ASSIGNMENT-10.4

Name:N.Soumya

H NO:2303A51869

Batch-13

Task 1: AI-Assisted Syntax and Code Quality Review

Scenario

You join a development team and are asked to review a junior developer's Python script that fails to run correctly due to basic coding mistakes. Before deployment, the code must be corrected and standardized.

Task Description

You are given a Python script containing:

• Syntax errors

• Indentation issues

• Incorrect variable names

• Faulty function calls

Use an AI tool (GitHub Copilot / Cursor AI) to:

• Identify all syntactic and structural errors

• Correct them systematically

• Generate an explanation of each fix made

Expected Outcome

• Fully corrected and executable Python code

• AI-generated explanation describing:

o Syntax fixes

o Naming corrections

o Structural improvements

• Clean, readable version of the script

#ERROR CODE

```python
def factorial(num)
if num < 0
    print("Negative number not allowed")
else
    fact = 1
    for i in range(1, num+1)
    fact = fact * i
return fact

Number = 5
result = Factorial(Number)
print("Fa factorial result)
```

```
                                ^
SyntaxError: expected ':'
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:\Users\neera\AppData\Local\Python\pythoncore-3.14-64\python.exe c:/Users/n
eera/OneDrive/Desktop/assignments/assss.py
  File "c:\Users\neera\OneDrive\Desktop\assignments\assss.py", line 1
    def factorial(num)
                      ^
SyntaxError: expected ':'
PS C:\Users\neera\OneDrive\Desktop\assignments>
```

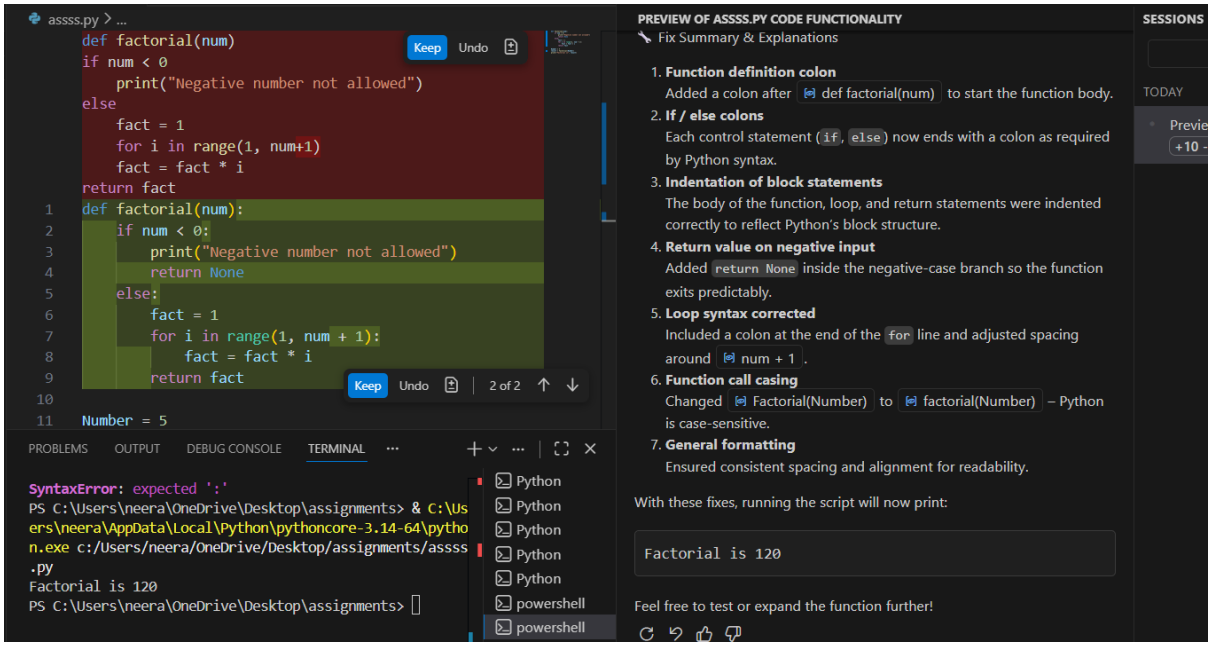
#CORRECTED CODE:

```python
def factorial(num):
    if num < 0:
        print("Negative number not allowed")
        return None
    else:
        fact = 1
        for i in range(1, num + 1):
            fact = fact * i
        return fact

number = 
result = factorial(number)

if result is not None:
    print("Factorial is", result)
```

(function) def factorial(num: Any) -> (int | None)

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE
                                ^
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:\Users\neera\AppData\Local\Python\pythoncore-3.14-64\python.exe c:/Users/n
eera/OneDrive/Desktop/assignments/assss.py
Factorial is 120
PS C:\Users\neera\OneDrive\Desktop\assignments>
```

#PREVIEW OF AI



Task 2: Performance-Oriented Code Review

Scenario

A data processing function works correctly but is inefficient and slows down the system when large datasets are used.

Task Description

You are provided with a function that identifies duplicate values in a list using inefficient nested loops.

Using AI-assisted code review:

- Analyze the logic for performance bottlenecks
- Refactor the code for better time complexity
- Preserve the correctness of the output

Ask the AI to explain:

- Why the original approach was inefficient
- How the optimized version improves performance

Expected Outcome

- Optimized duplicate-detection logic (e.g., using sets or hash-based structures)
- Improved time complexity

• AI explanation of performance improvement

• Clean, readable implementation

#ORIGINAL CODE

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


number = 35
result = fibonacci(number)
print("Fibonacci number is:", result)

```

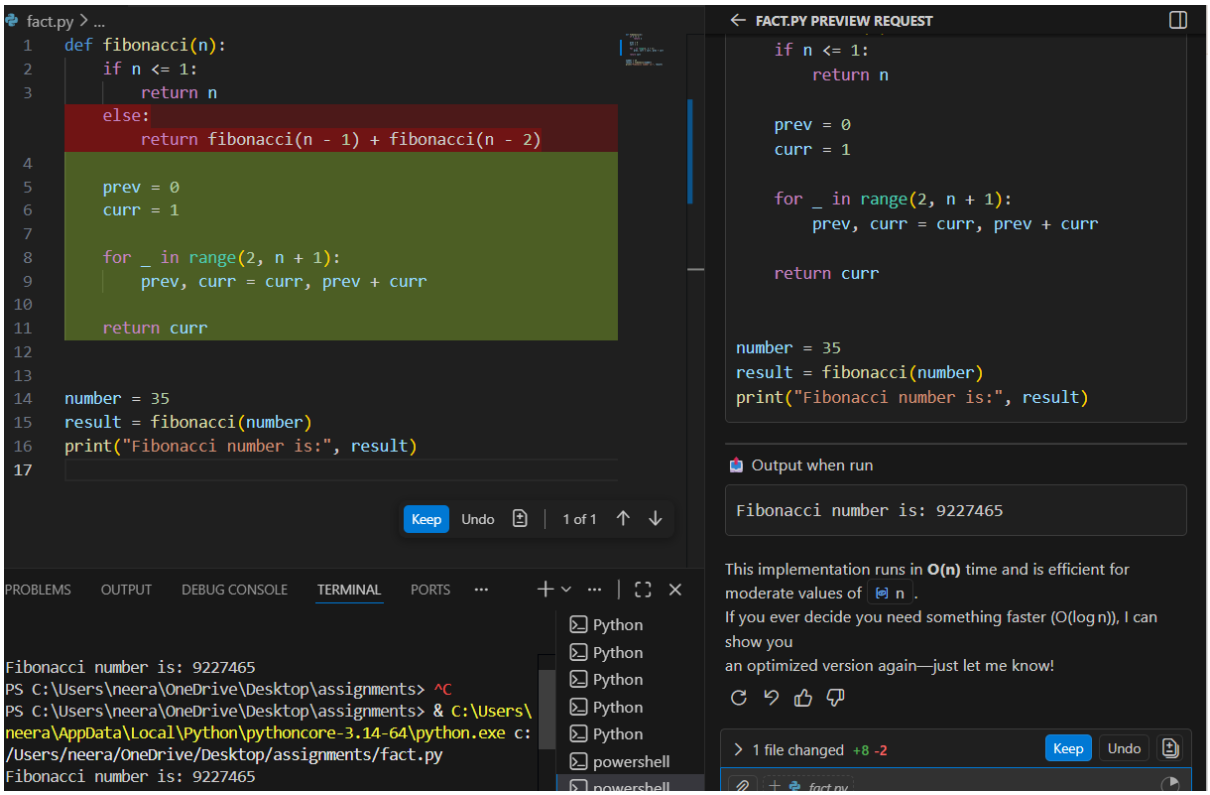
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE

ct.py
ct.py
Fibonacci number is: 9227465
PS C:\Users\neera\OneDrive\Desktop\assignments>
```

#OPTIMIZED VERSION

```python
def fibonacci(n):
    if n <= 1:
        return n
    prev = 0
    curr = 1
    for _ in range(2, n + 1):
        prev, curr = curr, prev + curr
    return curr
number = 35
result = fibonacci(number)
print("Fibonacci number is:", result)

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE

                                                    ^C
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:\Users\neera\AppData\Local\Python\pythoncore-3.14-64\python.exe
c:/Users/neera/OneDrive/Desktop/assignments/fact.py
Fibonacci number is: 9227465
```

#PREVIEW OF AI



Task 3: Readability and Maintainability Refactoring

Scenario

A working script exists in a project, but it is difficult to understand due to poor naming, formatting, and structure. The team wants it rewritten for long-term maintainability.

Task Description

You are given a poorly structured Python function with:

- Cryptic function names
- Poor indentation
- Unclear variable naming
- No documentation

Use AI-assisted review to:

- Refactor the code for clarity

• Apply PEP 8 formatting standards

• Improve naming conventions

• Add meaningful documentation

Expected Outcome

• Clean, well-structured code

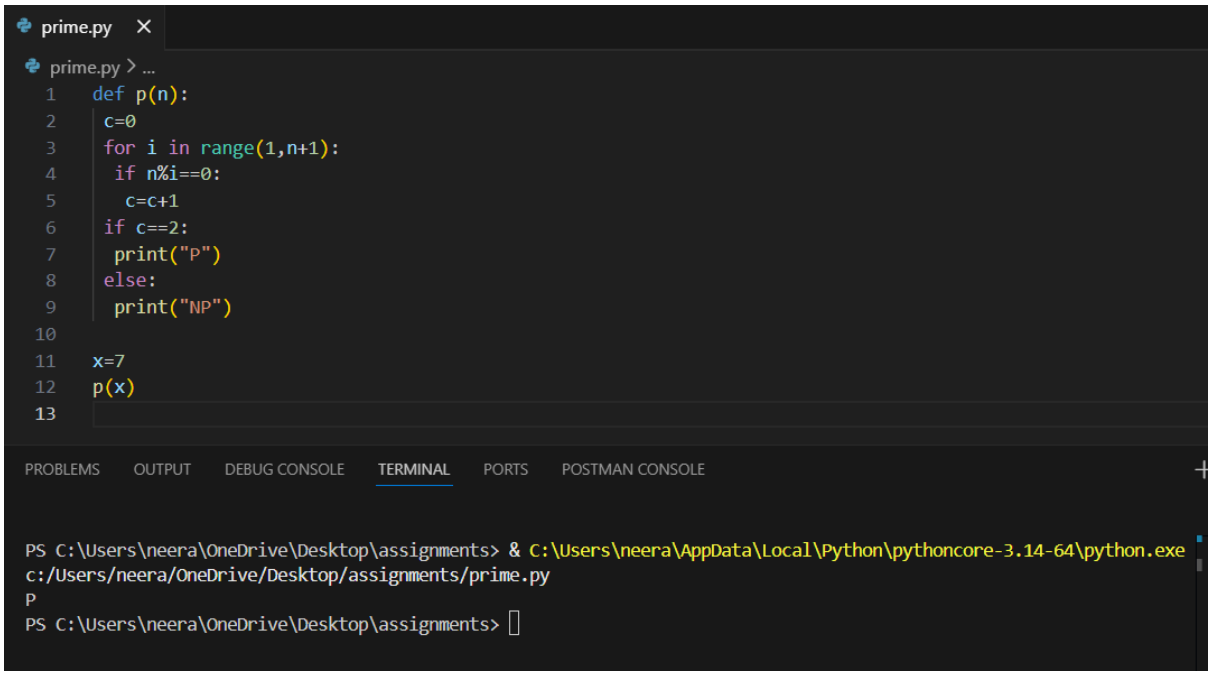
• Descriptive function and variable names

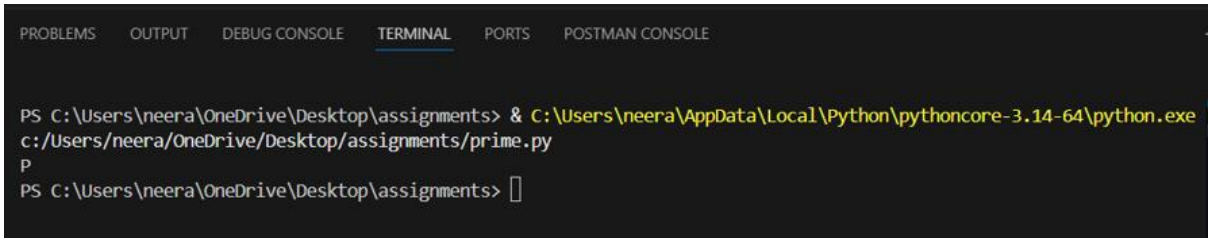
• Proper indentation and formatting

• Docstrings explaining the function purpose

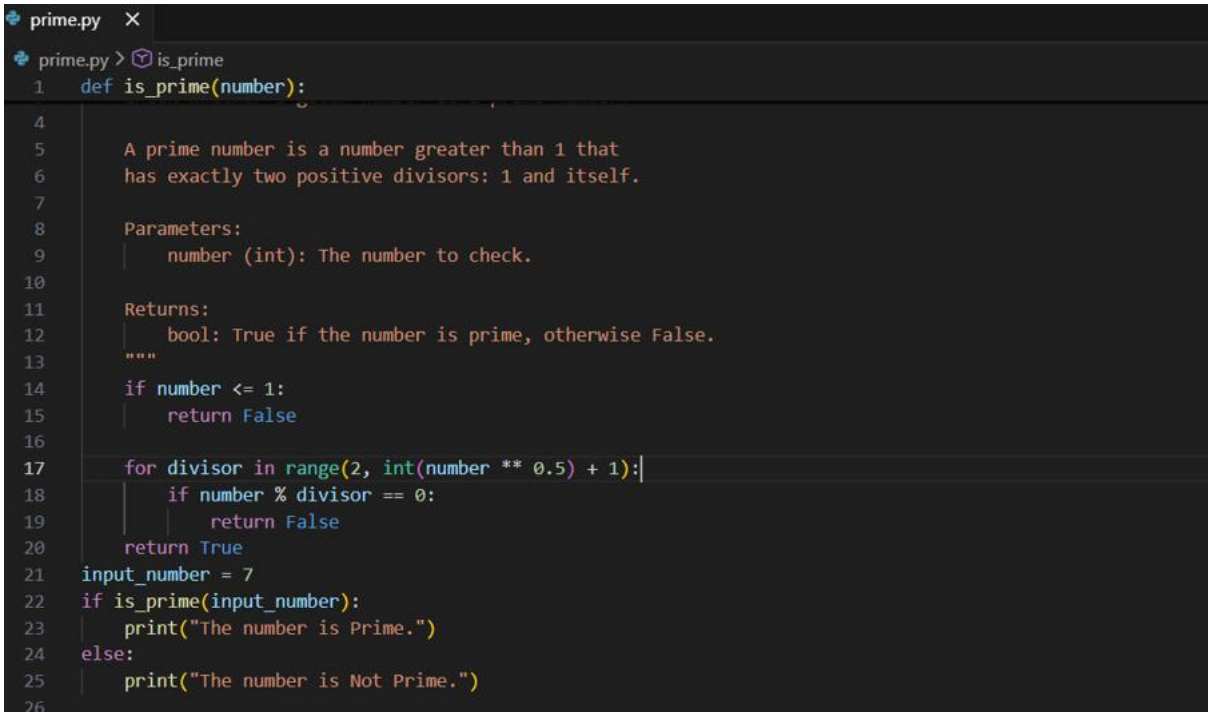
• AI explanation of readability improvements

#ORIGINAL CODE

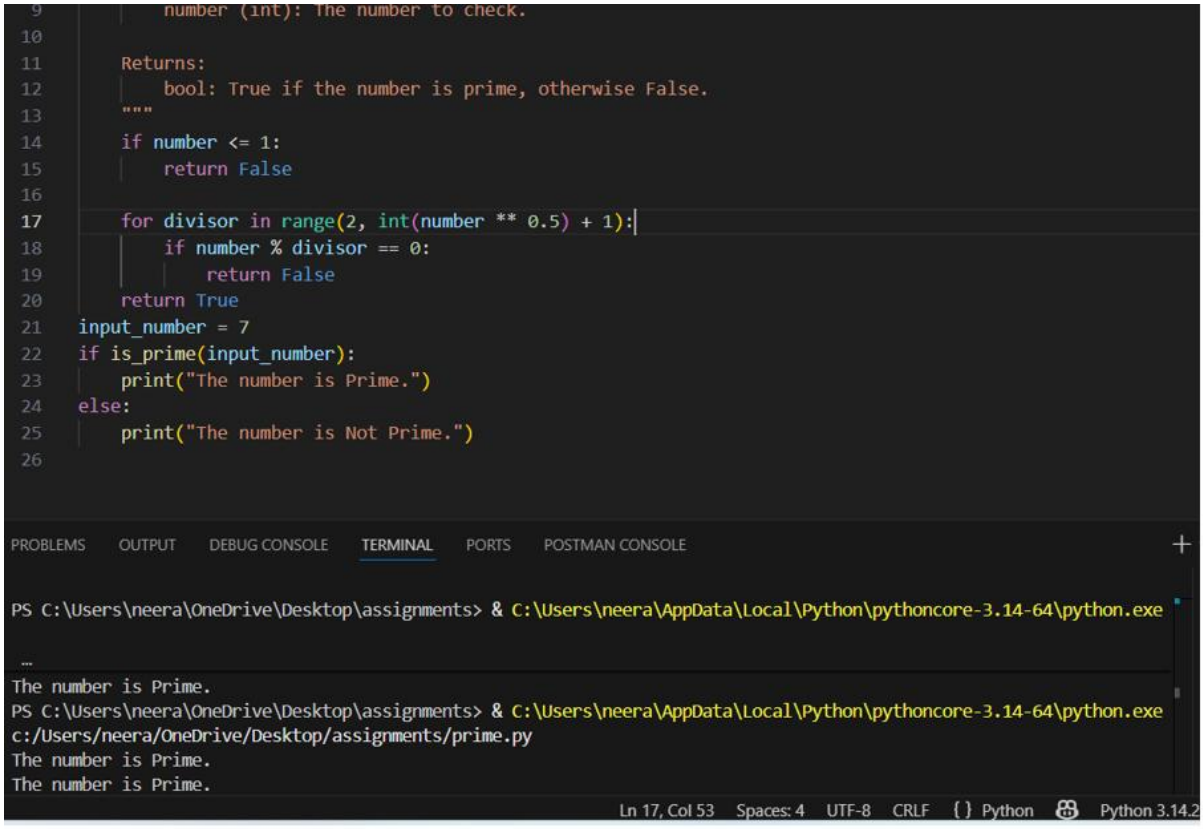




#OPTIMIZED CODE



```python
def is_prime(number):

    A prime number is a number greater than 1 that
    has exactly two positive divisors: 1 and itself.

    Parameters:
        number (int): The number to check.

    Returns:
        bool: True if the number is prime, otherwise False.
    """
    if number <= 1:
        return False

    for divisor in range(2, int(number ** 0.5) + 1):
        if number % divisor == 0:
            return False
    return True
input_number = 7
if is_prime(input_number):
    print("The number is Prime.")
else:
    print("The number is Not Prime.")
```

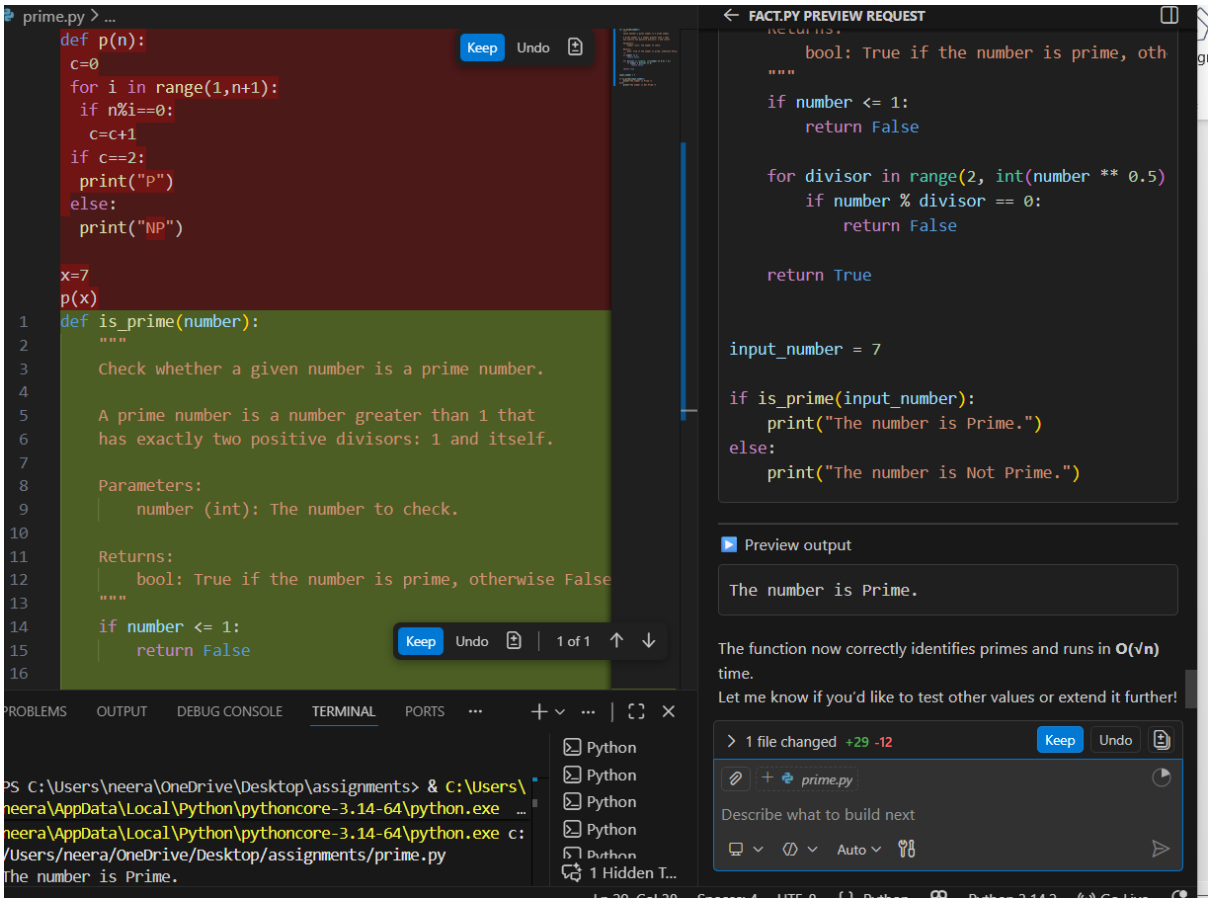


```python
        number (int): The number to check.

    Returns:
        bool: True if the number is prime, otherwise False.
    """
    if number <= 1:
        return False

    for divisor in range(2, int(number ** 0.5) + 1):
        if number % divisor == 0:
            return False
    return True
input_number = 7
if is_prime(input_number):
    print("The number is Prime.")
else:
    print("The number is Not Prime.")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE

PS C:\Users\neera\OneDrive\Desktop\assignments> & C:\Users\neera\AppData\Local\Python\pythoncore-3.14-64\python.exe
...
The number is Prime.
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:\Users\neera\AppData\Local\Python\pythoncore-3.14-64\python.exe
c:/Users/neera/OneDrive/Desktop/assignments/prime.py
The number is Prime.
The number is Prime.
```

Ln 17, Col 53 Spaces: 4 UTF-8 CRLF { } Python Python 3.14.2

#PREVIEW OF AI



Task 4: Secure Coding and Reliability Review

Scenario

A backend function retrieves user data from a database but has security vulnerabilities and poor error handling, making it unsafe for production deployment.

Task Description

You are given a Python script that:

- Uses unsafe SQL query construction
- Has no input validation
- Lacks exception handling

Use AI tools to:

- Identify security vulnerabilities

- Refactor the code using safe coding practices
- Add proper exception handling
- Improve robustness and reliability

Expected Outcome

- Secure SQL queries using parameterized statements
- Input validation logic
- Try-except blocks for runtime safety
- AI-generated explanation of security improvements
- Production-ready code structure

#ERROR CODE

SQL.py

```python
import sqlite3

def get_user(username):
    conn = sqlite3.connect("users.db")
    cursor = conn.cursor()

    query = "SELECT * FROM users WHERE username = '" + username + "'"
    cursor.execute(query)

    result = cursor.fetchall()
    conn.close()

    return result


user_input = input("Enter username: ")
data = get_user(user_input)
print(data)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE

PS C:\Users\neera\OneDrive\Desktop\assignments> & C:\Users\neera\AppData\Local\Python\pythoncore-3.14-64\python.exe
Enter username: Soumya
Traceback (most recent call last):
  File "c:\Users\neera\OneDrive\Desktop\assignments\SQL.py", line 17, in <module>
    data = get_user(user_input)
  File "c:\Users\neera\OneDrive\Desktop\assignments\SQL.py", line 8, in get_user
    cursor.execute(query)
    ~~~~~~~~~~~~~~^^^^^^^
sqlite3.OperationalError: no such table: users
PS C:\Users\neera\OneDrive\Desktop\assignments>
```

#OPTIMIZED CODE:

SQL.py > ...

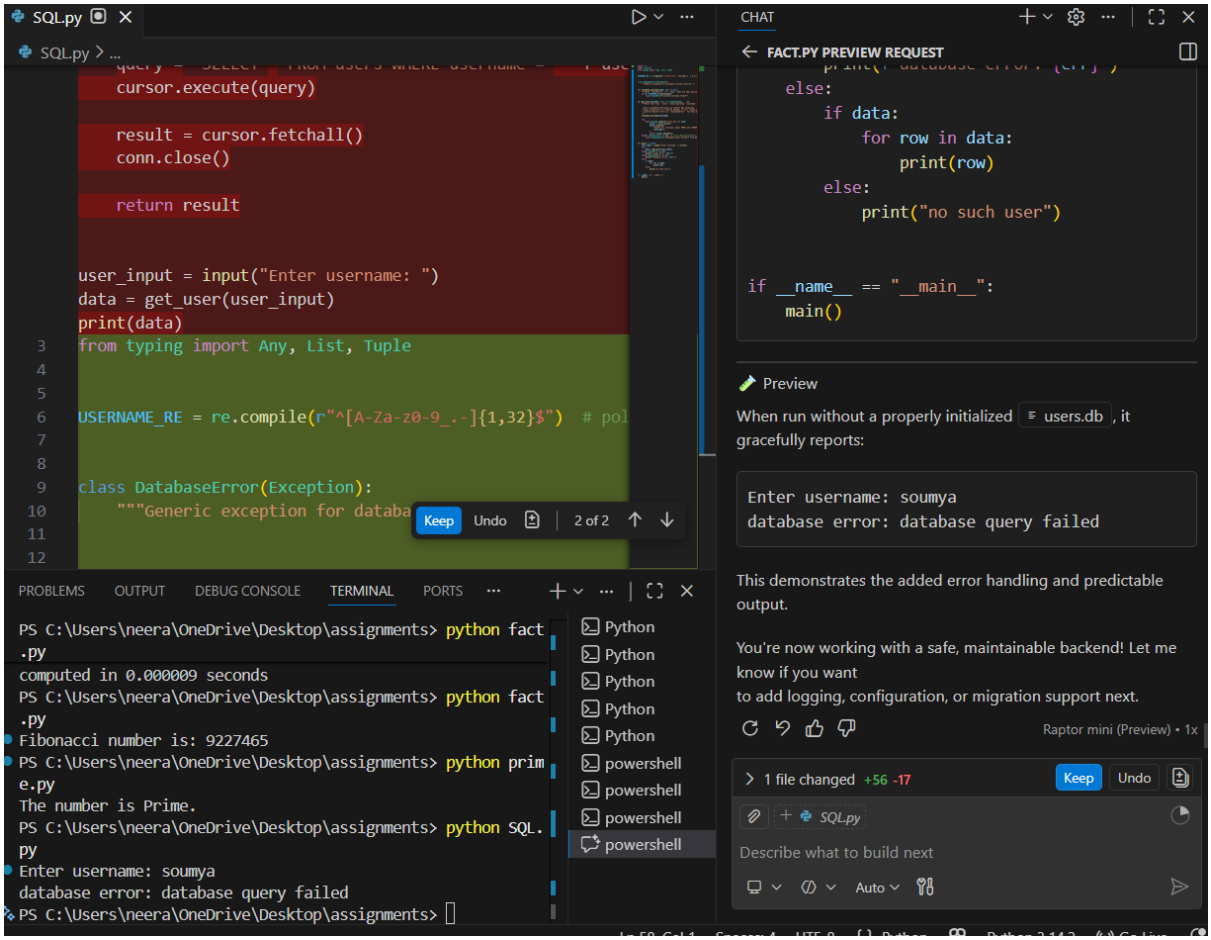
```python
import re
import sqlite3
from typing import Any, List, Tuple


USERNAME_RE = re.compile(r"^[A-Za-z0-9_.-]{1,32}$")  # policy can be adjusted


class DatabaseError(Exception):
    """Generic exception for database-related failures."""


def validate_username(name: str) -> None:
    """Raise ``ValueError`` if ``name`` does not meet policy."""
    if not USERNAME_RE.match(name):
        raise ValueError("invalid username format")


def get_user(username: str) -> List[Tuple[Any, ...]]:
    """Fetch rows from ``users`` table matching ``username``.

    - Uses a parameterized query to prevent SQL injection.
    - Wraps connection in a context manager so it's closed automatically.
    - Converts SQLite errors to ``DatabaseError`` for the caller.
```

```python
def main() -> None:
    user_input = input("Enter username: ").strip()
    try:
        data = get_user(user_input)
    except ValueError as err:
        print(f"input error: {err}")
    except DatabaseError as err:
        print(f"database error: {err}")
    else:
        if data:
            for row in data:
                print(row)
        else:
            print("no such user")


if __name__ == "__main__":
    main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

```
The number is Prime.
PS C:\Users\neera\OneDrive\Desktop\assignments> python SQL.py
Enter username: soumya
database error: database query failed
PS C:\Users\neera\OneDrive\Desktop\assignments>
```

#PREVIEW OF AI



Task 5: AI-Based Automated Code Review Report

Scenario

Your team uses AI tools to perform automated preliminary code reviews before human review, to improve code quality and consistency across projects.

Task Description

You are provided with a poorly written Python script.

Using AI-assisted review:

• Generate a structured code review report that evaluates:

o Code readability

o Naming conventions

o Formatting and style consistency

o Error handling

o Documentation quality

o Maintainability

The task is not just to fix the code, but to analyze and report on qualityissues.

Expected Outcome

• AI-generated review report including:

o Identified quality issues

o Risk areas

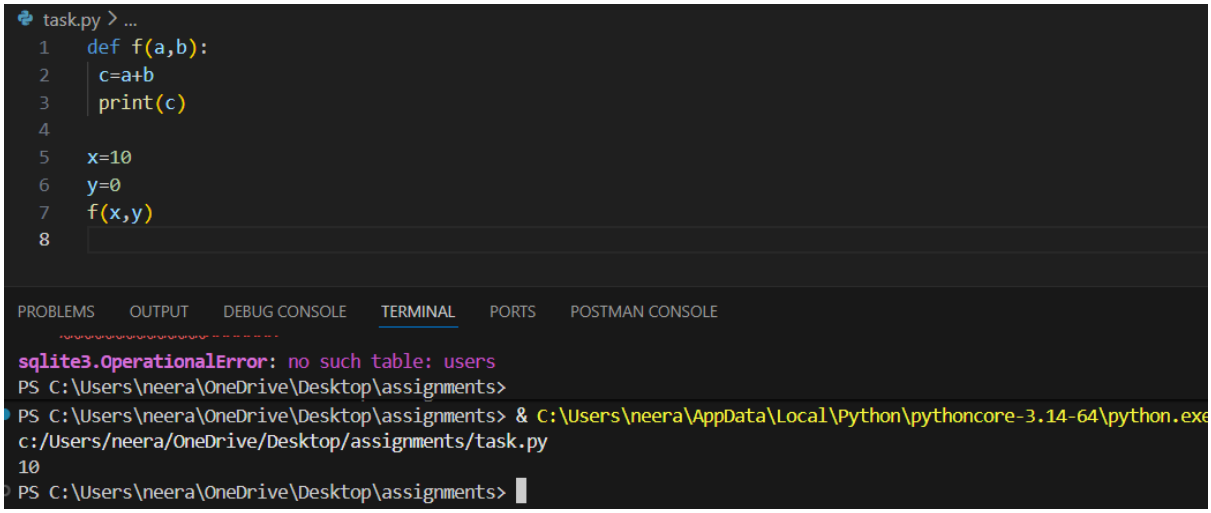
o Code smell detection

o Improvement suggestions

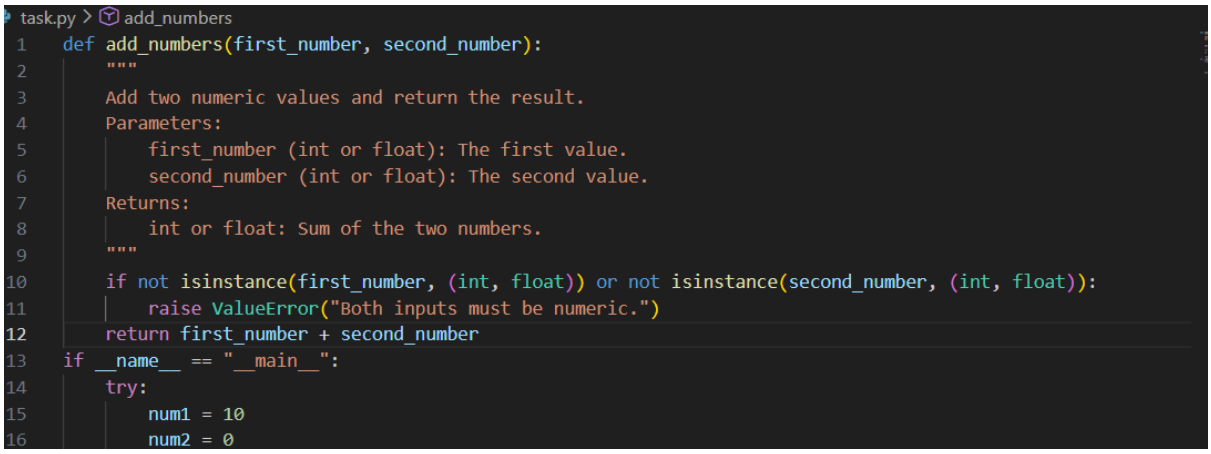
• Optional improved version of the code

• Demonstration of AI as a code reviewer, not just a code generator

#OPTIMIZED CODE



#ORIGINAL CODE:



```
if __name__ == "__main__":
    try:
        num1 = 10
        num2 = 0
        result = add_numbers(num1, num2)
        print("Result:", result)
    except ValueError as error:
        print("Input Error:", error)

ROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE

S C:\Users\neera\OneDrive\Desktop\assignments> & C:\Users\neera\AppData\Local\Python\pythoncore-3.14-64\python.ex
:/Users/neera/OneDrive/Desktop/assignments/task.py
esult: 10
S C:\Users\neera\OneDrive\Desktop\assignments>
```

#PREVIEW OF AI