range(n):
        sequence.append(a)
        # Simultaneous assignment to calculate the next term efficiently
        a, b = b, a + b

    return sequence

def main():
    """
    Handles user interaction and displays the sequence.
    """
    try:
        user_input = input("Enter the number of terms for the Fibonacci sequence: ")
        n_terms = int(user_input)

        if n_terms <= 0:
            print("Please enter a positive integer greater than zero.")
        else:
            # Calling the modular function
            fib_list = get_fibonacci_sequence(n_terms)

            print(f"\nGenerated Sequence ({n_terms} terms):")
            print(fib_list)

    except ValueError:
        print("Invalid input! Please enter a numeric whole number.")

if __name__ == "__main__":
    main()
```

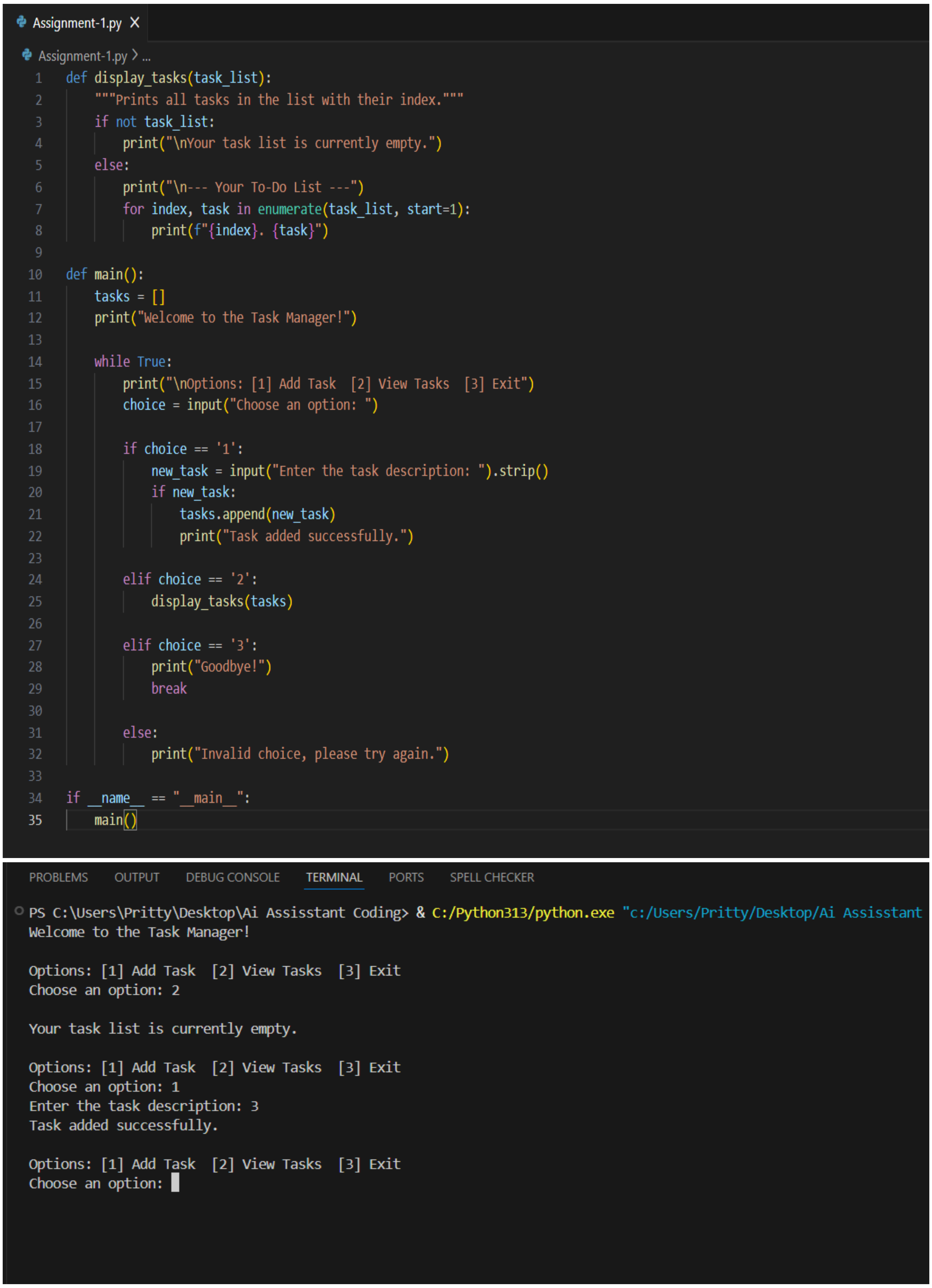


```
PS C:\Users\Pritty\Desktop\Ai Assistant Coding> & C:/Python313/python.exe "c:/Users/Pritty/Desktop/Ai Assistant
Enter the number of terms for the Fibonacci sequence: 9

Generated Sequence (9 terms):
[0, 1, 1, 2, 3, 5, 8, 13, 21]
PS C:\Users\Pritty\Desktop\Ai Assistant Coding> & C:/Python313/python.exe "c:/Users/Pritty/Desktop/Ai Assistant
Enter the number of terms for the Fibonacci sequence: 
```



```python
def generate_fibonacci(n: int) -> list:
    """
    Generates a list containing the Fibonacci sequence up to n terms.
    Optimized using list appending and tuple unpacking.
    """
    sequence = []
    a, b = 0, 1
    for _ in range(n):
        sequence.append(a)
        a, b = b, a + b
    return sequence

def main():
    try:
        n_terms = int(input("Enter the number of terms: "))
        if n_terms <= 0:
            print("Please enter a positive integer.")
            return

        # Calling the modular function
        result = generate_fibonacci(n_terms)
        print(f"Sequence: {result}")

    except ValueError:
        print("Invalid input. Please enter a number.")

if __name__ == "__main__":
    main()
```

```
PS C:\Users\Pritty\Desktop\Ai Assisstant Coding> & C:/Python313/python.exe "c:/Users/Pritty/Desktop/Ai Assisstant
Enter the number of terms: 12
Sequence: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
PS C:\Users\Pritty\Desktop\Ai Assisstant Coding>
```



```python
def display_tasks(task_list):
    """Prints all tasks in the list with their index."""
    if not task_list:
        print("\nYour task list is currently empty.")
    else:
        print("\n--- Your To-Do List ---")
        for index, task in enumerate(task_list, start=1):
            print(f"{index}. {task}")

def main():
    tasks = []
    print("Welcome to the Task Manager!")

    while True:
        print("\nOptions: [1] Add Task  [2] View Tasks  [3] Exit")
        choice = input("Choose an option: ")

        if choice == '1':
            new_task = input("Enter the task description: ").strip()
            if new_task:
                tasks.append(new_task)
                print("Task added successfully.")

        elif choice == '2':
            display_tasks(tasks)

        elif choice == '3':
            print("Goodbye!")
            break

        else:
            print("Invalid choice, please try again.")

if __name__ == "__main__":
    main()
```

```
PS C:\Users\Pritty\Desktop\Ai Assisstant Coding> & C:/Python313/python.exe "c:/Users/Pritty/Desktop/Ai Assisstant
Welcome to the Task Manager!

Options: [1] Add Task  [2] View Tasks  [3] Exit
Choose an option: 2

Your task list is currently empty.

Options: [1] Add Task  [2] View Tasks  [3] Exit
Choose an option: 1
Enter the task description: 3
Task added successfully.

Options: [1] Add Task  [2] View Tasks  [3] Exit
Choose an option: 
```