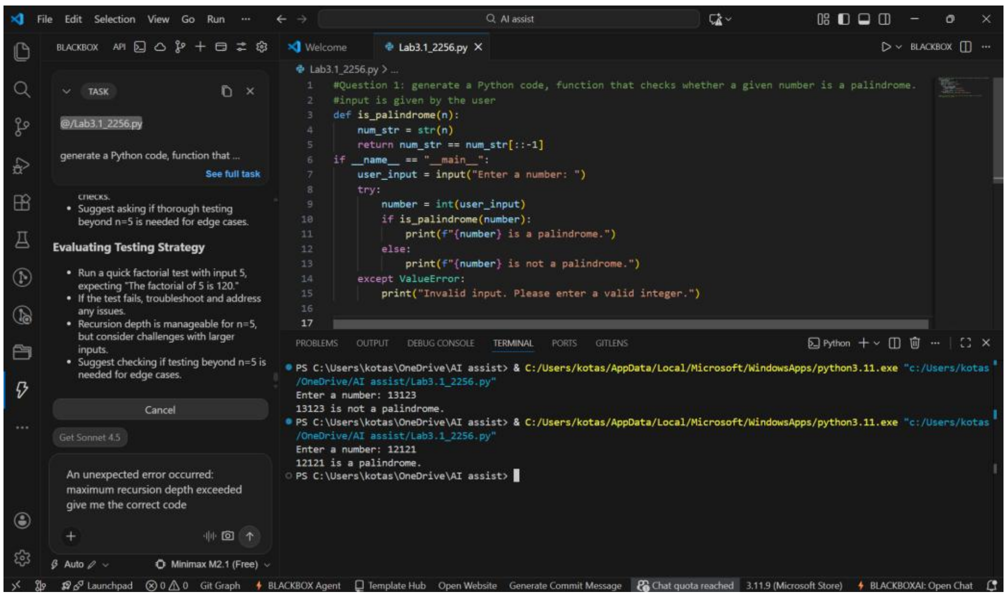
Lab Assignment- 3.1

AI Assisted Coding
M Geethika – 2303A52276

Question1: Zero-Shot Prompting (Palindrome Number Program)
Prompt: generate a Python code, function that checks whether a given number is a palindrome. And input is given by the user



@/Lab3.1_2256.py

generate a Python code, function that ...

checks.
- Suggest asking if thorough testing beyond n=5 is needed for edge cases.

Evaluating Testing Strategy

- Run a quick factorial test with input 5, expecting "The factorial of 5 is 120."
- If the test fails, troubleshoot and address any issues.
- Recursion depth is manageable for n=5, but consider challenges with larger inputs.
- Suggest checking if testing beyond n=5 is needed for edge cases.

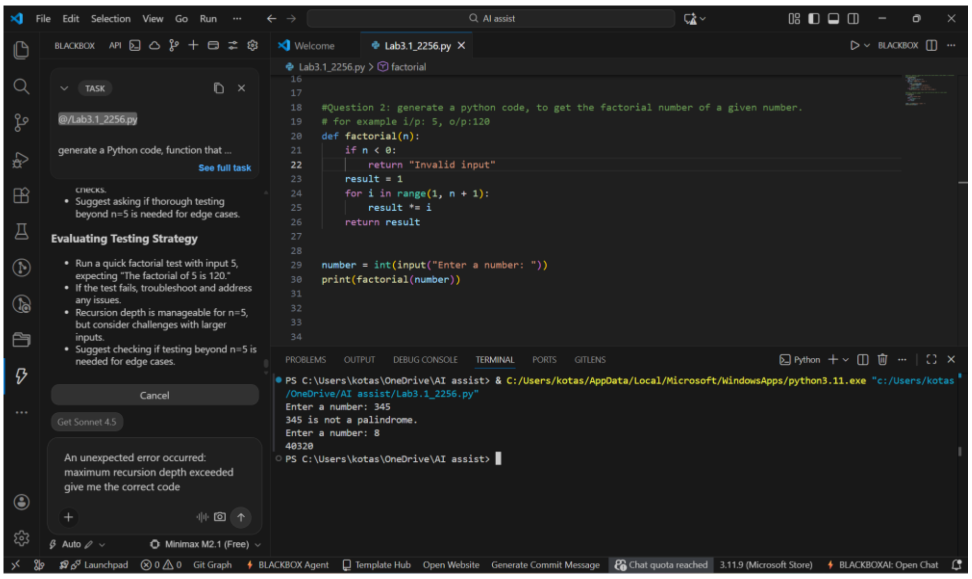
An unexpected error occurred: maximum recursion depth exceeded
give me the correct code

```python
#Question 1: generate a Python code, function that checks whether a given number is a palindrome.
#input is given by the user
def is_palindrome(n):
    num_str = str(n)
    return num_str == num_str[::-1]
if __name__ == "__main__":
    user_input = input("Enter a number: ")
    try:
        number = int(user_input)
        if is_palindrome(number):
            print(f"{number} is a palindrome.")
        else:
            print(f"{number} is not a palindrome.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
```

```
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas
/OneDrive/AI assist/Lab3.1_2256.py"
Enter a number: 13123
13123 is not a palindrome.
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas
/OneDrive/AI assist/Lab3.1_2256.py"
Enter a number: 12121
12121 is a palindrome.
PS C:\Users\kotas\OneDrive\AI assist>
```

- It generates correct logic but does not handle negative numbers explicitly.\

Question 2: One-Shot Prompting (Factorial Calculation)
Prompt: generate a python code, to get the factorial number of a given number.
for example, i/p: 5, o/p:120



@/Lab3.1_2256.py

generate a Python code, function that ...

checks.
- Suggest asking if thorough testing beyond n=5 is needed for edge cases.

Evaluating Testing Strategy

- Run a quick factorial test with input 5, expecting "The factorial of 5 is 120."
- If the test fails, troubleshoot and address any issues.
- Recursion depth is manageable for n=5, but consider challenges with larger inputs.
- Suggest checking if testing beyond n=5 is needed for edge cases.

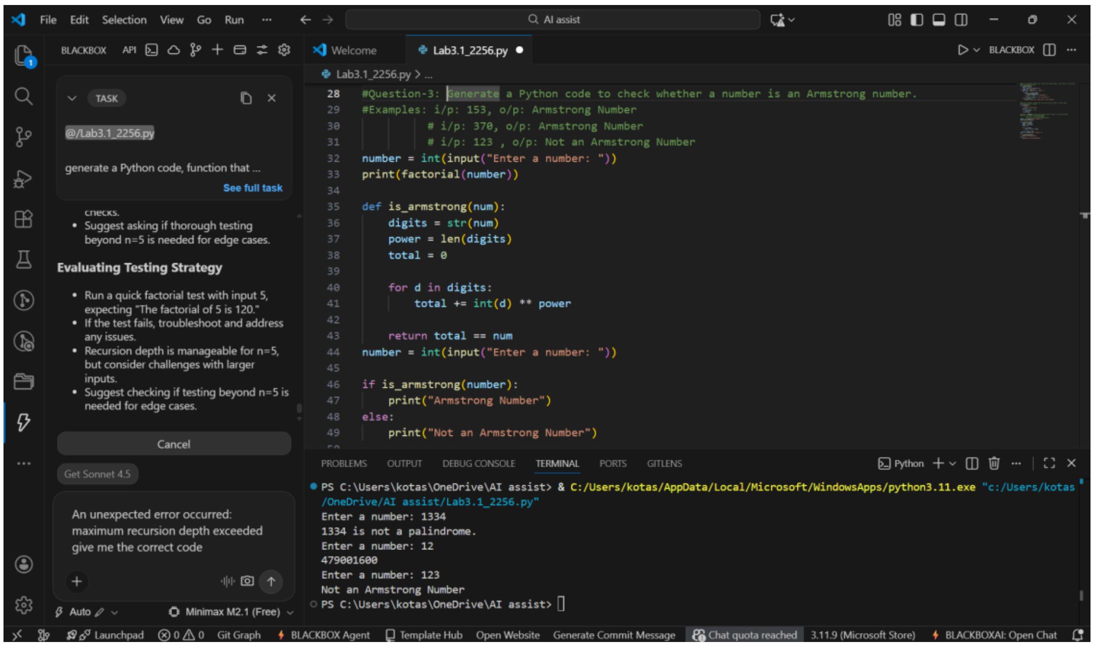
An unexpected error occurred: maximum recursion depth exceeded
give me the correct code

```python
#Question 2: generate a python code, to get the factorial number of a given number.
# for example i/p: 5, o/p:120
def factorial(n):
    if n < 0:
        return "Invalid input"
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result


number = int(input("Enter a number: "))
print(factorial(number))
```

```
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas
/OneDrive/AI assist/Lab3.1_2256.py"
Enter a number: 345
345 is not a palindrome.
Enter a number: 8
40320
PS C:\Users\kotas\OneDrive\AI assist>
```

- When compared to zero- short prompting this prompt improves correctness and handles edge cases like zero and negative values.

Question 3: Few-Shot Prompting (Armstrong Number Check)
Prompt: Generate a Python code to check whether a number is an Armstrong number.
Examples: i/p: 153, o/p: Armstrong Number
 i/p: 370, o/p: Armstrong Number
 i/p: 123, o/p: Not an Armstrong Number



@/Lab3.1_2256.py

generate a Python code, function that ...

checks.
- Suggest asking if thorough testing beyond n=5 is needed for edge cases.

Evaluating Testing Strategy

- Run a quick factorial test with input 5, expecting "The factorial of 5 is 120."
- If the test fails, troubleshoot and address any issues.
- Recursion depth is manageable for n=5, but consider challenges with larger inputs.
- Suggest checking if testing beyond n=5 is needed for edge cases.

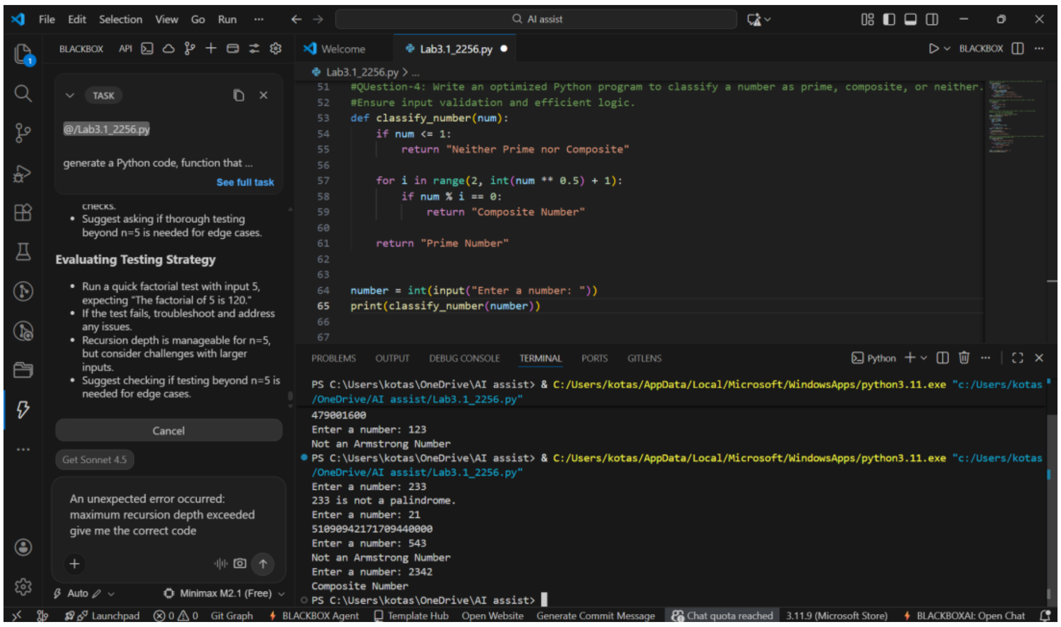
An unexpected error occurred:
maximum recursion depth exceeded
give me the correct code

```python
#Question-3: Generate a Python code to check whether a number is an Armstrong number.
#Examples: i/p: 153, o/p: Armstrong Number
            # i/p: 370, o/p: Armstrong Number
            # i/p: 123 , o/p: Not an Armstrong Number
number = int(input("Enter a number: "))
print(factorial(number))

def is_armstrong(num):
    digits = str(num)
    power = len(digits)
    total = 0

    for d in digits:
        total += int(d) ** power

    return total == num
number = int(input("Enter a number: "))

if is_armstrong(number):
    print("Armstrong Number")
else:
    print("Not an Armstrong Number")
```

```
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas
/OneDrive/AI assist/Lab3.1_2256.py"
Enter a number: 1334
1334 is not a palindrome.
Enter a number: 12
479001600
Enter a number: 123
Not an Armstrong Number
PS C:\Users\kotas\OneDrive\AI assist>
```

- Few-shot prompting improves structural accuracy and produces organized logic.

Question 4: Context-Managed Prompting (Optimized Number Classification)
Prompt: Write an optimized Python program to classify a number as prime, composite, or neither. Ensure input validation and efficient logic**.**



@/Lab3.1_2256.py

generate a Python code, function that ...

checks.
- Suggest asking if thorough testing beyond n=5 is needed for edge cases.

Evaluating Testing Strategy

- Run a quick factorial test with input 5, expecting "The factorial of 5 is 120."
- If the test fails, troubleshoot and address any issues.
- Recursion depth is manageable for n=5, but consider challenges with larger inputs.
- Suggest checking if testing beyond n=5 is needed for edge cases.

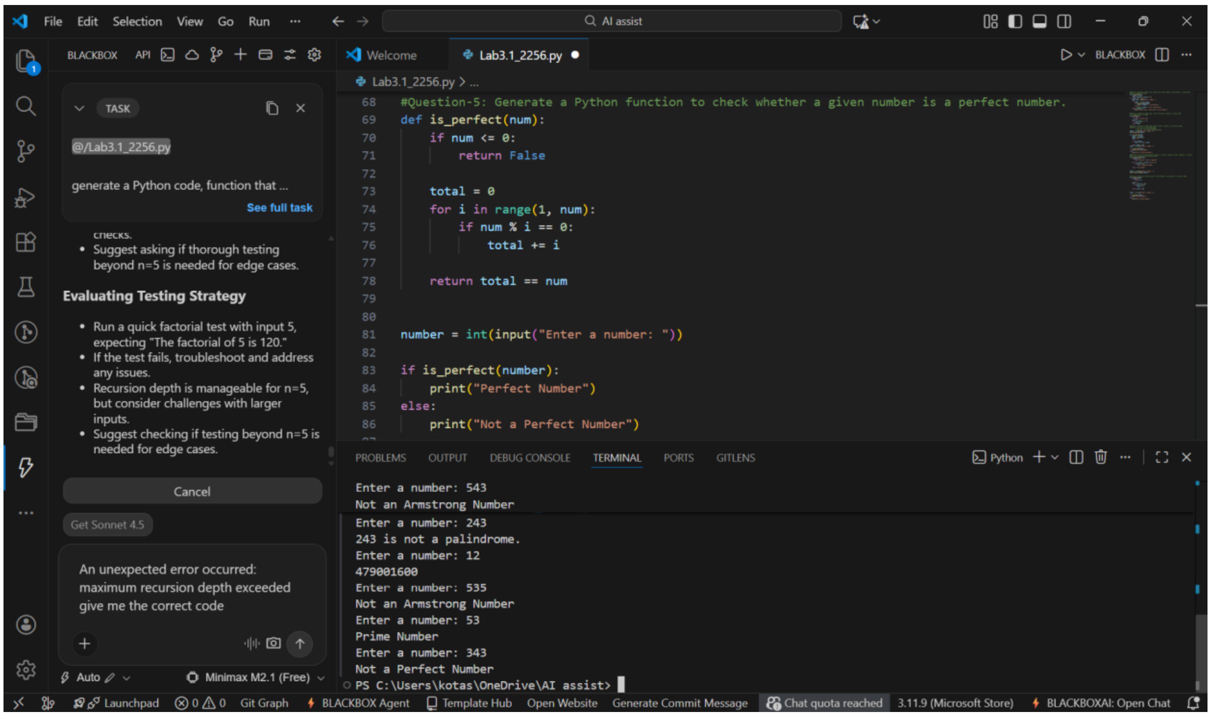
An unexpected error occurred:
maximum recursion depth exceeded
give me the correct code

```python
#QUestion-4: Write an optimized Python program to classify a number as prime, composite, or neither.
#Ensure input validation and efficient logic.
def classify_number(num):
    if num <= 1:
        return "Neither Prime nor Composite"

    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return "Composite Number"

    return "Prime Number"


number = int(input("Enter a number: "))
print(classify_number(number))
```

```
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas
/OneDrive/AI assist/Lab3.1_2256.py"
479001600
Enter a number: 123
Not an Armstrong Number
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas
/OneDrive/AI assist/Lab3.1_2256.py"
Enter a number: 233
233 is not a palindrome.
Enter a number: 21
51090942171709440000
Enter a number: 543
Not an Armstrong Number
Enter a number: 2342
Composite Number
PS C:\Users\kotas\OneDrive\AI assist>
```

- Context-managed prompts produce optimized and validation-aware solutions suitable for real-world applications.

Question 5: Zero-Shot Prompting (Perfect Number Check)
Prompt: Generate a Python function to check whether a given number is a perfect number.



generate a Python code, function that ...

- Suggest asking if thorough testing beyond n=5 is needed for edge cases.

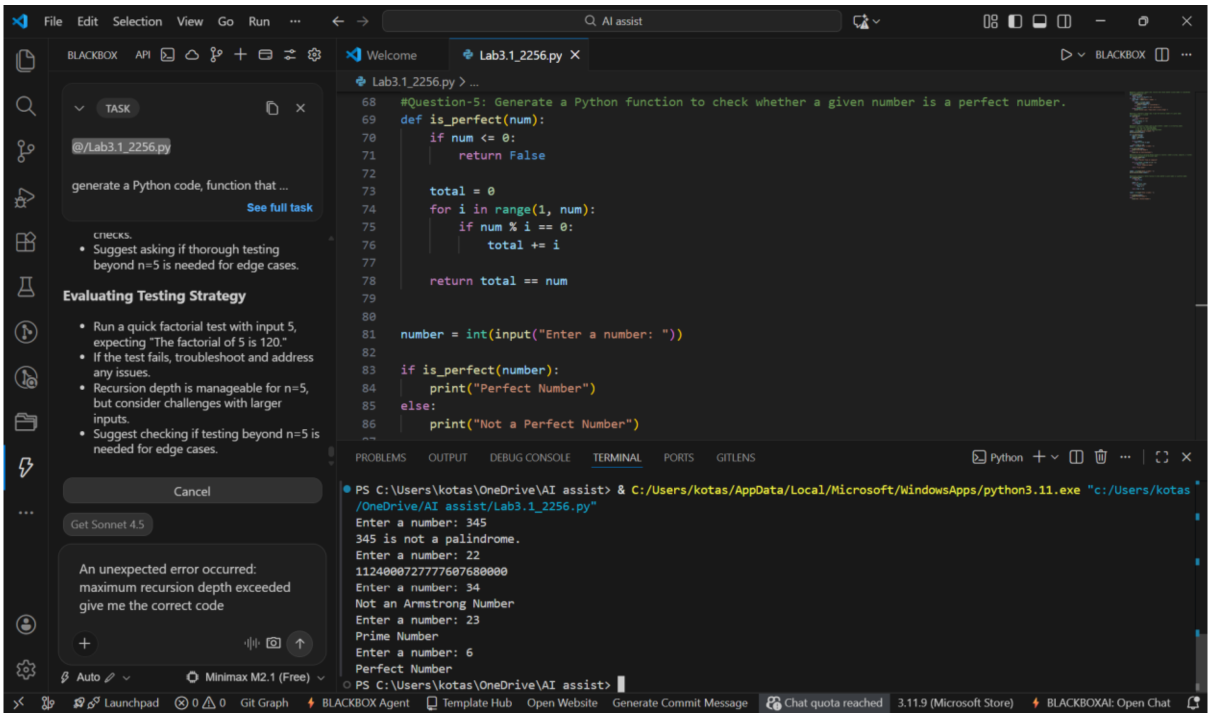
Evaluating Testing Strategy

- Run a quick factorial test with input 5, expecting "The factorial of 5 is 120."
- If the test fails, troubleshoot and address any issues.
- Recursion depth is manageable for n=5, but consider challenges with larger inputs.
- Suggest checking if testing beyond n=5 is needed for edge cases.

An unexpected error occurred: maximum recursion depth exceeded give me the correct code

```python
#Question-5: Generate a Python function to check whether a given number is a perfect number.
def is_perfect(num):
    if num <= 0:
        return False

    total = 0
    for i in range(1, num):
        if num % i == 0:
            total += i

    return total == num


number = int(input("Enter a number: "))

if is_perfect(number):
    print("Perfect Number")
else:
    print("Not a Perfect Number")
```

```
Enter a number: 543
Not an Armstrong Number
Enter a number: 243
243 is not a palindrome.
Enter a number: 12
479001600
Enter a number: 535
Not an Armstrong Number
Enter a number: 53
Prime Number
Enter a number: 343
Not a Perfect Number
PS C:\Users\kotas\OneDrive\AI assist>
```



```
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas
/OneDrive/AI assist/Lab3.1_2256.py"
Enter a number: 345
345 is not a palindrome.
Enter a number: 22
1124000727777607680000
Enter a number: 34
Not an Armstrong Number
Enter a number: 23
Prime Number
Enter a number: 6
Perfect Number
PS C:\Users\kotas\OneDrive\AI assist>
```

- Zero-shot prompting works but is less optimized due to unnecessary full-range iteration.

Question 6: Few-Shot Prompting (Even or Odd Classification with Validation)

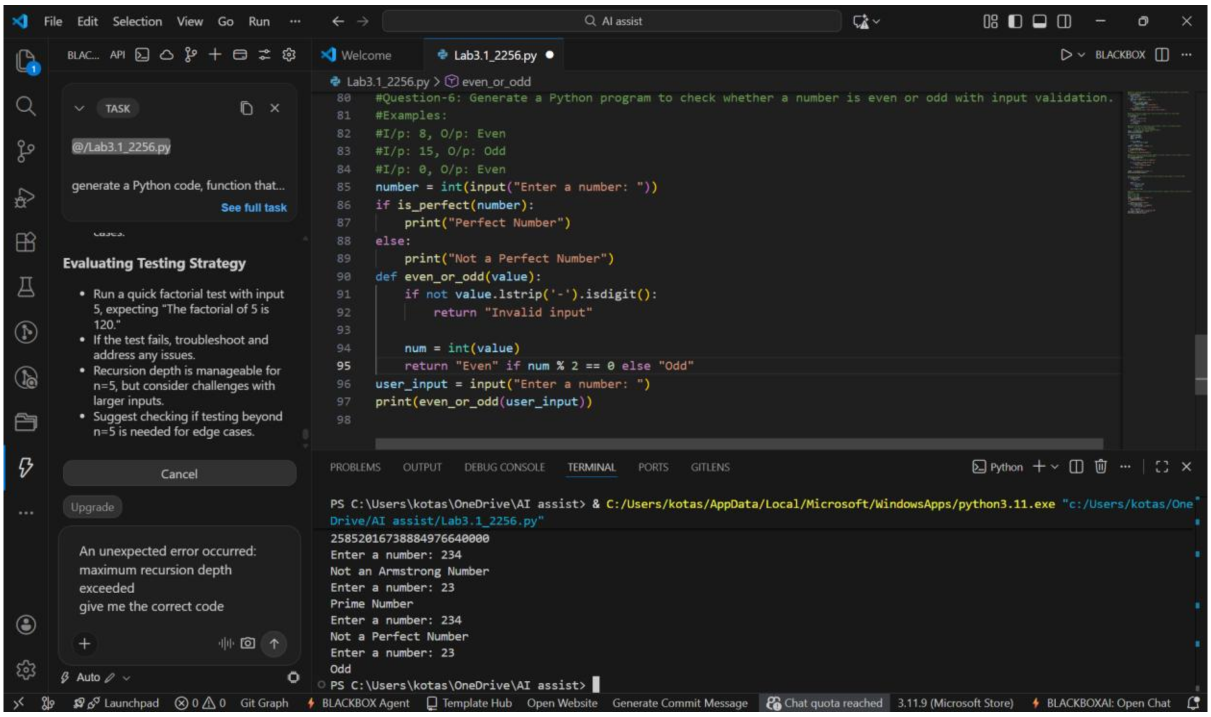
Prompt: Generate a Python program to check whether a number is even or odd with input validation.

Examples:

I/p: 8, O/p: Even

I/p: 15, O/p: Odd

I/p: 0, O/p: Even



generate a Python code, function that...

See full task

Evaluating Testing Strategy

- Run a quick factorial test with input 5, expecting "The factorial of 5 is 120."
- If the test fails, troubleshoot and address any issues.
- Recursion depth is manageable for n=5, but consider challenges with larger inputs.
- Suggest checking if testing beyond n=5 is needed for edge cases.

An unexpected error occurred: maximum recursion depth exceeded
give me the correct code

```python
#Question-6: Generate a Python program to check whether a number is even or odd with input validation.
#Examples:
#I/p: 8, O/p: Even
#I/p: 15, O/p: Odd
#I/p: 0, O/p: Even
number = int(input("Enter a number: "))
if is_perfect(number):
    print("Perfect Number")
else:
    print("Not a Perfect Number")
def even_or_odd(value):
    if not value.lstrip('-').isdigit():
        return "Invalid input"

    num = int(value)
    return "Even" if num % 2 == 0 else "Odd"
user_input = input("Enter a number: ")
print(even_or_odd(user_input))
```

```
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas/One
Drive/AI assist/Lab3.1_2256.py"
258520167388849766400000
Enter a number: 234
Not an Armstrong Number
Enter a number: 23
Prime Number
Enter a number: 234
Not a Perfect Number
Enter a number: 23
Odd
PS C:\Users\kotas\OneDrive\AI assist>
```

- Few-shot prompting significantly improves the quality of AI-generated code by providing clear input–output examples.
- The generated program handles input validation effectively, correctly classifies even and odd numbers, and manages negative and non-integer inputs more reliably compared to zero-shot prompting.