ASSIGNMENT – 3.1

2303A51850

B-13

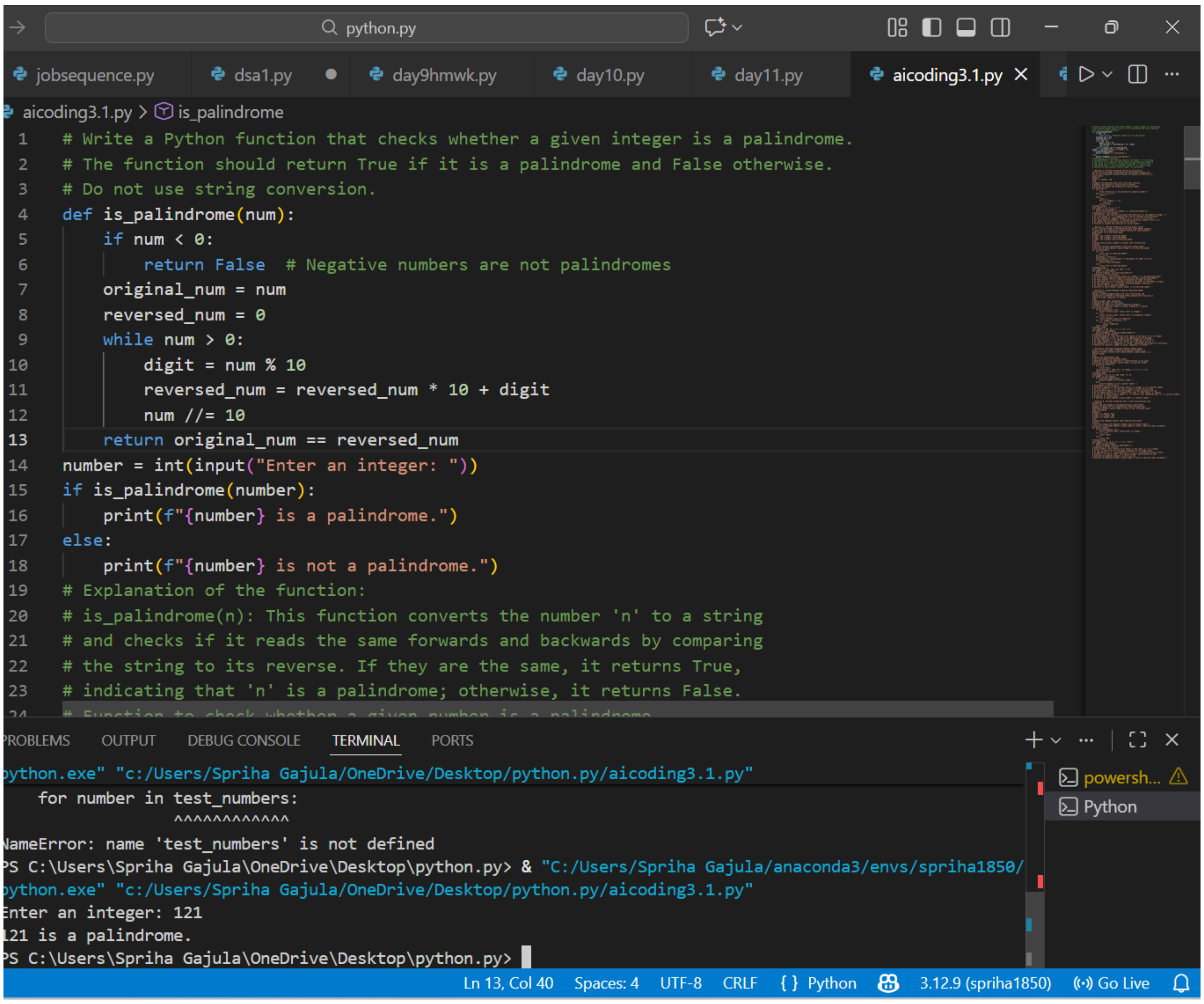
Question 1: Zero-Shot Prompting (Palindrome Number Program)

Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a palindrome.

Task:

- Record the AI-generated code.
- Test the code with multiple inputs.
- Identify any logical errors or missing edge-case handling.

CODE:



```python
# Write a Python function that checks whether a given integer is a palindrome.
# The function should return True if it is a palindrome and False otherwise.
# Do not use string conversion.
def is_palindrome(num):
    if num < 0:
        return False  # Negative numbers are not palindromes
    original_num = num
    reversed_num = 0
    while num > 0:
        digit = num % 10
        reversed_num = reversed_num * 10 + digit
        num //= 10
    return original_num == reversed_num
number = int(input("Enter an integer: "))
if is_palindrome(number):
    print(f"{number} is a palindrome.")
else:
    print(f"{number} is not a palindrome.")
# Explanation of the function:
# is_palindrome(n): This function converts the number 'n' to a string
# and checks if it reads the same forwards and backwards by comparing
# the string to its reverse. If they are the same, it returns True,
# indicating that 'n' is a palindrome; otherwise, it returns False.
# Function to check whether a given number is a palindrome
```

```
ython.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/aicoding3.1.py"
    for number in test_numbers:
                  ^^^^^^^^^^^^
NameError: name 'test_numbers' is not defined
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/
ython.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/aicoding3.1.py"
Enter an integer: 121
121 is a palindrome.
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py>
```

Explanation:

This function converts the number 'n' to a string and checks if it reads the same forwards and backwards by comparing the string to its reverse. If they are the same, it returns True, indicating that 'n' is a palindrome; otherwise, it returns False. Function to check whether a given number is a palindrome.

Question 2: One-Shot Prompting (Factorial Calculation)

Write a one-shot prompt by providing one input-output example and ask the AI to generate a Python function to compute the factorial of a given number.

Example:

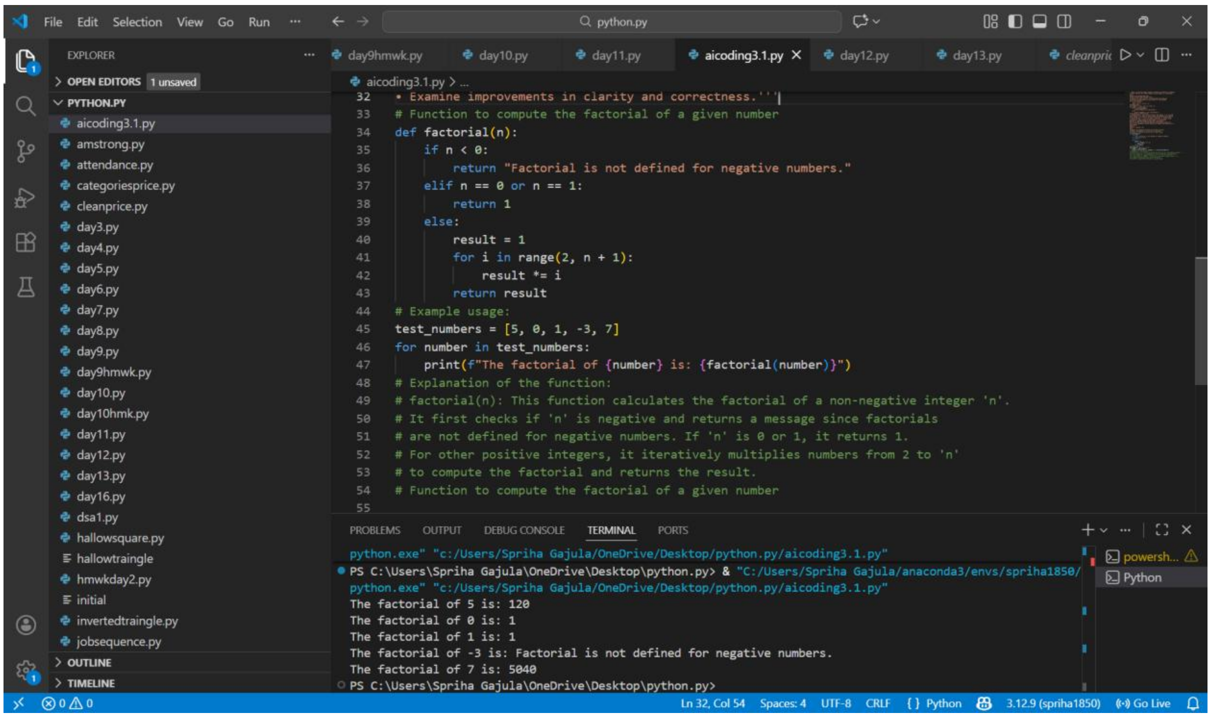
Input: 5 → Output: 120

Task:

• Compare the generated code with a zero-shot solution.

• Examine improvements in clarity and correctness.

One-shot code:



```python
• Examine improvements in clarity and correctness.'''
# Function to compute the factorial of a given number
def factorial(n):
    if n < 0:
        return "Factorial is not defined for negative numbers."
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result
# Example usage:
test_numbers = [5, 0, 1, -3, 7]
for number in test_numbers:
    print(f"The factorial of {number} is: {factorial(number)}")
# Explanation of the function:
# factorial(n): This function calculates the factorial of a non-negative integer 'n'.
# It first checks if 'n' is negative and returns a message since factorials
# are not defined for negative numbers. If 'n' is 0 or 1, it returns 1.
# For other positive integers, it iteratively multiplies numbers from 2 to 'n'
# to compute the factorial and returns the result.
# Function to compute the factorial of a given number
```

```
python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/aicoding3.1.py"
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/
python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/aicoding3.1.py"
The factorial of 5 is: 120
The factorial of 0 is: 1
The factorial of 1 is: 1
The factorial of -3 is: Factorial is not defined for negative numbers.
The factorial of 7 is: 5040
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py>
```

Explaination:

factorial(n): This function calculates the factorial of a non-negative integer 'n'.

It first checks if 'n' is negative and returns a message since factorials

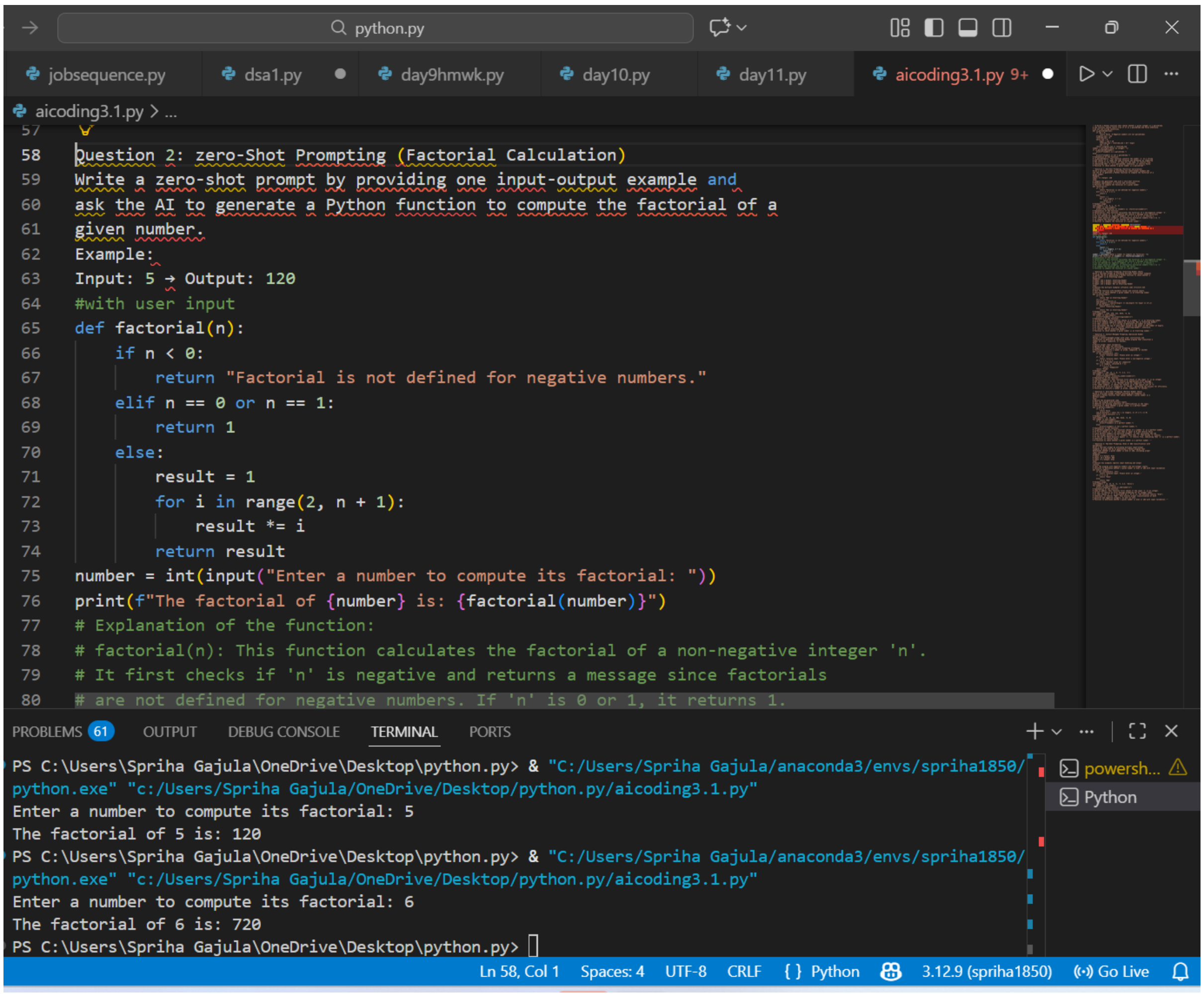
are not defined for negative numbers. If 'n' is 0 or 1, it returns 1.

For other positive integers, it iteratively multiplies numbers from 2 to 'n'

to compute the factorial and returns the result.

Function to compute the factorial of a given number

Zero-shot code:



```
Question 2: zero-Shot Prompting (Factorial Calculation)
Write a zero-shot prompt by providing one input-output example and
ask the AI to generate a Python function to compute the factorial of a
given number.
Example:
Input: 5 → Output: 120
#with user input
def factorial(n):
    if n < 0:
        return "Factorial is not defined for negative numbers."
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result
number = int(input("Enter a number to compute its factorial: "))
print(f"The factorial of {number} is: {factorial(number)}")
# Explanation of the function:
# factorial(n): This function calculates the factorial of a non-negative integer 'n'.
# It first checks if 'n' is negative and returns a message since factorials
# are not defined for negative numbers. If 'n' is 0 or 1, it returns 1.
```

```
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/
python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/aicoding3.1.py"
Enter a number to compute its factorial: 5
The factorial of 5 is: 120
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/
python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/aicoding3.1.py"
Enter a number to compute its factorial: 6
The factorial of 6 is: 720
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py>
```

Explaination:

factorial(n): This function calculates the factorial of a non-negative integer 'n'.

It first checks if 'n' is negative and returns a message since factorials are not defined for negative numbers. If 'n' is 0 or 1, it returns 1.

For other positive integers, it iteratively multiplies numbers from 2 to 'n'

to compute the factorial and returns the result. Function to compute the

factorial of a given number

Question 3: Few-Shot Prompting (Armstrong Number Check)

Write a few-shot prompt by providing multiple input-output examples to guide

the AI in generating a Python function to check whether a given number is an

Armstrong number.

Examples:

• Input: 153 → Output: Armstrong Number

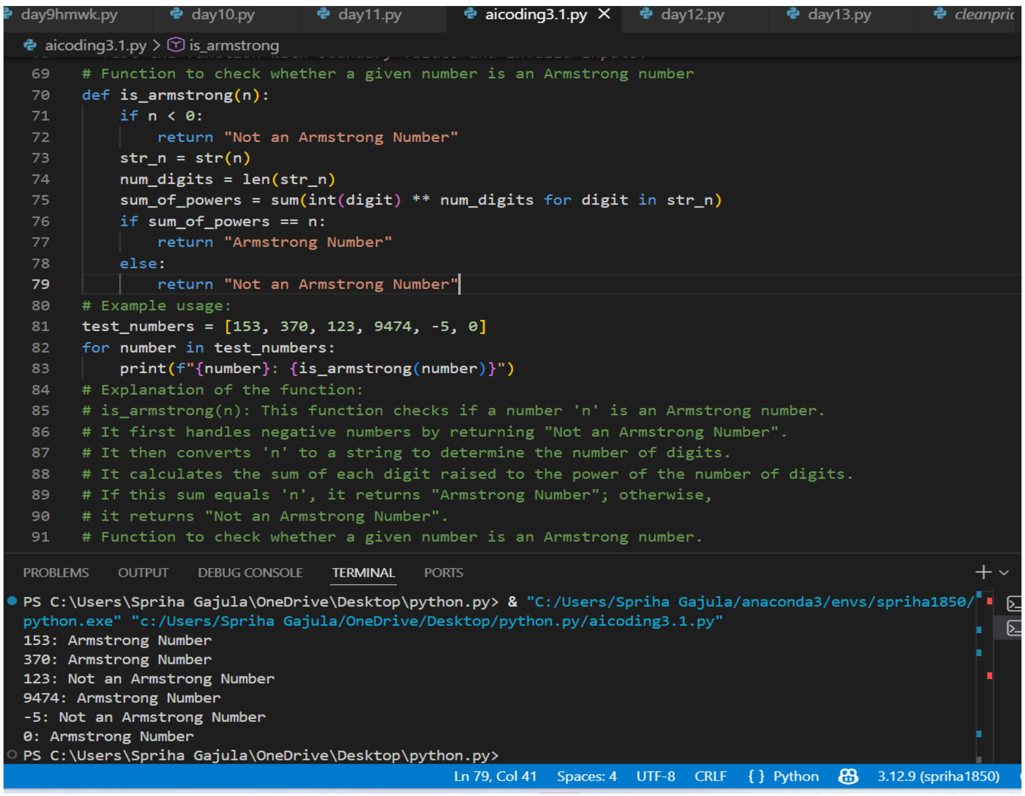
• Input: 370 → Output: Armstrong Number

Task:

• Analyze how multiple examples influence code structure and accuracy.

• Test the function with boundary values and invalid inputs.

Code:



```python
# Function to check whether a given number is an Armstrong number
def is_armstrong(n):
    if n < 0:
        return "Not an Armstrong Number"
    str_n = str(n)
    num_digits = len(str_n)
    sum_of_powers = sum(int(digit) ** num_digits for digit in str_n)
    if sum_of_powers == n:
        return "Armstrong Number"
    else:
        return "Not an Armstrong Number"
# Example usage:
test_numbers = [153, 370, 123, 9474, -5, 0]
for number in test_numbers:
    print(f"{number}: {is_armstrong(number)}")
# Explanation of the function:
# is_armstrong(n): This function checks if a number 'n' is an Armstrong number.
# It first handles negative numbers by returning "Not an Armstrong Number".
# It then converts 'n' to a string to determine the number of digits.
# It calculates the sum of each digit raised to the power of the number of digits.
# If this sum equals 'n', it returns "Armstrong Number"; otherwise,
# it returns "Not an Armstrong Number".
# Function to check whether a given number is an Armstrong number.
```

```
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/aicoding3.1.py"
153: Armstrong Number
370: Armstrong Number
123: Not an Armstrong Number
9474: Armstrong Number
-5: Not an Armstrong Number
0: Armstrong Number
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py>
```

Explanation:

is_armstrong(n): This function checks if a number 'n' is an Armstrong number.

It first handles negative numbers by returning "Not an Armstrong Number".

It then converts 'n' to a string to determine the number of digits.

It calculates the sum of each digit raised to the power of the number of digits.

If this sum equals 'n', it returns "Armstrong Number"; otherwise,

it returns "Not an Armstrong Number".

Function to check whether a given number is an Armstrong number

Question 4: Context-Managed Prompting (Optimized Number Classification)

Design a context-managed prompt with clear instructions and constraints to generate an optimized Python program that classifies a number as prime, composite, or neither.

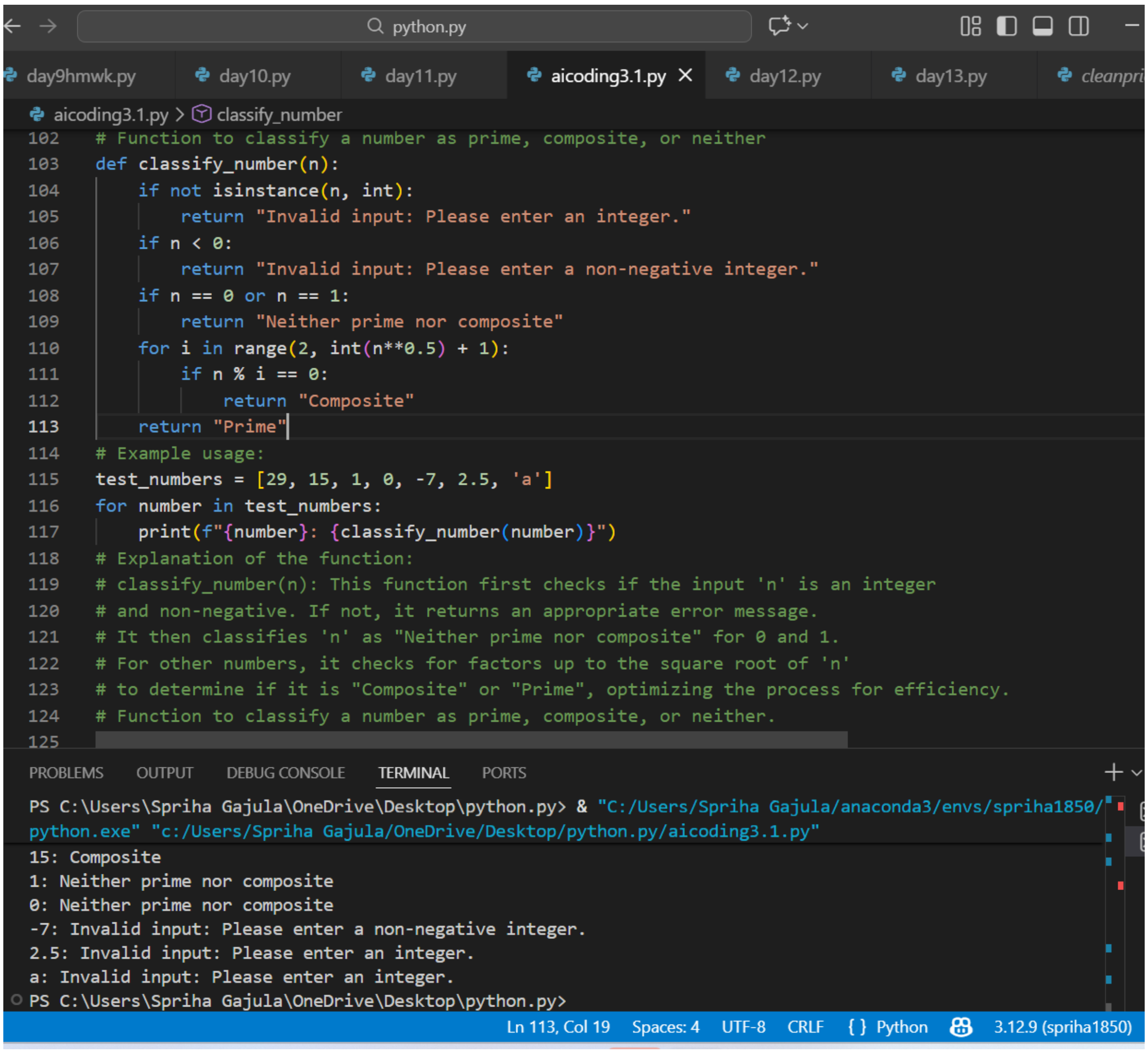
Task:

• Ensure proper input validation.

• Optimize the logic for efficiency.

• Compare the output with earlier prompting strategies.

Code:



```python
# Function to classify a number as prime, composite, or neither
def classify_number(n):
    if not isinstance(n, int):
        return "Invalid input: Please enter an integer."
    if n < 0:
        return "Invalid input: Please enter a non-negative integer."
    if n == 0 or n == 1:
        return "Neither prime nor composite"
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return "Composite"
    return "Prime"
# Example usage:
test_numbers = [29, 15, 1, 0, -7, 2.5, 'a']
for number in test_numbers:
    print(f"{number}: {classify_number(number)}")
# Explanation of the function:
# classify_number(n): This function first checks if the input 'n' is an integer
# and non-negative. If not, it returns an appropriate error message.
# It then classifies 'n' as "Neither prime nor composite" for 0 and 1.
# For other numbers, it checks for factors up to the square root of 'n'
# to determine if it is "Composite" or "Prime", optimizing the process for efficiency.
# Function to classify a number as prime, composite, or neither.
```

```
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/
python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/aicoding3.1.py"
15: Composite
1: Neither prime nor composite
0: Neither prime nor composite
-7: Invalid input: Please enter a non-negative integer.
2.5: Invalid input: Please enter an integer.
a: Invalid input: Please enter an integer.
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py>
```

Explanation:

classify_number(n): This function first checks if the input 'n' is an integer

and non-negative. If not, it returns an appropriate error message.

It then classifies 'n' as "Neither prime nor composite" for 0 and 1.

For other numbers, it checks for factors up to the square root of 'n'

to determine if it is "Composite" or "Prime", optimizing the process for efficiency.

Function to classify a number as prime, composite, or neither.

Question 5: Zero-Shot Prompting (Perfect Number Check)

Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is perfect number.

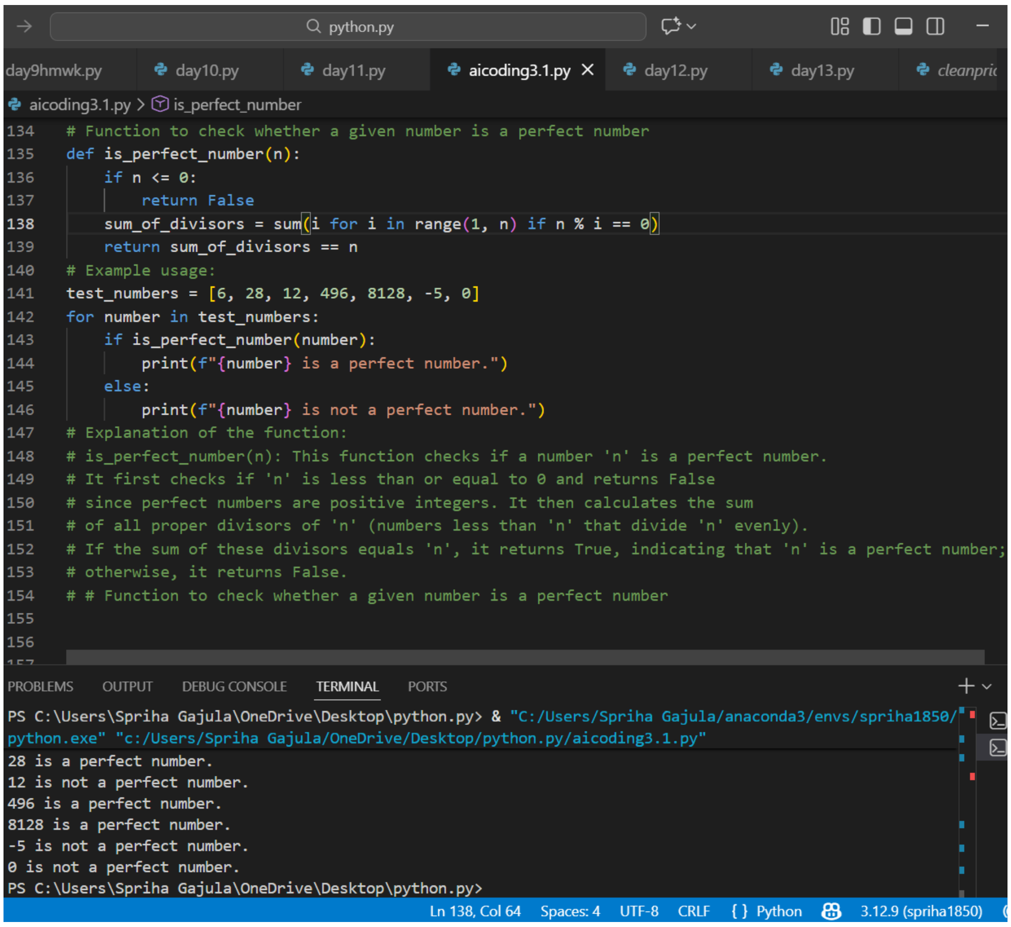
Task:

• Record the AI-generated code.

• Test the program with multiple inputs.

• Identify any missing conditions or inefficiencies in the logic.

Code:



```python
# Function to check whether a given number is a perfect number
def is_perfect_number(n):
    if n <= 0:
        return False
    sum_of_divisors = sum(i for i in range(1, n) if n % i == 0)
    return sum_of_divisors == n
# Example usage:
test_numbers = [6, 28, 12, 496, 8128, -5, 0]
for number in test_numbers:
    if is_perfect_number(number):
        print(f"{number} is a perfect number.")
    else:
        print(f"{number} is not a perfect number.")
# Explanation of the function:
# is_perfect_number(n): This function checks if a number 'n' is a perfect number.
# It first checks if 'n' is less than or equal to 0 and returns False
# since perfect numbers are positive integers. It then calculates the sum
# of all proper divisors of 'n' (numbers less than 'n' that divide 'n' evenly).
# If the sum of these divisors equals 'n', it returns True, indicating that 'n' is a perfect number;
# otherwise, it returns False.
# # Function to check whether a given number is a perfect number
```

```
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/
python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/aicoding3.1.py"
28 is a perfect number.
12 is not a perfect number.
496 is a perfect number.
8128 is a perfect number.
-5 is not a perfect number.
0 is not a perfect number.
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py>
```

Explanation:

is_perfect_number(n): This function checks if a number 'n' is a perfect number.

It first checks if 'n' is less than or equal to 0 and returns False

since perfect numbers are positive integers. It then calculates the sum of all

proper divisors of 'n' (numbers less than 'n' that divide 'n' evenly).

If the sum of these divisors equals 'n', it returns True, indicating that 'n' is a

perfect number; otherwise, it returns False.

Function to check whether a given number is a perfect number

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

Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python program that determines whether a given number is even or odd, including proper input validation.

Examples:

• Input: 8 → Output: Even

• Input: 15 → Output: Odd

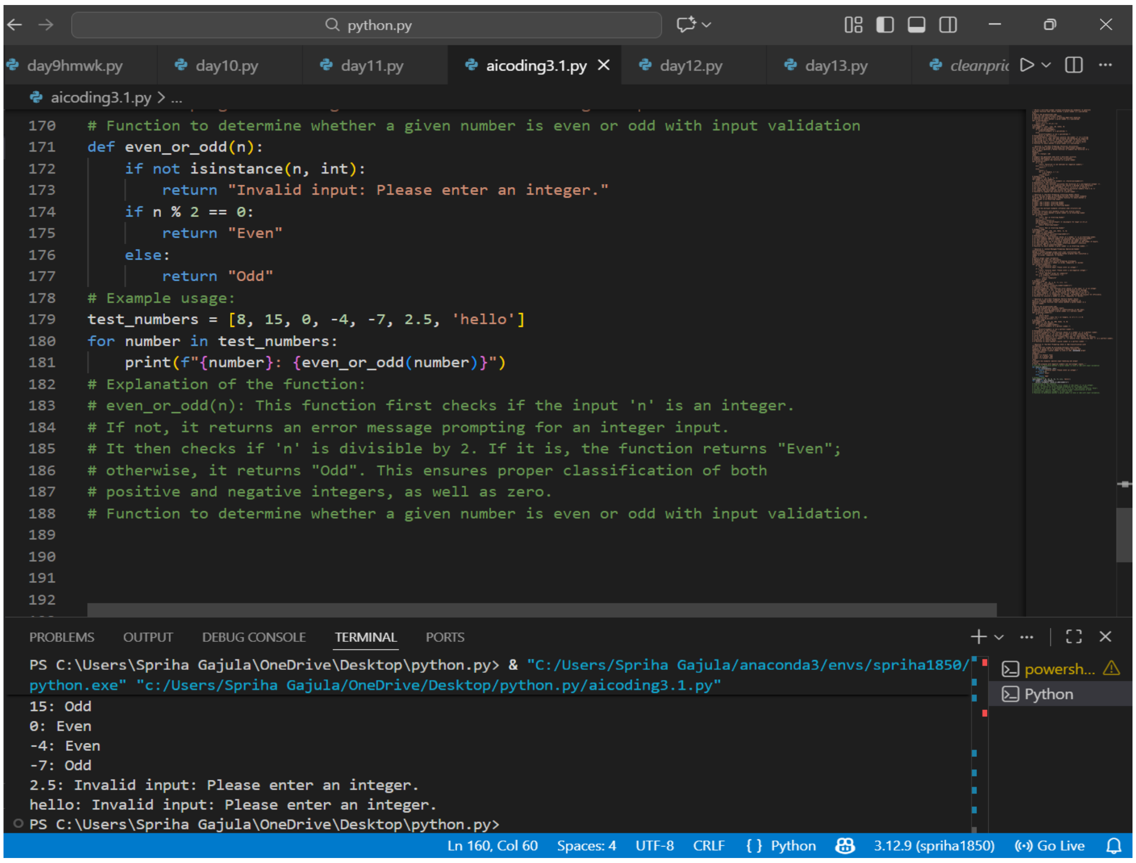
• Input: 0 → Output: Even

Task:

• Analyze how examples improve input handling and outputclarity.

• Test the program with negative numbers and non-integer inputs.

Code:



```python
# Function to determine whether a given number is even or odd with input validation
def even_or_odd(n):
    if not isinstance(n, int):
        return "Invalid input: Please enter an integer."
    if n % 2 == 0:
        return "Even"
    else:
        return "Odd"
# Example usage:
test_numbers = [8, 15, 0, -4, -7, 2.5, 'hello']
for number in test_numbers:
    print(f"{number}: {even_or_odd(number)}")
# Explanation of the function:
# even_or_odd(n): This function first checks if the input 'n' is an integer.
# If not, it returns an error message prompting for an integer input.
# It then checks if 'n' is divisible by 2. If it is, the function returns "Even";
# otherwise, it returns "Odd". This ensures proper classification of both
# positive and negative integers, as well as zero.
# Function to determine whether a given number is even or odd with input validation.
```

```
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/aicoding3.1.py"
15: Odd
0: Even
-4: Even
-7: Odd
2.5: Invalid input: Please enter an integer.
hello: Invalid input: Please enter an integer.
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py>
```

Explanation:
even_or_odd(n): This function first checks if the input 'n' is an integer.

If not, it returns an error message prompting for an integer input.

It then checks if 'n' is divisible by 2. If it is, the function returns "Even"; otherwise,

it returns "Odd". This ensures proper classification of both

positive and negative integers, as well as zero.

Function to determine whether a given number is even or odd with input validation.