

## AIAC LAB 5.5

B.SPURTHI

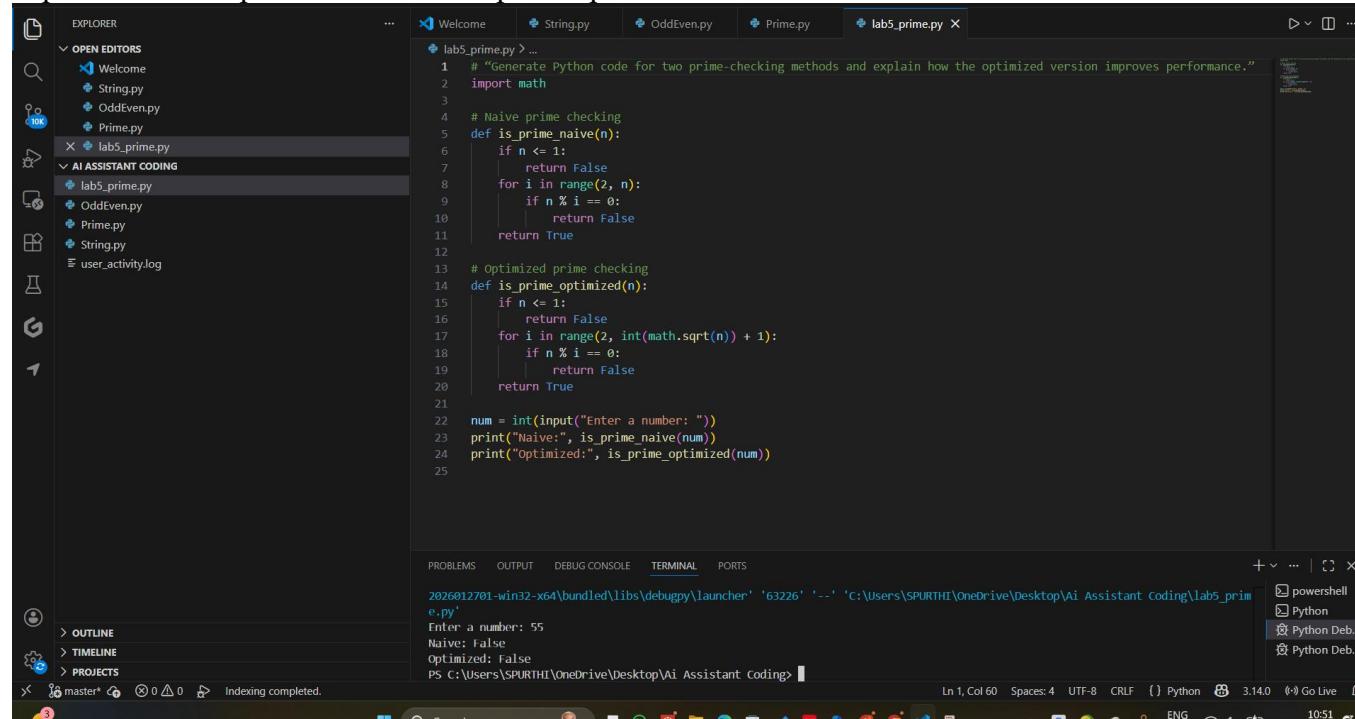
2303A51802

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 the Visual Studio Code interface. The Explorer sidebar on the left lists files: Welcome, String.py, OddEven.py, Prime.py, and lab5\_prime.py (which is currently selected). The AI ASSISTANT CODING section also lists lab5\_prime.py, OddEven.py, Prime.py, and String.py. The terminal at the bottom shows the execution of the code and its output.

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
2026012701-win32-x64\bundled\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

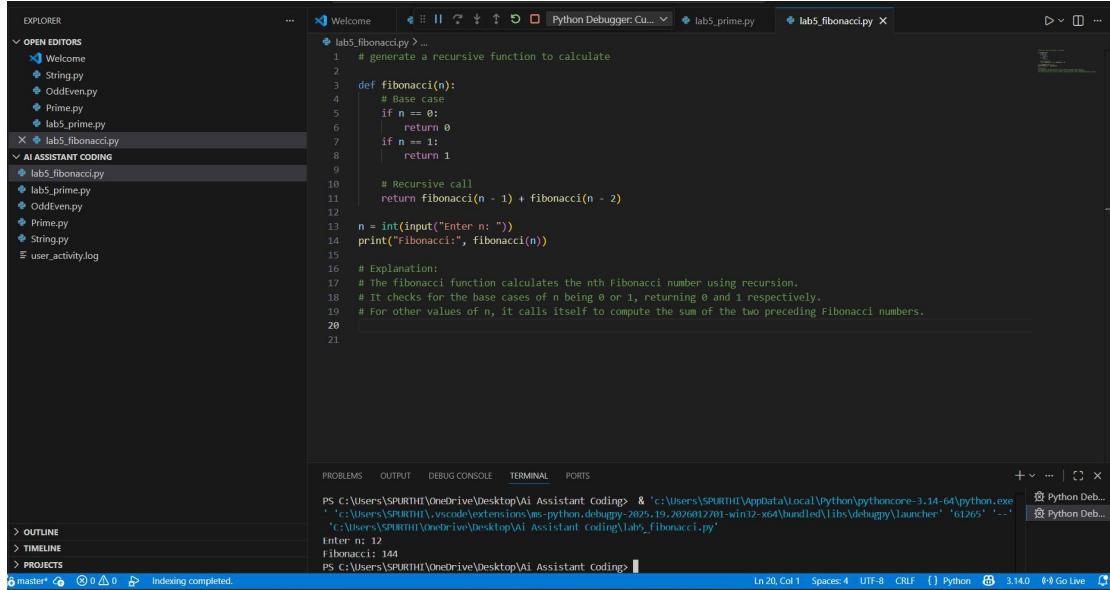
Improves performance for large numbers

## Demonstrates transparency in algorithm design

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

#### Prompt:

generate a recursive function to calculate Fibonacci numbers.



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

- EXPLORER:** Shows files: Welcome, String.py, OddEven.py, Prime.py, lab5\_prime.py, lab5\_fibonacci.py (selected), AI ASSISTANT CODING, lab5\_fibonacci (disabled), lab5\_prime (disabled), OddEven (disabled), Prime (disabled), String (disabled), user\_activity.log.
- EDITOR:** Displays the code for `lab5_fibonacci.py`. The code is as follows:

```
# generate a recursive function to calculate
# Base case
if n == 0:
    return 0
if n == 1:
    return 1
# Recursive call
return fibonacci(n - 1) + fibonacci(n - 2)

n = int(input("Enter n: "))
print("Fibonacci:", fibonacci(n))

# Explanation:
# The fibonacci function calculates the nth Fibonacci number using recursion.
# It checks for the base cases of n being 0 or 1, returning 0 and 1 respectively.
# For other values of n, it calls itself to compute the sum of the two preceding Fibonacci numbers.
```

The terminal below shows the output of running the script:

```
PS C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding> python lab5_fibonacci.py
Enter n: 12
Fibonacci: 144
PS C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding>
```

Bottom status bar: In 20, Col 1 | Spaces: 4 | UTF-8 | CRLF | Python | 3.14.0 | Go Live |

## 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.”

```

OPEN EDITORS
lab5_filehandling.py > ...
Welcome
String.py
OddEven.py
Prime.py
lab5_prime.py
lab5_fibonacci.py
lab5_filehandling.py
AI ASSISTANT CODING
lab5_fibonacci.py
lab5_filehandling.py
lab5_prime.py
OddEven.py
Prime.py
String.py
user_activity.log

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding> cd 'C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding' & 'C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding\lab5_filehandling.py'
2026012701-win32-x64\bundled\libs\debug\my\launcher' '56795' '--' 'C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding\extensions\vs-python\debugpy-2025.19.2026012701-win32-x64\bundled\libs\debug\my\launcher' '53340' '--' '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.

```

OPEN EDITORS
lab5_login.py > ...
String.py
lab5_fibonacci.py
lab5_filehandling.py
lab5_login.py
lab5_prime.py
OddEven.py
Prime.py
String.py
user_activity.log

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Open folder in new window (ctrl + click)
2026012701-win32-x64\bundled\libs\debug\my\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>

```

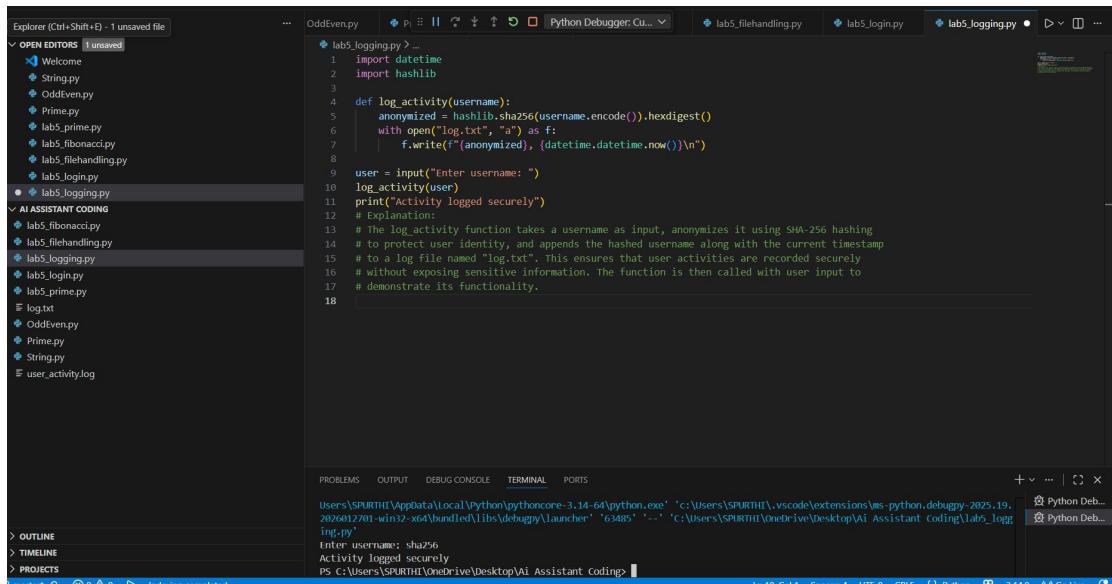
```

# 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.



```

import datetime
import hashlib

def log_activity(username):
    anonymized = hashlib.sha256(username.encode()).hexdigest()
    with open("log.txt", "a") as f:
        f.write(f"[{anonymized}], {datetime.datetime.now()}\n")

user = input("Enter username: ")
log_activity(user)
print("Activity logged securely")

```

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

- Explorer View:** Shows files like OddEven.py, String.py, OddEven.py, Prime.py, lab5\_prime.py, lab5\_fibonacci.py, lab5\_filehandling.py, lab5\_login.py, and lab5\_logging.py.
- Editor View:** The lab5\_logging.py file is open, displaying the provided code.
- Terminal View:** Shows the command `python lab5\_logging.py` being run, and the output:
 

```

C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding> python lab5_logging.py
Enter username: sha256
Activity logged securely
C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding>
      
```
- Bottom Status Bar:** Shows the current file path as C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding.