Assignment-1

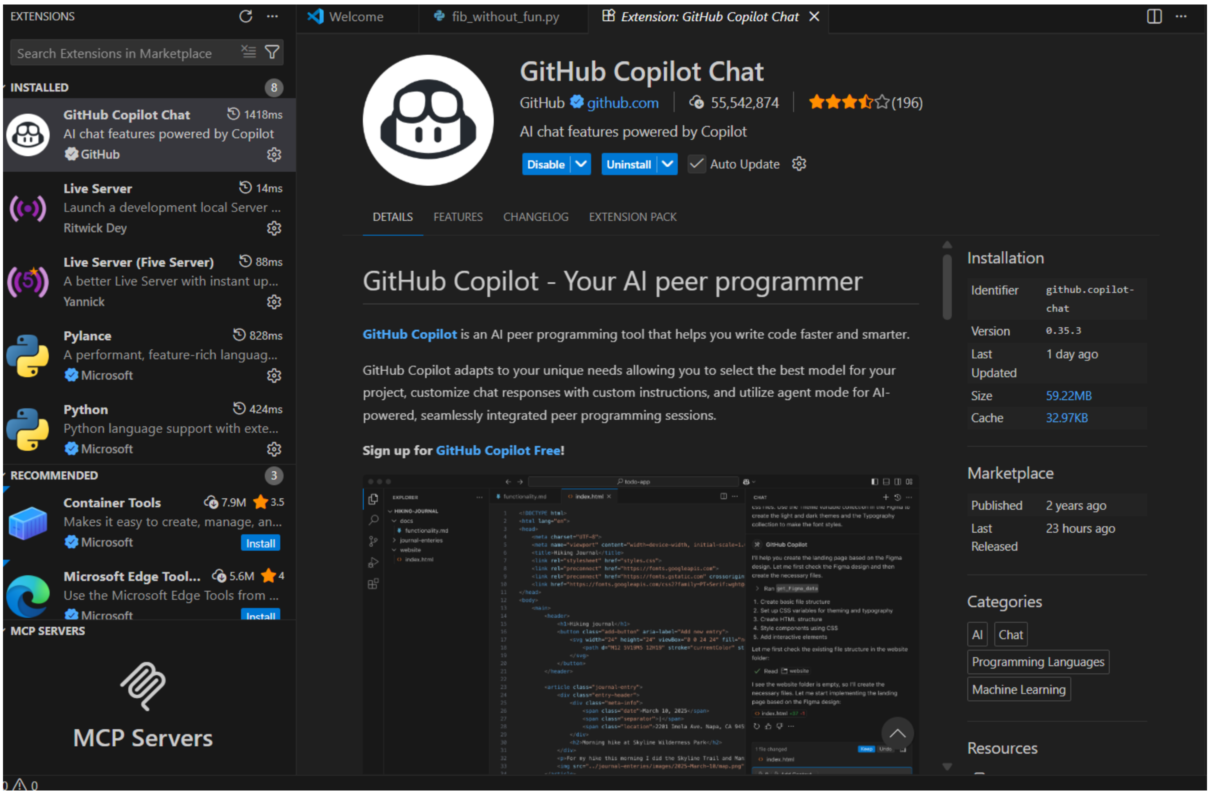
U.Poojitha

2303A51304

Batch-05

Task-0:

Installation of GitHub copilot


EXTENSIONS
Search Extensions in Marketplace
INSTALLED 8
GitHub Copilot Chat 1418ms
AI chat features powered by Copilot
GitHub
Live Server 14ms
Launch a development local Server ...
Ritwick Dey
Live Server (Five Server) 88ms
A better Live Server with instant up...
Yannick
Pylance 828ms
A performant, feature-rich languag...
Microsoft
Python 424ms
Python language support with exte...
Microsoft
RECOMMENDED 3
Container Tools 7.9M 3.5
Makes it easy to create, manage, an...
Microsoft
Install
Microsoft Edge Tool... 5.6M 4
Use the Microsoft Edge Tools from ...
Microsoft
Install
MCP SERVERS
MCP Servers
Welcome
fib_without_fun.py
Extension: GitHub Copilot Chat
GitHub Copilot Chat
GitHub github.com 55,542,874 (196)
AI chat features powered by Copilot
Disable
Uninstall
Auto Update
DETAILS
FEATURES
CHANGELOG
EXTENSION PACK
Installation
Marketplace
Categories
AI
Chat
Programming Languages
Machine Learning
Resources

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!

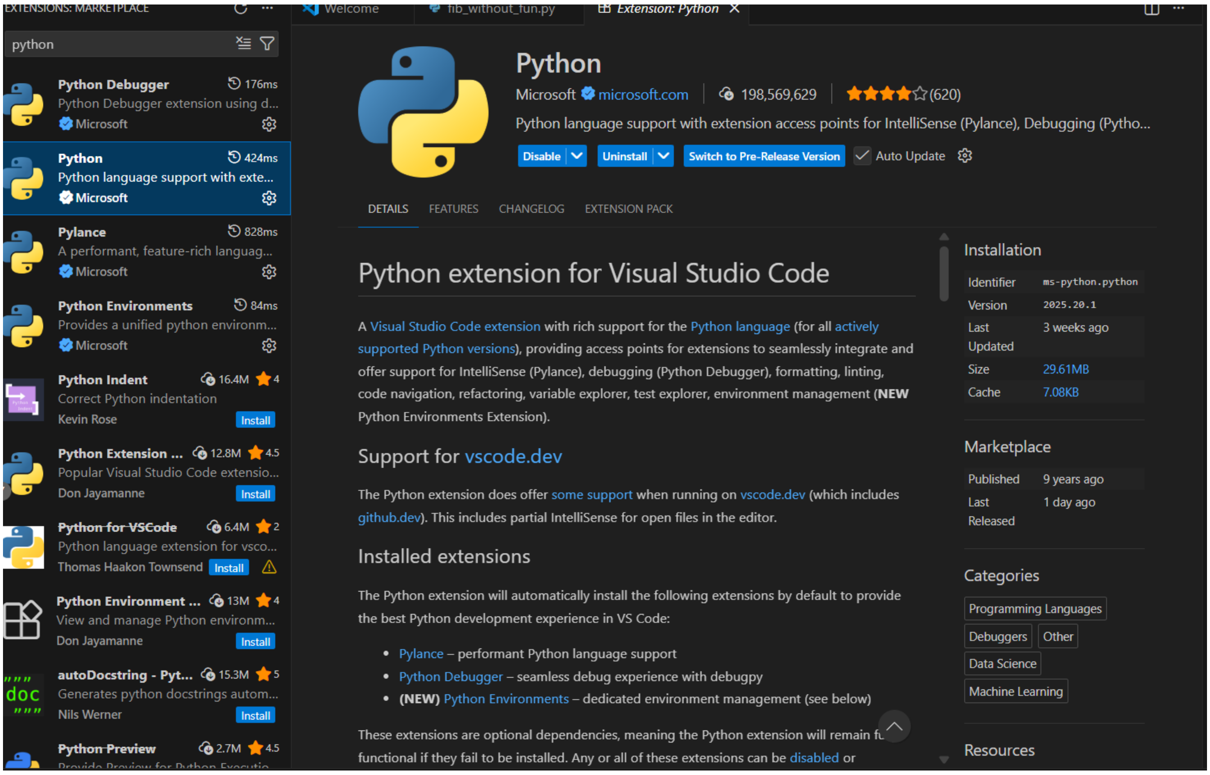
Installation

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

Marketplace

Published	2 years ago
Last Released	23 hours ago

Installation of Python


EXTENSIONS: MARKETPLACE
python
Welcome
fib_without_fun.py
Extension: Python
Python Debugger 176ms
Python Debugger extension using d...
Microsoft
Python 424ms
Python language support with exte...
Microsoft
Pylance 828ms
A performant, feature-rich languag...
Microsoft
Python Environments 84ms
Provides a unified python environm...
Microsoft
Python Indent 16.4M 4
Correct Python indentation
Kevin Rose
Install
Python Extension ... 12.8M 4.5
Popular Visual Studio Code extensio...
Don Jayamanne
Install
~~Python for VSCode~~ 6.4M 2
Python language extension for vsco...
Thomas Haakon Townsend
Install
Python Environment ... 13M 4
View and manage Python environm...
Don Jayamanne
Install
autoDocstring - Pyt... 15.3M 5
Generates python docstrings autom...
Nils Werner
Install
~~Python Preview~~ 2.7M 4.5
Disable
Uninstall
Switch to Pre-Release Version
Auto Update
DETAILS
FEATURES
CHANGELOG
EXTENSION PACK

Python

Microsoft microsoft.com | 198,569,629 | ★★★★☆ (620)

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 fu 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	3 weeks ago
Size	29.61MB
Cache	7.08KB

Marketplace

Published	9 years ago
Last Released	1 day ago

Categories

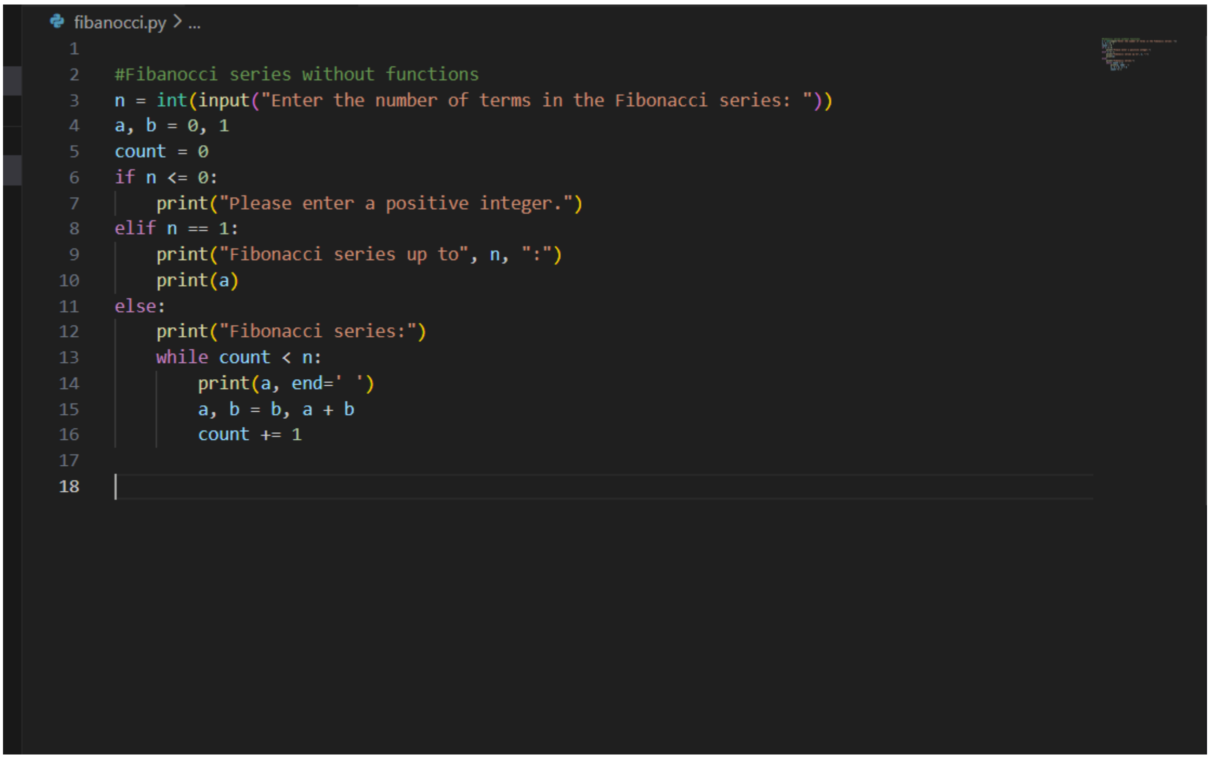
Programming Languages
Debuggers
Other
Data Science
Machine Learning

Resources

Task-1:

Prompt: Fibanocci series without using functions

Code:

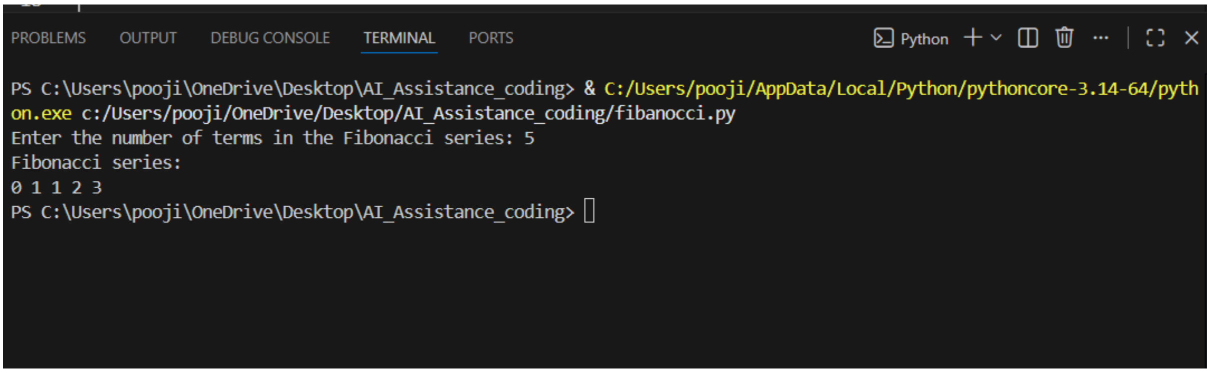


```python

#Fibanocci series without functions
n = int(input("Enter the number of terms in the Fibonacci series: "))
a, b = 0, 1
count = 0
if n <= 0:
    print("Please enter a positive integer.")
elif n == 1:
    print("Fibonacci series up to", n, ":")
    print(a)
else:
    print("Fibonacci series:")
    while count < n:
        print(a, end=' ')
        a, b = b, a + b
        count += 1

```

Output:



```
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/AppData/Local/Python/pythoncore-3.14-64/pyth
on.exe c:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/fibanocci.py
Enter the number of terms in the Fibonacci series: 5
Fibonacci series:
0 1 1 2 3
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> 
```

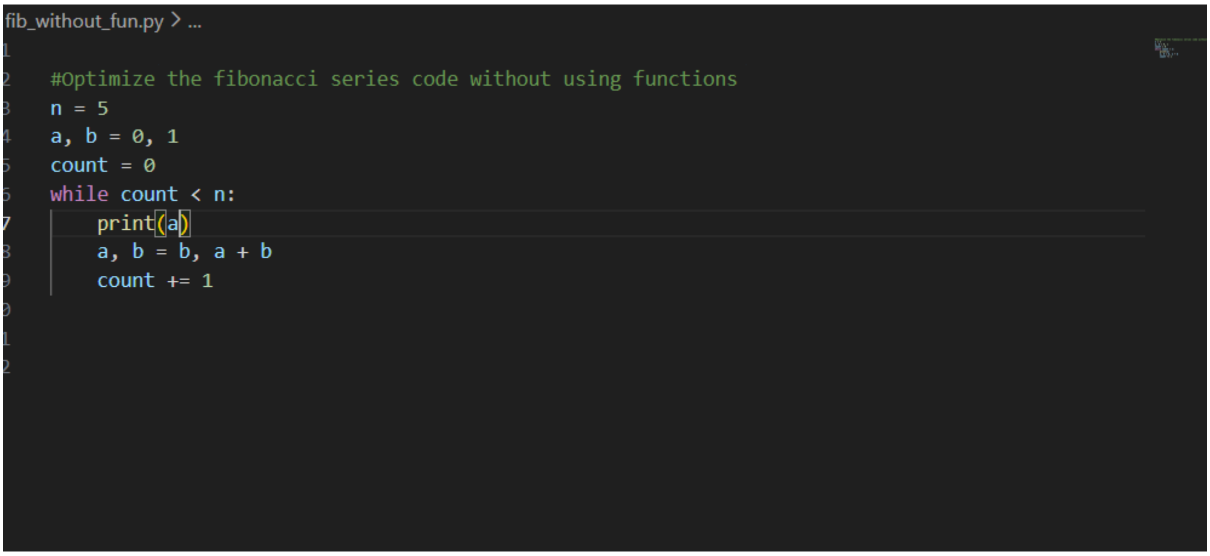
Explanation:

The above code we have used an integer n to take the input from the user to print the fibanocci series by using the above logic ,which I have mentioned without using the functions. It is an basic code for finding the fibanocci series.

Task-2:

Prompt: Optimize the fibanocci series code without using functions

Code:

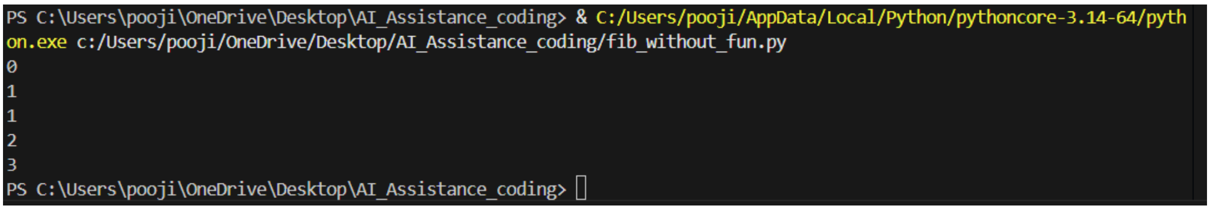


fib_without_fun.py > ...

```

#Optimize the fibonacci series code without using functions
n = 5
a, b = 0, 1
count = 0
while count < n:
    print(a)
    a, b = b, a + b
    count += 1
```

Output:



```
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/AppData/Local/Python/pythoncore-3.14-64/pyth
on.exe c:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/fib_without_fun.py
0
1
1
2
3
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> 
```

Explanation:

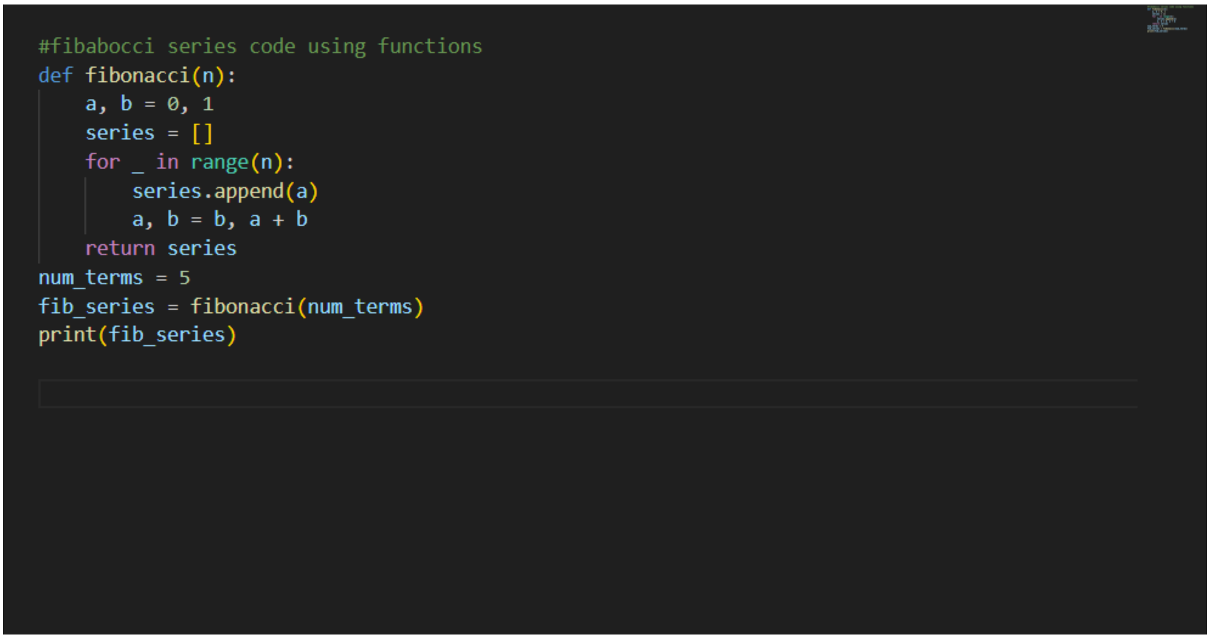
Normal fibanocci series which I have used in task-1 is inefficient because it recalculates the values repeatedly using the loops. which may increase the time complexity.

Whereas the optimized code stores the previous two fibanocci numbers and computes each value only once. This reduces time complexity and it works more efficiently. Optimized code is easier to understand

Task-3:

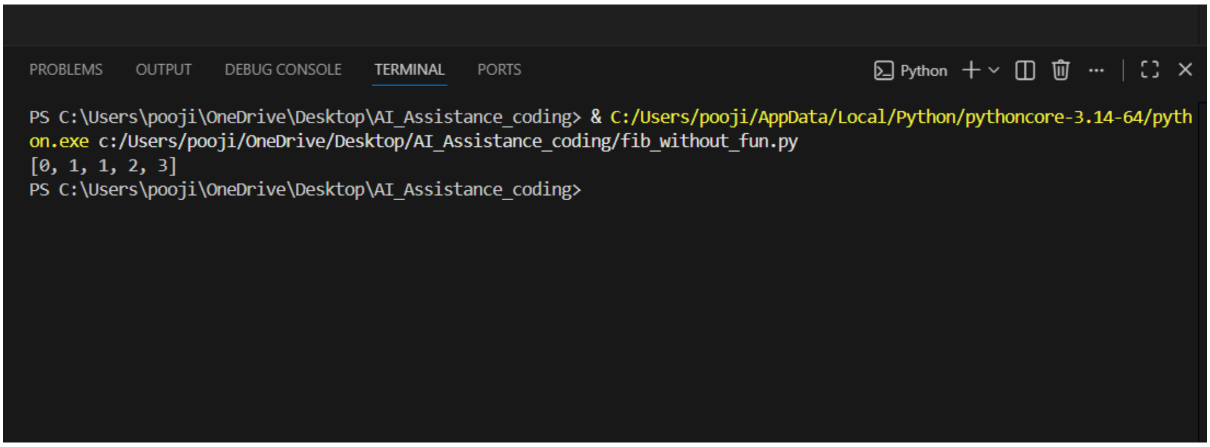
Prompt: Fibanocci series code using functions

Code:



```python
#fibabocci series code using functions
def fibonacci(n):
    a, b = 0, 1
    series = []
    for _ in range(n):
        series.append(a)
        a, b = b, a + b
    return series
num_terms = 5
fib_series = fibonacci(num_terms)
print(fib_series)
```

Output:



```
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/AppData/Local/Python/pythoncore-3.14-64/pyth
on.exe c:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/fib_without_fun.py
[0, 1, 1, 2, 3]
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding>
```

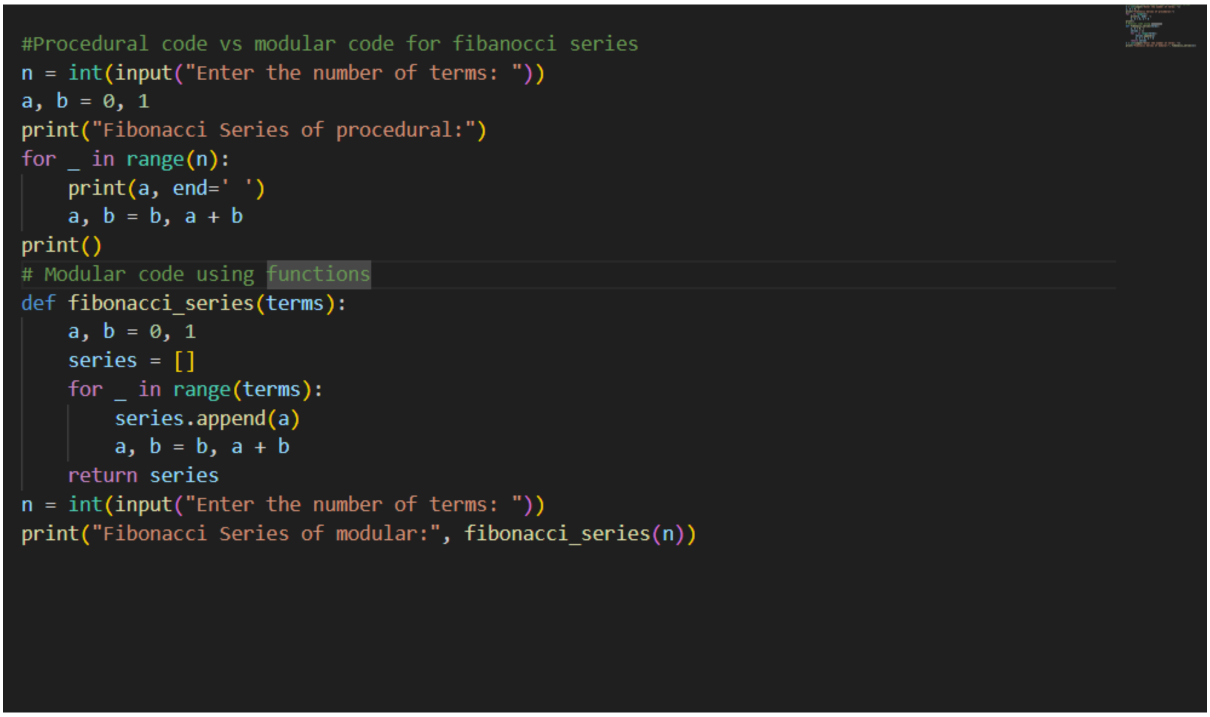
Explanation:

By using the above code we can organize the code into reusable blocks. They avoid code repetition by calling the same logic multiple times. Functions make programs easier to understand. Functions improve the modularity.

Task-4:

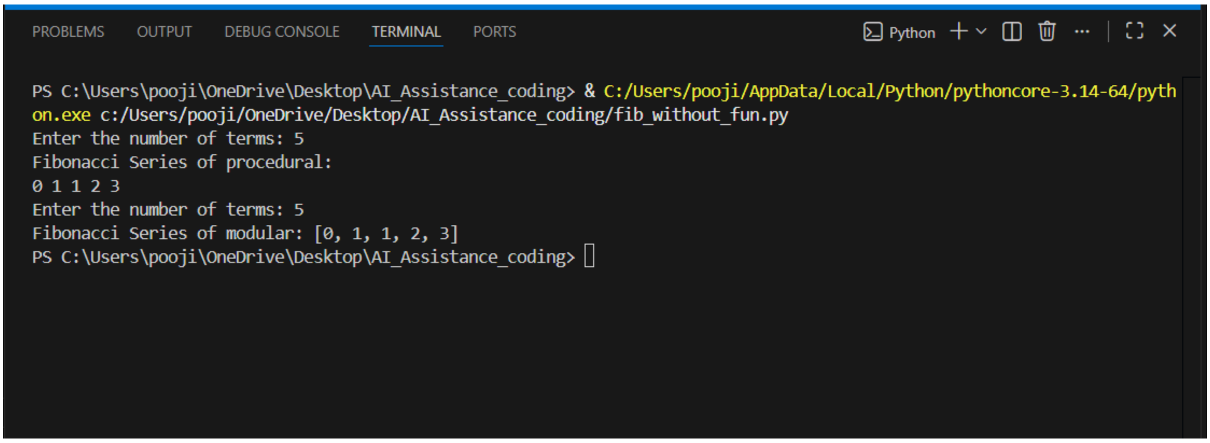
Prompt: Procedural vs modular code for fibanocci series

Code:



```python
#Procedural code vs modular code for fibanocci series
n = int(input("Enter the number of terms: "))
a, b = 0, 1
print("Fibonacci Series of procedural:")
for _ in range(n):
    print(a, end=' ')
    a, b = b, a + b
print()
# Modular code using functions
def fibonacci_series(terms):
    a, b = 0, 1
    series = []
    for _ in range(terms):
        series.append(a)
        a, b = b, a + b
    return series
n = int(input("Enter the number of terms: "))
print("Fibonacci Series of modular:", fibonacci_series(n))
```

Output:



```
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/AppData/Local/Python/pythoncore-3.14-64/pyth
on.exe c:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/fib_without_fun.py
Enter the number of terms: 5
Fibonacci Series of procedural:
0 1 1 2 3
Enter the number of terms: 5
Fibonacci Series of modular: [0, 1, 1, 2, 3]
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> 
```

Explanation:

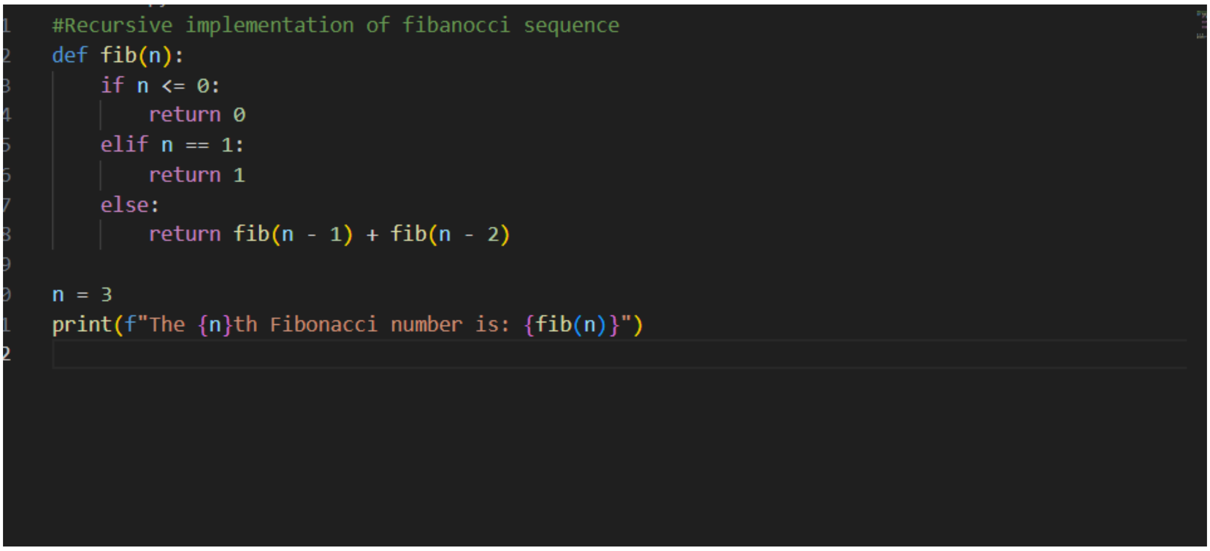
Procedural Fibanocci code writes all logic in one block, it makes the program longer and harder to understand. It is difficult to reuse

Modular Fibanocci code divides the logic into functions, improves the structure and clarity. It allows to reuse the fibanocci function in multiple programs. Modular code improves the readability, scalability.

Task-5:

Prompt: Recursive implementation of fibanocci sequence

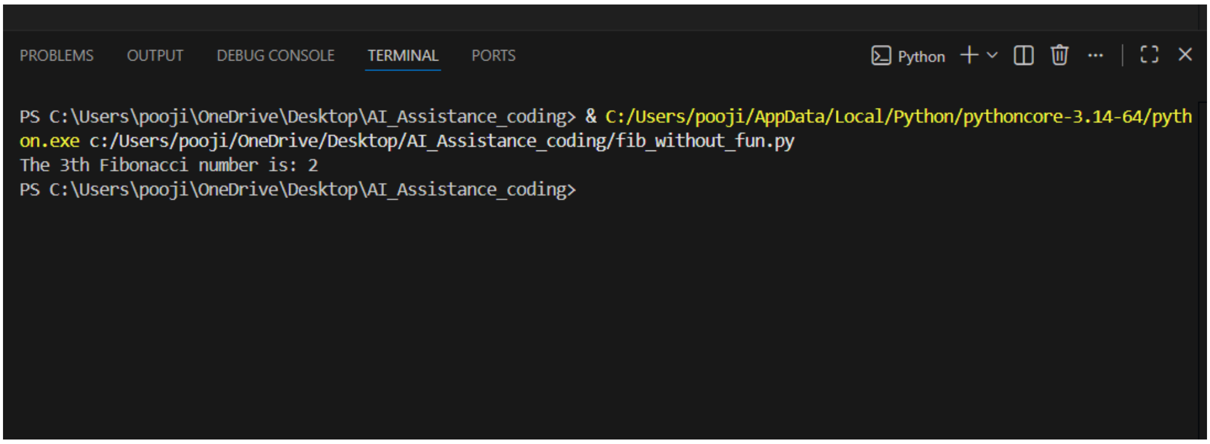
Input:



```python
#Recursive implementation of fibanocci sequence
def fib(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n - 1) + fib(n - 2)

n = 3
print(f"The {n}th Fibonacci number is: {fib(n)}")
```

Output:



```
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/AppData/Local/Python/pythoncore-3.14-64/pyth
on.exe c:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/fib_without_fun.py
The 3th Fibonacci number is: 2
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding>
```

Explanation:

This code uses recursion to calculate the fibanocci series by breaking the problem into sub problems. The function fib(n) calls itself to compute fib(n-1) and fib(n-2) until it reaches the base cases. When n is 0 or negative , the function returns 0, when n is 1, it returns 1.

For n=3 ,the recursive calls compute fib(2) and fib(1) and add their results. The final output prints the 3rd fibanocci number which is 2