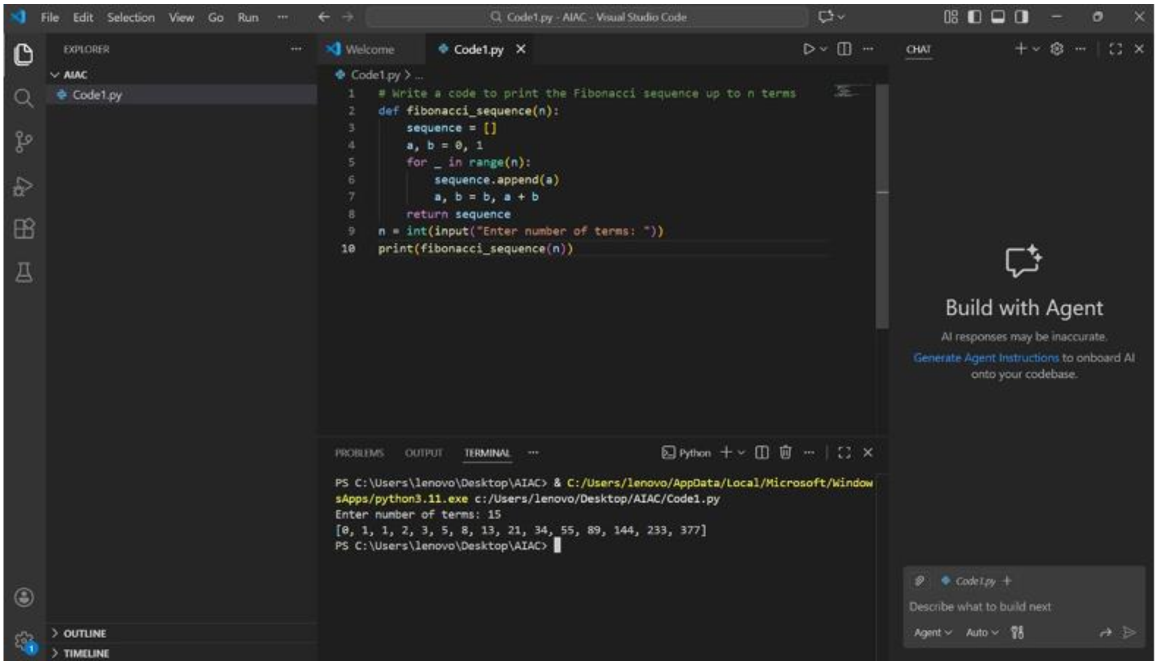
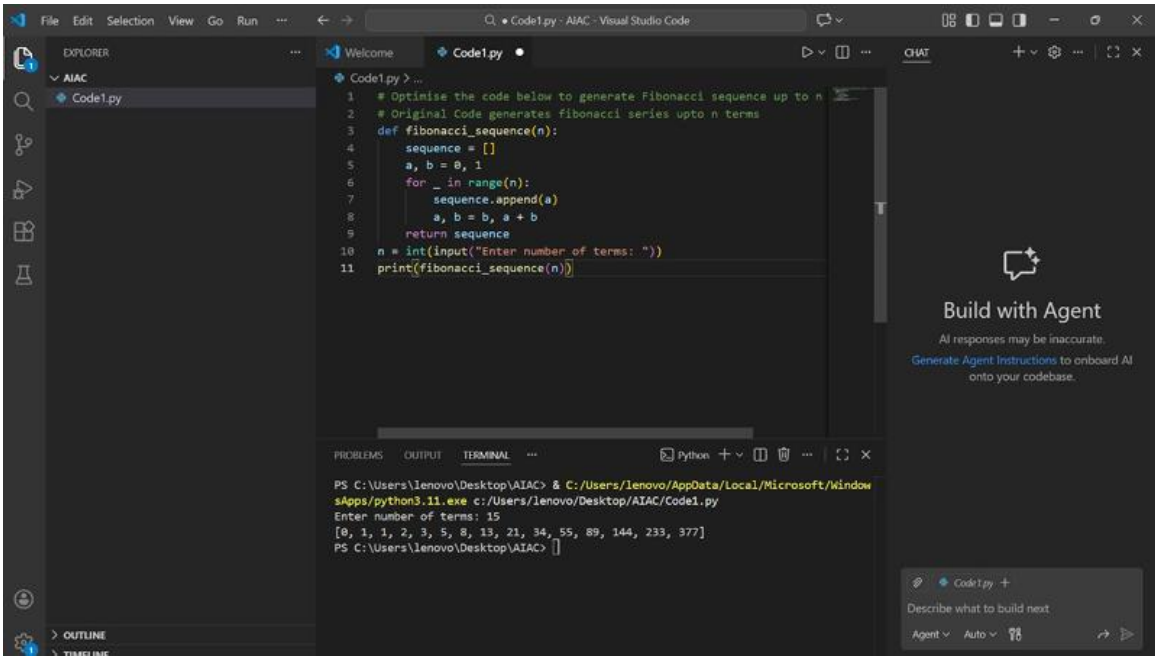
AI-ASSISTED CODING ASSIGNMENT – 1

Thumma Hasini | 2303A52076 | Batch-37


Code1.py - AIAC - Visual Studio Code
EXPLORER
AIAC
Code1.py
OUTLINE
TIMELINE
Welcome
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Describe what to build next
Agent
Auto
PROBLEMS
OUTPUT
TERMINAL
Python

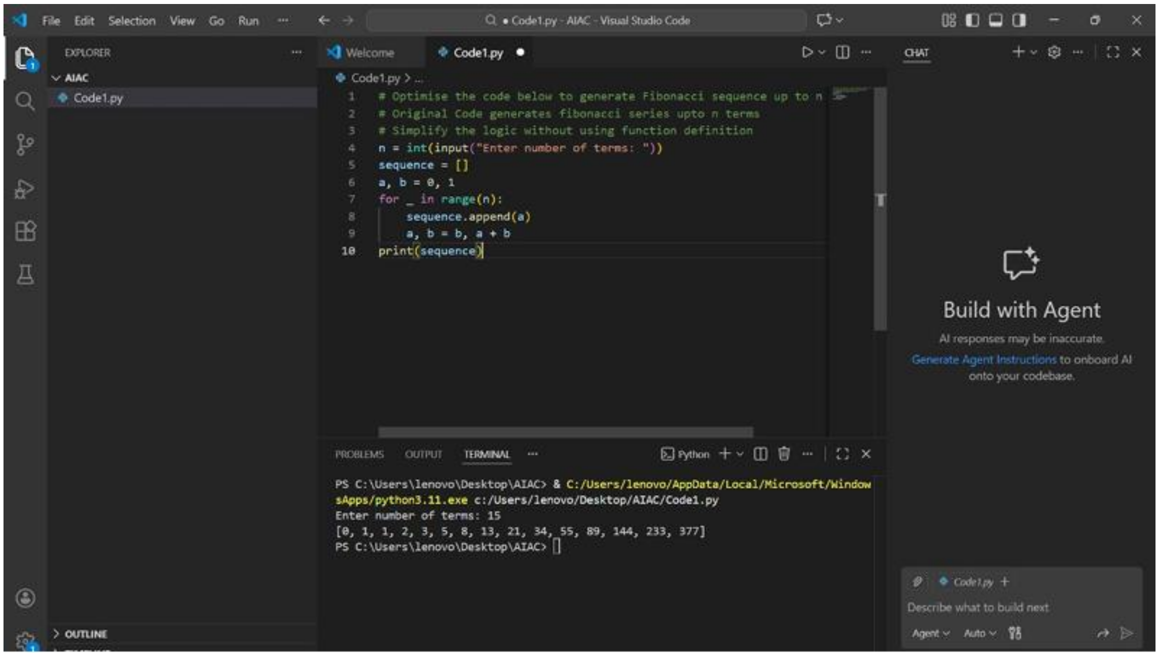
```python
# Write a code to print the Fibonacci sequence up to n terms
def fibonacci_sequence(n):
    sequence = []
    a, b = 0, 1
    for _ in range(n):
        sequence.append(a)
        a, b = b, a + b
    return sequence
n = int(input("Enter number of terms: "))
print(fibonacci_sequence(n))
```

```
PS C:\Users\lenovo\Desktop\AIAC> & C:/Users/lenovo/AppData/Local/Microsoft/Window
sApps/python3.11.exe c:/Users/lenovo/Desktop/AIAC/Code1.py
Enter number of terms: 15
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
PS C:\Users\lenovo\Desktop\AIAC>
```


● Code1.py - AIAC - Visual Studio Code
EXPLORER
AIAC
Code1.py
OUTLINE
TIMELINE
Welcome
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Describe what to build next
Agent
Auto
PROBLEMS
OUTPUT
TERMINAL
Python

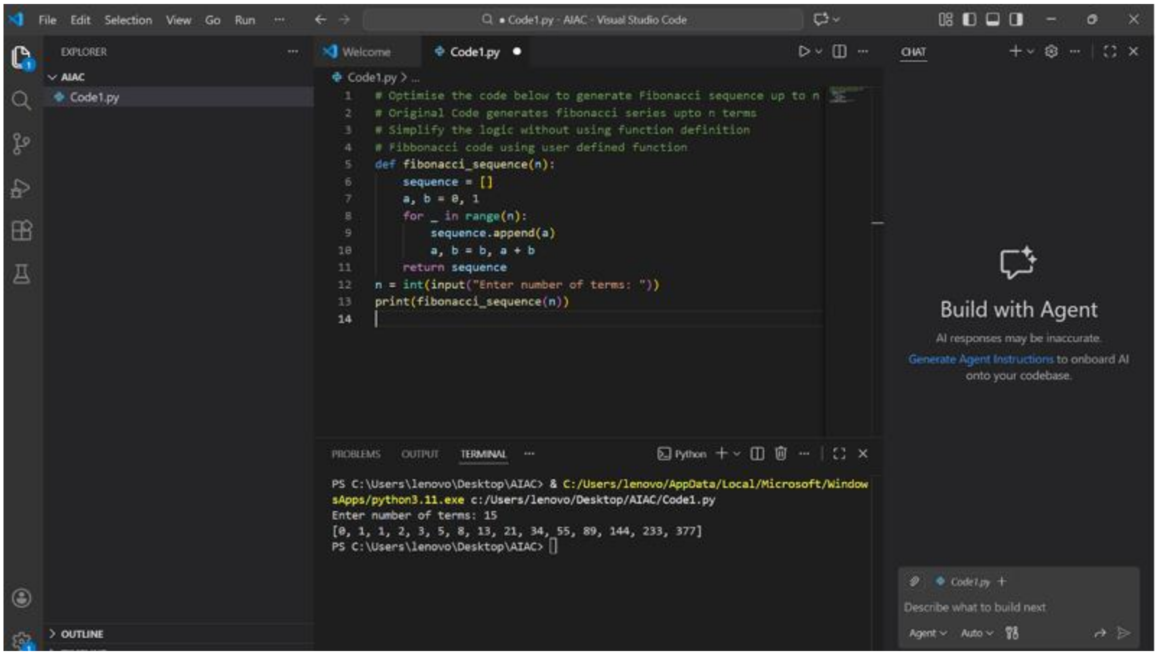
```python
# Optimise the code below to generate Fibonacci sequence up to n
# Original Code generates fibonacci series upto n terms
def fibonacci_sequence(n):
    sequence = []
    a, b = 0, 1
    for _ in range(n):
        sequence.append(a)
        a, b = b, a + b
    return sequence
n = int(input("Enter number of terms: "))
print(fibonacci_sequence(n))
```

```
PS C:\Users\lenovo\Desktop\AIAC> & C:/Users/lenovo/AppData/Local/Microsoft/Window
sApps/python3.11.exe c:/Users/lenovo/Desktop/AIAC/Code1.py
Enter number of terms: 15
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
PS C:\Users\lenovo\Desktop\AIAC>
```


File Edit Selection View Go Run ···
• Code1.py - AIAC - Visual Studio Code
EXPLORER
AIAC
Code1.py
Welcome
Code1.py
Code1.py > ...
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Code1.py
Describe what to build next
Agent
Auto
PROBLEMS
OUTPUT
TERMINAL
Python
OUTLINE

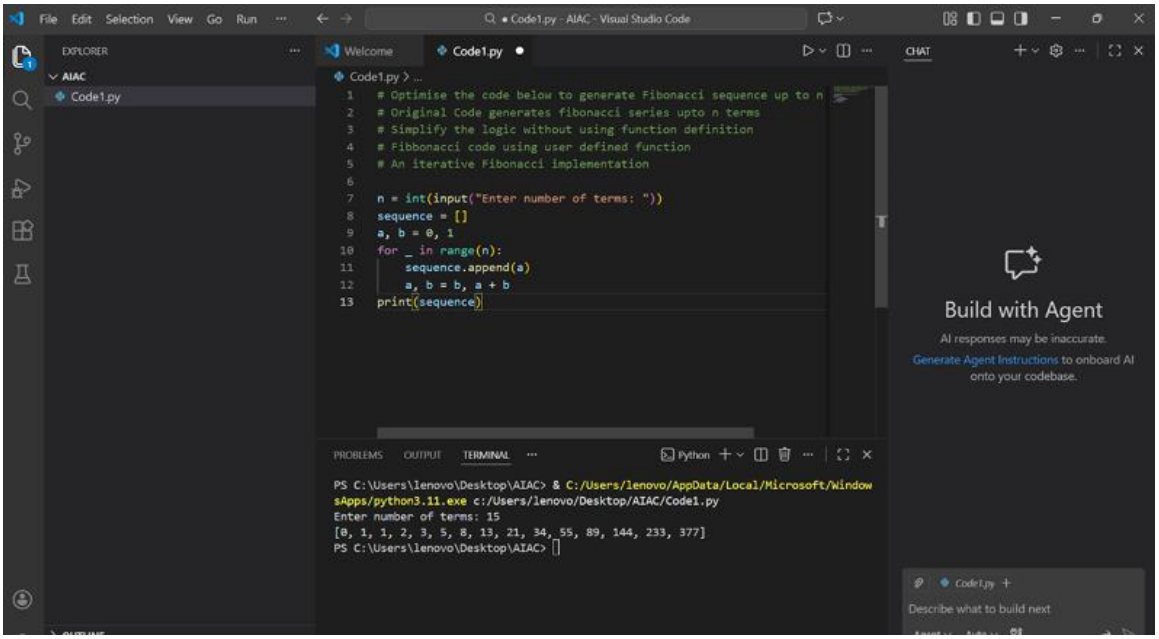
```python
# Optimise the code below to generate Fibonacci sequence up to n
# Original Code generates fibonacci series upto n terms
# Simplify the logic without using function definition
n = int(input("Enter number of terms: "))
sequence = []
a, b = 0, 1
for _ in range(n):
    sequence.append(a)
    a, b = b, a + b
print(sequence)
```

```
PS C:\Users\lenovo\Desktop\AIAC> & C:/Users/lenovo/AppData/Local/Microsoft/Window
sApps/python3.11.exe c:/Users/lenovo/Desktop/AIAC/Code1.py
Enter number of terms: 15
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
PS C:\Users\lenovo\Desktop\AIAC>
```


File Edit Selection View Go Run ···
• Code1.py - AIAC - Visual Studio Code
EXPLORER
AIAC
Code1.py
Welcome
Code1.py
Code1.py > ...
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Code1.py
Describe what to build next
Agent
Auto
PROBLEMS
OUTPUT
TERMINAL
Python
OUTLINE

```python
# Optimise the code below to generate Fibonacci sequence up to n
# Original Code generates fibonacci series upto n terms
# Simplify the logic without using function definition
# Fibbonacci code using user defined function
def fibonacci_sequence(n):
    sequence = []
    a, b = 0, 1
    for _ in range(n):
        sequence.append(a)
        a, b = b, a + b
    return sequence
n = int(input("Enter number of terms: "))
print(fibonacci_sequence(n))
```

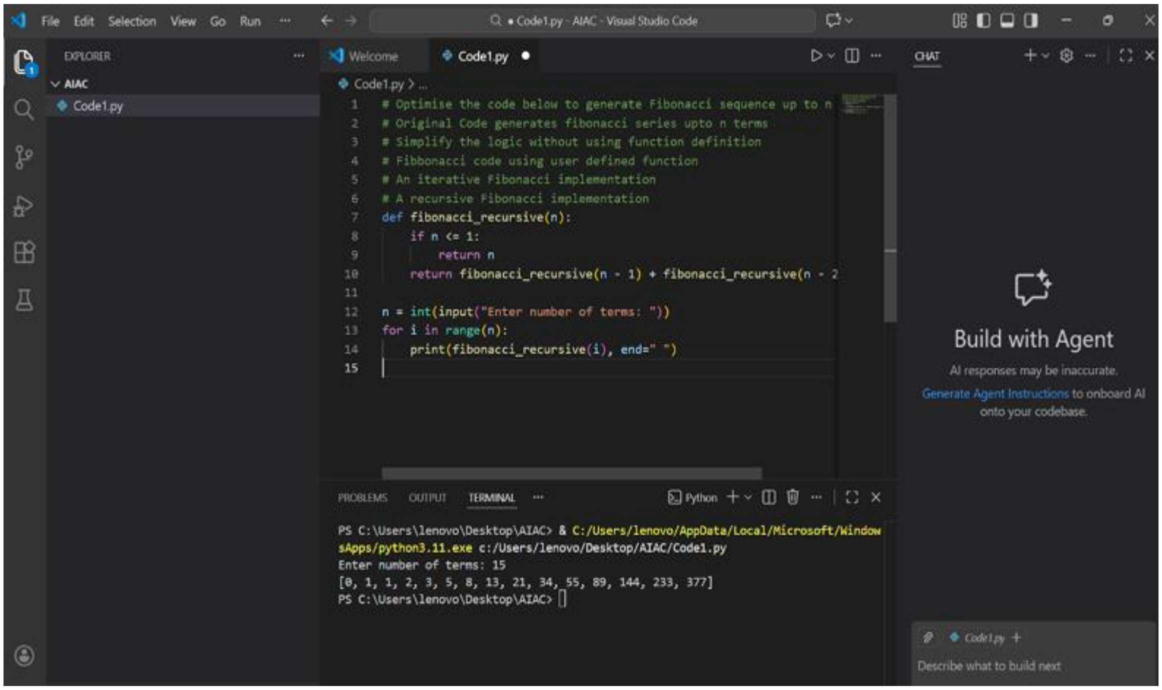
```
PS C:\Users\lenovo\Desktop\AIAC> & C:/Users/lenovo/AppData/Local/Microsoft/Window
sApps/python3.11.exe c:/Users/lenovo/Desktop/AIAC/Code1.py
Enter number of terms: 15
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
PS C:\Users\lenovo\Desktop\AIAC>
```


File Edit Selection View Go Run ···
● Code1.py - AIAC - Visual Studio Code
EXPLORER
AIAC
Code1.py
Welcome
Code1.py
Code1.py > ...
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Describe what to build next
PROBLEMS
OUTPUT
TERMINAL
Python

```python
# Optimise the code below to generate Fibonacci sequence up to n
# Original Code generates fibonacci series upto n terms
# Simplify the logic without using function definition
# Fibbonacci code using user defined function
# An iterative Fibonacci implementation

n = int(input("Enter number of terms: "))
sequence = []
a, b = 0, 1
for _ in range(n):
    sequence.append(a)
    a, b = b, a + b
print(sequence)
```

```
PS C:\Users\lenovo\Desktop\AIAC> & C:/Users/lenovo/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/lenovo/Desktop/AIAC/Code1.py
Enter number of terms: 15
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
PS C:\Users\lenovo\Desktop\AIAC>
```


File Edit Selection View Go Run ···
● Code1.py - AIAC - Visual Studio Code
EXPLORER
AIAC
Code1.py
Welcome
Code1.py
Code1.py > ...
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Describe what to build next
PROBLEMS
OUTPUT
TERMINAL
Python

```python
# Optimise the code below to generate Fibonacci sequence up to n
# Original Code generates fibonacci series upto n terms
# Simplify the logic without using function definition
# Fibbonacci code using user defined function
# An iterative Fibonacci implementation
# A recursive Fibonacci implementation
def fibonacci_recursive(n):
    if n <= 1:
        return n
    return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2

n = int(input("Enter number of terms: "))
for i in range(n):
    print(fibonacci_recursive(i), end=" ")

```

```
PS C:\Users\lenovo\Desktop\AIAC> & C:/Users/lenovo/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/lenovo/Desktop/AIAC/Code1.py
Enter number of terms: 15
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
PS C:\Users\lenovo\Desktop\AIAC>
```

Time and Space Complexity Comparison

Approach	Time Complexity	Space Complexity
Iterative Fibonacci	O(n)	O(1)
Recursive Fibonacci	O(2^n)	O(n)

Performance for Large n

Aspect	Iterative	Recursive
Execution speed	Very fast	Very slow
Memory usage	Minimal	High (call stack)
Scalability	Excellent	Poor
Risk of crash	None	Stack overflow

Conclusion

- Iterative Fibonacci works efficiently even for large values like n = 10000.
- Recursive Fibonacci becomes extremely slow and may crash for values above n = 40.