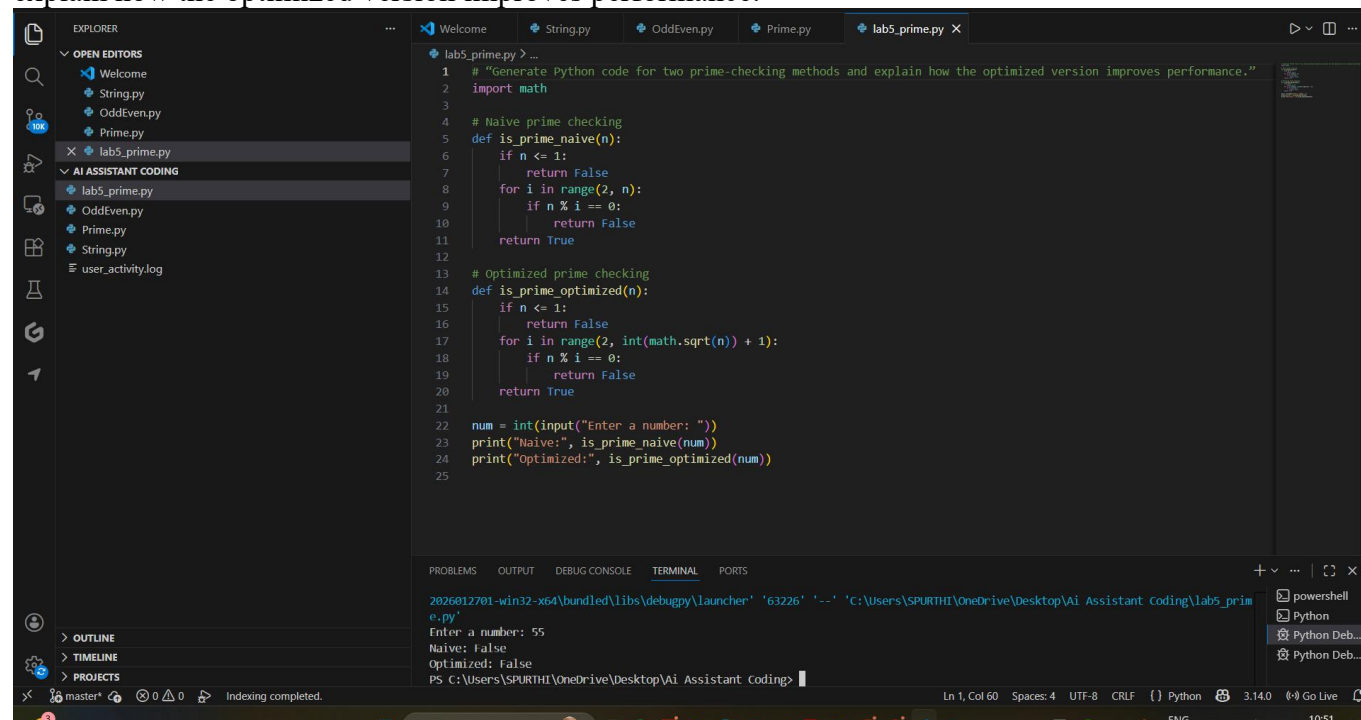


**AIAC LAB 5.5**  
**B.SPURTHI**  
**2303A51802**  
**Batch 29**

### Task Description #1 (Transparency in Algorithm Optimization)

Prompt:

“Generate Python code for two prime-checking methods and explain how the optimized version improves performance.”



The screenshot shows a Visual Studio Code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'lab5\_prime.py' with several files: 'Welcome', 'String.py', 'OddEven.py', 'Prime.py', 'lab5\_prime.py', 'AI ASSISTANT CODING', 'lab5\_prime.py', 'OddEven.py', 'Prime.py', 'String.py', and 'user\_activity.log'. The main editor window displays the code for 'lab5\_prime.py'.

```
1 # "Generate Python code for two prime-checking methods and explain how the optimized version improves performance."
2 import math
3
4 # Naive prime checking
5 def is_prime_naive(n):
6     if n <= 1:
7         return False
8     for i in range(2, n):
9         if n % i == 0:
10            return False
11    return True
12
13 # Optimized prime checking
14 def is_prime_optimized(n):
15     if n <= 1:
16         return False
17     for i in range(2, int(math.sqrt(n)) + 1):
18         if n % i == 0:
19             return False
20    return True
21
22 num = int(input("Enter a number: "))
23 print("Naive:", is_prime_naive(num))
24 print("Optimized:", is_prime_optimized(num))
25
```

The terminal output shows the execution of the code:

```
2026012701-win32-x64\bundle\libs\debugpy\launcher '63226' '-' 'C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\lab5_prime.py'
Enter a number: 55
Naive: false
Optimized: false
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

### Explanation & Time Complexity

#### Naive approach:

Checks all numbers from 2 to  $n-1$

**Time Complexity:**  $O(n)$

#### Optimized approach:

Checks only up to  $\sqrt{n}$

**Time Complexity:**  $O(\sqrt{n})$

### Efficiency Comparison

Optimized version significantly reduces iterations

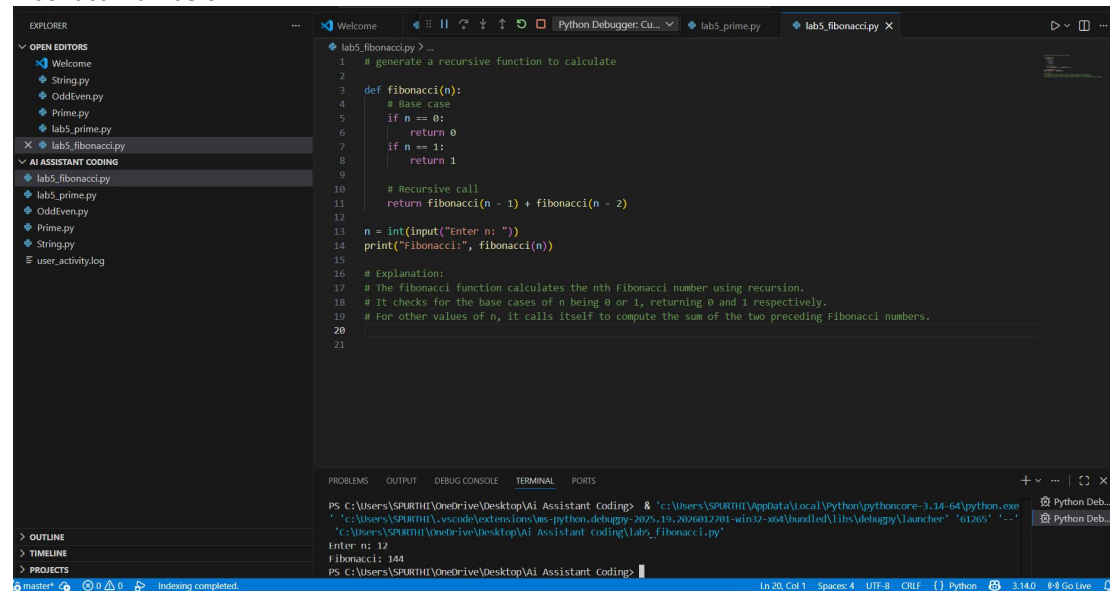
Improves performance for large numbers

## Demonstrates transparency in algorithm design

### Task Description #2 (Transparency in Recursive Algorithms)

#### Prompt:

generate a recursive function to calculate Fibonacci numbers.



```
1 # generate a recursive function to calculate
2
3 def fibonacci(n):
4     # Base case
5     if n == 0:
6         return 0
7     if n == 1:
8         return 1
9
10    # Recursive call
11    return fibonacci(n - 1) + fibonacci(n - 2)
12
13 n = int(input("Enter n: "))
14 print("Fibonacci:", fibonacci(n))
15
16 # Explanation:
17 # The fibonacci function calculates the nth Fibonacci number using recursion.
18 # It checks for the base cases of n being 0 or 1, returning 0 and 1 respectively.
19 # For other values of n, it calls itself to compute the sum of the two preceding Fibonacci numbers.
20
21
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding> & 'c:\Users\SPURTHI\AppData\Local\Python\pythoncore-3.14-64\python.exe'
'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundle\libs\debugpy\launcher' '61265' '-'
'c:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding\lab5_fibonacci.py'
Enter n: 12
Fibonacci: 144
PS C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding>
```

## Explanation

**Base cases:** Stop recursion at  $n = 0$  and  $n = 1$

**Recursive calls:**

`fibonacci(n-1)` and `fibonacci(n-2)`

Each function waits for results from smaller subproblems

Execution matches explanation step-by-step

### Task Description #3 (Transparency in Error Handling)

#### Prompt:

“Generate code with proper error handling and clear explanations

for each exception.”

```
def read_file(filename):
    try:
        with open(filename, "r") as file:
            print(file.read())
    except FileNotFoundError:
        print("Error: File not found")
    except PermissionError:
        print("Error: Permission denied")
    except Exception as e:
        print("Unexpected error:", e)

read_file("sample.txt")
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding> c:\cd 'c:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding'; & 'c:\Users\SPURTHI\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\SPURTHI\vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher' '51330' '-' 'c:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding\lab5_filehandling.py'
Error: File not found
PS C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding>
```

```
# explanation:
# The read_file function attempts to open and read the contents of a file specified by
# filename.
# It uses a try-except block to handle potential errors:
# 1. FileNotFoundError: Caught if the specified file does not exist, printing an appropriate
# error message.
# 2. PermissionError: Caught if there are insufficient permissions to read the file, printing
# an appropriate error message.
# 3. A general Exception catch to handle any other unexpected errors, printing the error
# message.
# Finally, the function is called with "sample.txt" as an argument to demonstrate its
# functionality.
```

## Task Description #4 (Security in User Authentication)

generate a Python-based login system. Check whether the AI uses secure password handling practices.

```
import hashlib

users = {
    "admin": hashlib.sha256("1234".encode()).hexdigest()
}

username = input("Username: ").strip()
password = input("Password: ").strip()

hashed = hashlib.sha256(password.encode()).hexdigest()

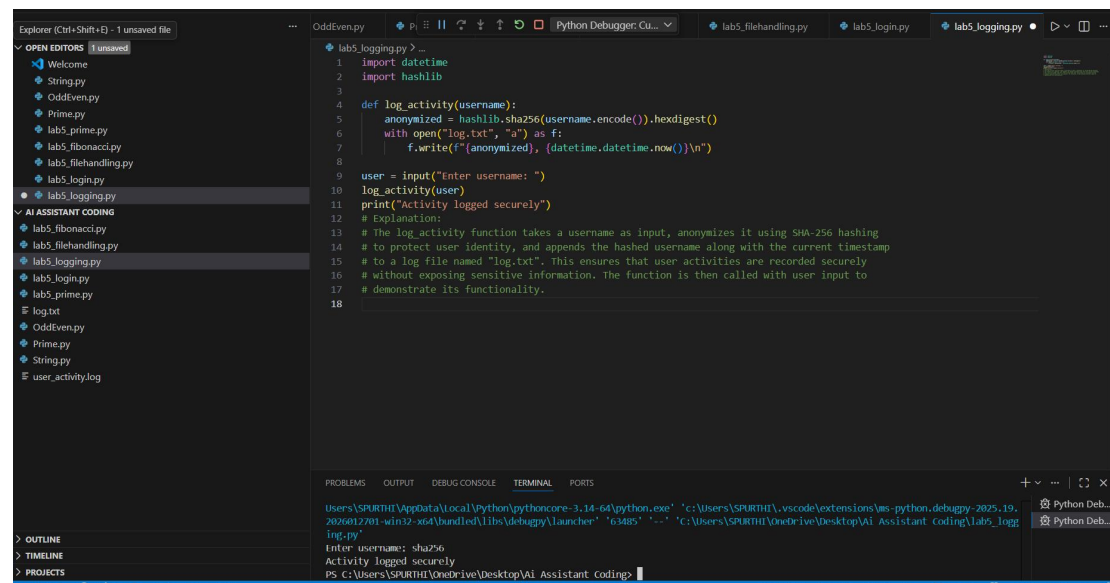
if username in users and users[username] == hashed:
    print("Login successful")
else:
    print("Login failed")
```

```
2026012701-win32-x64\bundled\libs\debugpy\launcher' '56795' '-' 'c:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding\lab5_login.py'
Username: spurthi
Password: 1234
Login failed
PS C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding>
```

```
# Explanation:
# This code implements a simple login system using hashed passwords for security.
# It defines a dictionary of users with their usernames and SHA-256 hashed passwords.
# The user is prompted to enter their username and password.
# The entered password is hashed and compared to the stored hash for the given username.
# If the hashes match, a success message is printed; otherwise, a failure message is shown.
```

## Task Description #5 (Privacy in Data Logging)

Generate a Python script that logs user activity (username, IP address, timestamp). & Examine whether sensitive data is logged unnecessarily or insecurely.



```
lab5_logging.py > ...
1 import datetime
2 import hashlib
3
4 def log_activity(username):
5     anonymized = hashlib.sha256(username.encode()).hexdigest()
6     with open("log.txt", "a") as f:
7         f.write(f"{anonymized}, {datetime.datetime.now()}\n")
8
9 user = input("Enter username: ")
10 log_activity(user)
11 print("Activity logged securely")
12 # Explanation:
13 # The log_activity function takes a username as input, anonymizes it using SHA-256 hashing
14 # to protect user identity, and appends the hashed username along with the current timestamp
15 # to a log file named "log.txt". This ensures that user activities are recorded securely
16 # without exposing sensitive information. The function is then called with user input to
17 # demonstrate its functionality.
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Users\SPURTHI\AppData\Local\Python\pythoncore-3.14-64\python.exe 'c:\Users\SPURTHI\vscode\extensions\ms-python-debugpy-2025.19.2026012701-win32-x64\bundle\libs\debugpy\launcher' '63485' '-' 'c:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding\lab5\_logging.py'

Enter username: sha256  
Activity logged securely  
PS C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding>