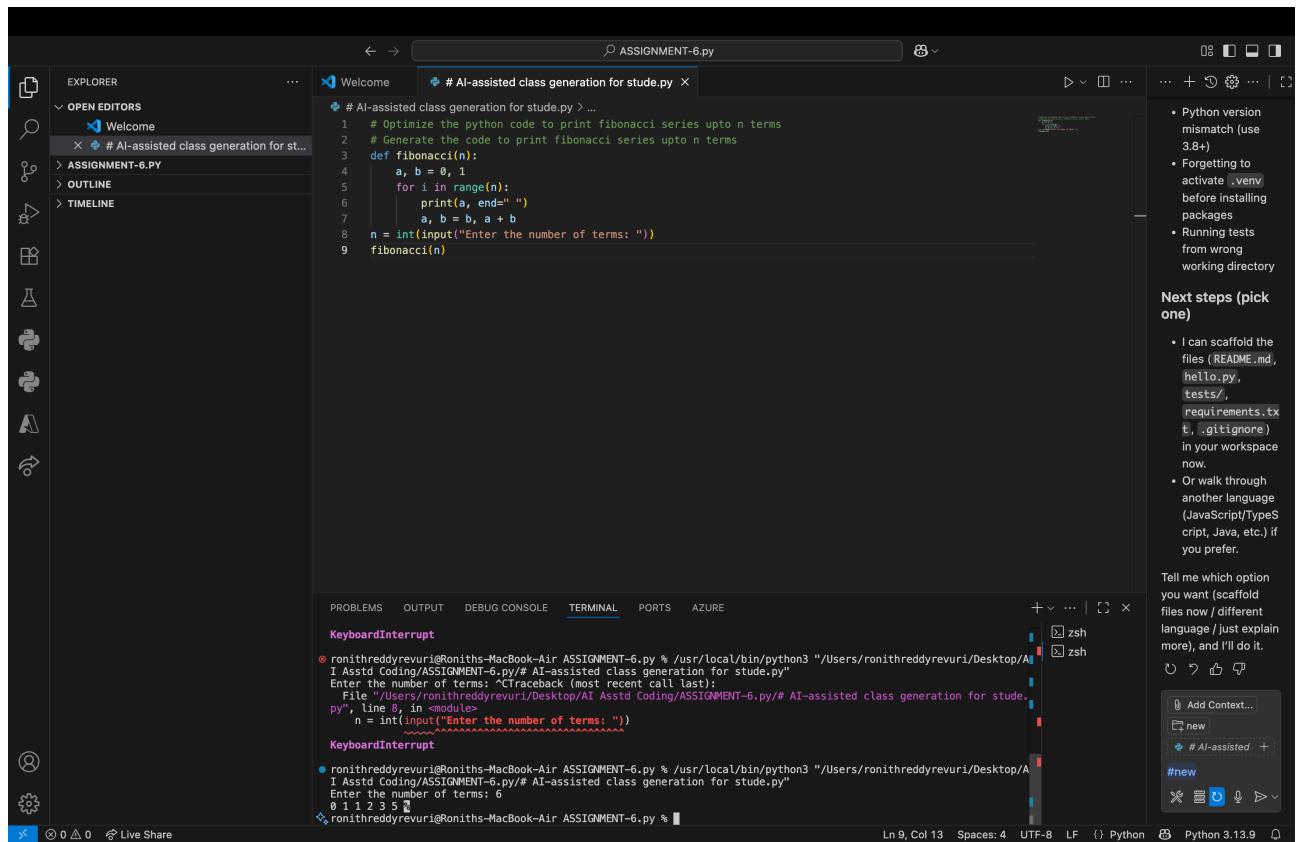


The screenshot shows the Visual Studio Code interface with the following details:

- Explorer:** Shows an open editor for "Assignment-6.py" and a "Welcome" file.
- Editor:** Displays the following Python code for generating a Fibonacci series:

```
1  # AI-assisted class generation for stude.py
2  # Generate python code to print fibonacci series upto n terms without using functions
3  n = int(input("Enter the number of terms: "))
4  for i in range(n):
5      print(a, end=" ")
6      a, b = b, a + b
7  def print_fibonacci(n):
8      a, b = 0, 1
9      for i in range(n):
10         print(a, end=" ")
11         a, b = b, a + b
```
- Terminal:** Shows the command line output of the script execution. The user enters "6" and the script prints the first 6 terms of the Fibonacci sequence: 0 1 1 2 3 5.
- Output:** Shows the command line output of the script execution. The user enters "6" and the script prints the first 6 terms of the Fibonacci sequence: 0 1 1 2 3 5.
- Problems:** Shows no problems.
- Terminal:** Shows the command line output of the script execution. The user enters "6" and the script prints the first 6 terms of the Fibonacci sequence: 0 1 1 2 3 5.
- Right Panel:** Includes a "Next steps (pick one)" section with the following items:
 - I can scaffold the files (README.md, hello.py, tests/, requirements.txt t..gitignore) in your workspace now.
 - Or walk through another language (JavaScript/TypeScript, Java, etc.) if you prefer. and a "Tell me which option you want (scaffold files now / different language / just explain more), and I'll do it." input field.



The screenshot shows the Visual Studio Code interface with the following details:

- Explorer:** Shows an open editor for "Assignment-6.py" and a "Welcome" file.
- Editor:** Displays the following Python code for generating a Fibonacci series:

```
1  # Optimize the python code to print fibonacci series upto n terms
2  # Generate the code to print fibonacci series upto n terms
3  def fibonacci(n):
4      a, b = 0, 1
5      for i in range(n):
6          print(a, end=" ")
7          a, b = b, a + b
8  n = int(input("Enter the number of terms: "))
9  fibonacci(n)
```
- Terminal:** Shows the command line output of the script execution. The user enters "6" and the script prints the first 6 terms of the Fibonacci sequence: 0 1 1 2 3 5. However, the output is cut off at the end.
- Output:** Shows the command line output of the script execution. The user enters "6" and the script prints the first 6 terms of the Fibonacci sequence: 0 1 1 2 3 5.
- Problems:** Shows no problems.
- Terminal:** Shows the command line output of the script execution. The user enters "6" and the script prints the first 6 terms of the Fibonacci sequence: 0 1 1 2 3 5.
- Right Panel:** Includes a "Next steps (pick one)" section with the following items:
 - I can scaffold the files (README.md, hello.py, tests/, requirements.txt t..gitignore) in your workspace now.
 - Or walk through another language (JavaScript/TypeScript, Java, etc.) if you prefer. and a "Tell me which option you want (scaffold files now / different language / just explain more), and I'll do it." input field.

VS Code interface showing AI-assisted class generation for `stude.py`. The editor displays a simplified version of the Fibonacci series code. The terminal shows the execution of the script and its output. A sidebar on the right provides next steps and a scaffold menu.

```
# Optimize the python code to print fibonacci series upto n terms
# Generate the code to print fibonacci series upto n terms
# Simplified version of the code to print fibonacci series
# Generate the fibonacci series using defined function
def fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b
    print()
fibonacci(10)
```

Terminal output:

```
Enter the number of terms: 6
0 1 1 2 3 5 8
ronithreddyrevuri@Roniths-MacBook-Air:~/Desktop/AI Asstd Coding/ASSIGNMENT-6.py %
```

Next steps (pick one):

- I can scaffold the files (`README.md`, `hello.py`, `tests/`, `requirements.txt`, `.gitignore`) in your workspace now.
- Or walk through another language (JavaScript/TypeScript, Java, etc.) if you prefer.

Tell me which option you want (scaffold files now / different language / just explain more), and I'll do it.

Add Context...
new
AI-assisted
#new
X C V D > v

VS Code interface showing AI-assisted class generation for `stude.py`. The editor displays a simplified version of the Fibonacci series code. The terminal shows the execution of the script and its output. A sidebar on the right provides next steps and a scaffold menu.

```
# Optimize the python code to print fibonacci series upto n terms
# Generate the code to print fibonacci series upto n terms
# Simplified version of the code to print fibonacci series
# Generate the fibonacci series using defined function
# Generate the fibonacci series using defined function
def fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b
    print()
n = int(input("Enter the number of terms: "))
fibonacci(10)
```

Terminal output:

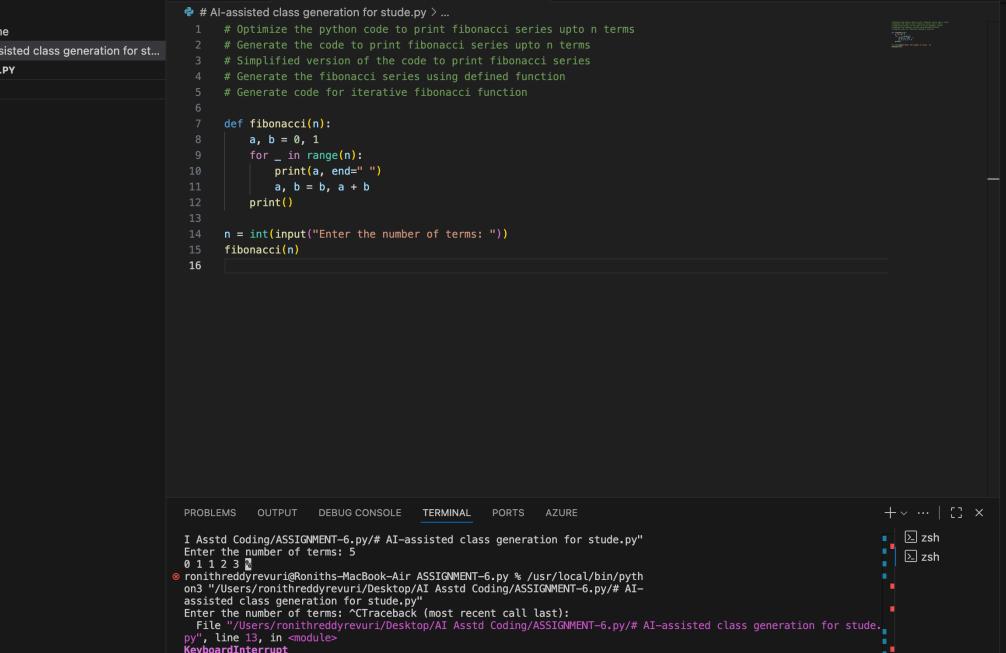
```
assisted class generation for stude.py"
ronithreddyrevuri@Roniths-MacBook-Air:~/Desktop/AI Asstd Coding/ASSIGNMENT-6.py %
```

Next steps (pick one):

- I can scaffold the files (`README.md`, `hello.py`, `tests/`, `requirements.txt`, `.gitignore`) in your workspace now.
- Or walk through another language (JavaScript/TypeScript, Java, etc.) if you prefer.

Tell me which option you want (scaffold files now / different language / just explain more), and I'll do it.

Add Context...
new
AI-assisted
#new
X C V D > v



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

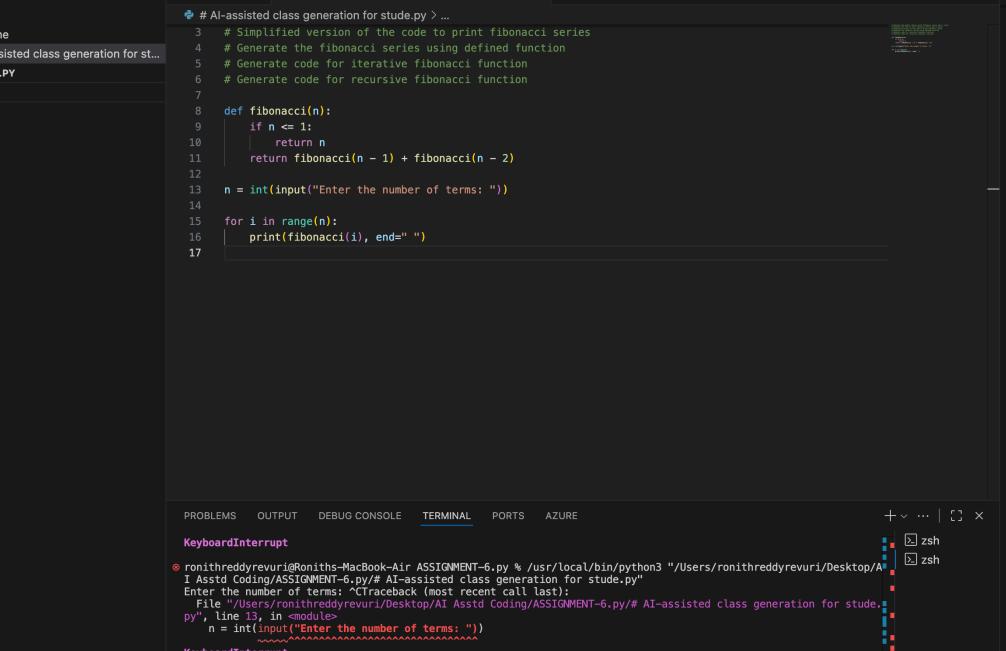
- EXPLORER** sidebar: Shows 'OPEN EDITORS' with 'Welcome' and '# AI-assisted class generation for stude.py'. Other items include 'ASSIGNMENT-6.PY', 'OUTLINE', and 'TIMELINE'.
- CODE EDITOR**: The main editor shows Python code for generating a Fibonacci series. The code is as follows:

```
# AI-assisted class generation for stude.py ...
# Optimize the python code to print fibonacci series upto n terms
# Generate the code to print fibonacci series upto n terms
# Simplified version of the code to print fibonacci series
# Generate the fibonacci series using defined function
# Generate code for iterative fibonacci function

def fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b
    print()

n = int(input("Enter the number of terms: "))
fibonacci(n)
```

- TERMINAL**: The terminal shows the execution of the code and its output. It also shows an AI-assisted traceback for a keyboard interrupt.
- OUTPUT**: The output pane shows the scaffold files created by AI, including README.md, hello.py, tests/, requirements.txt, and .gitignore.
- PROBLEMS**: The problems pane is empty.
- STATUS BAR**: Shows the current file is 'ASSIGNMENT-6.py', line 16, column 1. It also shows the Python version is 3.13.9.



The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows an open editor for 'stude.py' under the 'ASSIGNMENT-6.PY' folder.
- Code Editor:** Displays the following Python code for generating a Fibonacci series:

```
# AI-assisted class generation for stude.py > ...
# Simplified version of the code to print fibonacci series
# Generate the fibonacci series using defined function
# Generate code for iterative fibonacci function
# Generate code for recursive fibonacci function

def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)

n = int(input("Enter the number of terms: "))

for i in range(n):
    print(fibonacci(i), end=" ")
```

- Terminal:** Shows a 'KeyboardInterrupt' error and a stack trace from the terminal window.
- Bottom Status Bar:** Shows the file name 'stude.py', line 17, column 1, and the Python version 'Python 3.13.9'.
- Right Sidebar:** Lists 'Next steps (pick one)' and a list of bullet points.

TIME AND SPACE COMPLEXITY COMPARISON

ABSTRACT

THE FIBONACCI SERIES IS ACHIEVED THROUGH AN ITERATIVE AND RECURSIVE APPROACH.

THE ITERATIVE SOLUTION FOLLOWS THE LOOP MECHANISM WITH A TIME COMPLEXITY OF $O(N)$ AND SPACE COMPLEXITY OF $O(1)$. IT IS VERY FAST WITH MINIMAL MEMORY REQUIREMENT, MAKING THE SOLUTION EFFICIENT FOR LARGE NUMBERS TOO.

TO BE SPECIFIC, IN THE RECURSIVE METHOD, THE FUNCTION IS CALLED MANY TIMES. AS A RESULT, THE TIME COMPLEXITY IS $O(2^n)$ AND THE SPACE COMPLEXITY IS $O(N)$.

CONCLUSION

THE ITERATIVE METHOD IS EFFICIENT AND CAN BE USED WITH LARGER NUMBERS, WHILE THE RECURSIVE METHOD IS INEFFICIENT AND IS NOT RECOMMENDED FOR LARGER VALUES OF N .