ASSIGNMENT – 3.1

RAVULA AJAY
2303A52370
BATCH- 44

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.

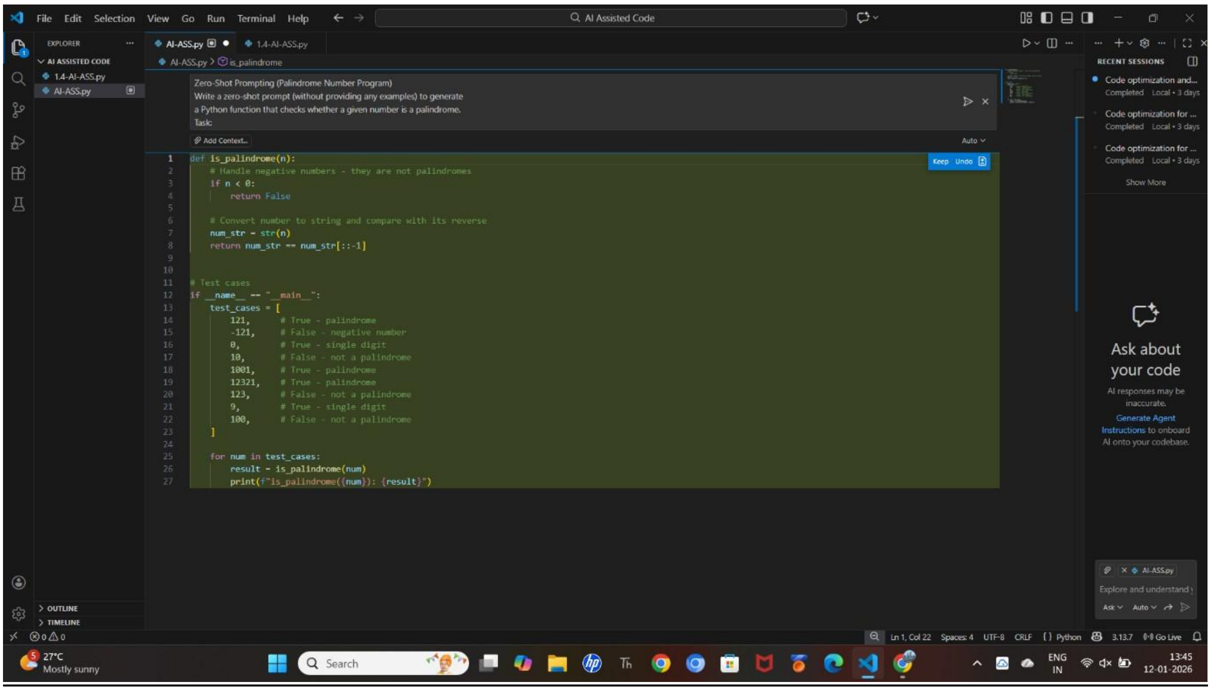
Task:

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

PROMPT:

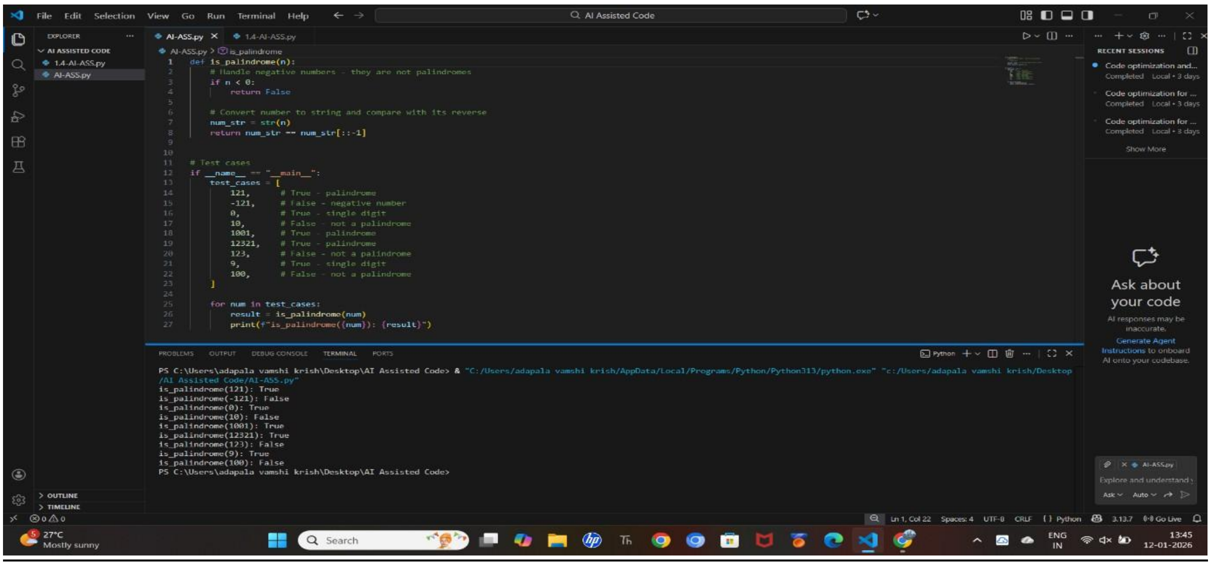
checking whether given number it is palindrome or not

CODE:



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:

```python
def is_palindrome(n):
    # Handle negative numbers - they are not palindromes
    if n < 0:
        return False

    # Convert number to string and compare with its reverse
    num_str = str(n)
    return num_str == num_str[::-1]


# Test cases
if __name__ == "__main__":
    test_cases = [
        121,      # True - palindrome
        -121,     # False - negative number
        0,        # True - single digit
        10,       # False - not a palindrome
        1001,     # True - palindrome
        12321,    # True - palindrome
        123,      # False - not a palindrome
        9,        # True - single digit
        100,      # False - not a palindrome
    ]

    for num in test_cases:
        result = is_palindrome(num)
        print(f"is_palindrome({num}): {result}")
```


File Edit Selection View Go Run Terminal Help
AI Assisted Code
EXPLORER
AI ASSISTED CODE
1.4-AI-ASS.py
AI-ASS.py
AI-ASS.py > is_palindrome
RECENT SESSIONS
Ask about your code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Ln 1, Col 22 Spaces: 4 UTF-8 CRLF {} Python 3.13.7 Go Live
27°C Mostly sunny
13:45 12-01-2026

```python
def is_palindrome(n):
    # Handle negative numbers - they are not palindromes
    if n < 0:
        return False

    # Convert number to string and compare with its reverse
    num_str = str(n)
    return num_str == num_str[::-1]


# Test cases
if __name__ == "__main__":
    test_cases = [
        121,       # True - palindrome
        -121,      # False - negative number
        0,         # True - single digit
        10,        # False - not a palindrome
        1001,      # True - palindrome
        12321,     # True - palindrome
        123,       # False - not a palindrome
        9,         # True - single digit
        100,       # False - not a palindrome
    ]

    for num in test_cases:
        result = is_palindrome(num)
        print(f"is_palindrome({num}): {result}")
```

```
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code> & "C:/Users/adapala vamshi krish/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/adapala vamshi krish/Desktop/AI Assisted Code/AI-ASS.py"
is_palindrome(121): True
is_palindrome(-121): False
is_palindrome(0): True
is_palindrome(10): False
is_palindrome(1001): True
is_palindrome(12321): True
is_palindrome(123): False
is_palindrome(9): True
is_palindrome(100): False
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code>
```

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.

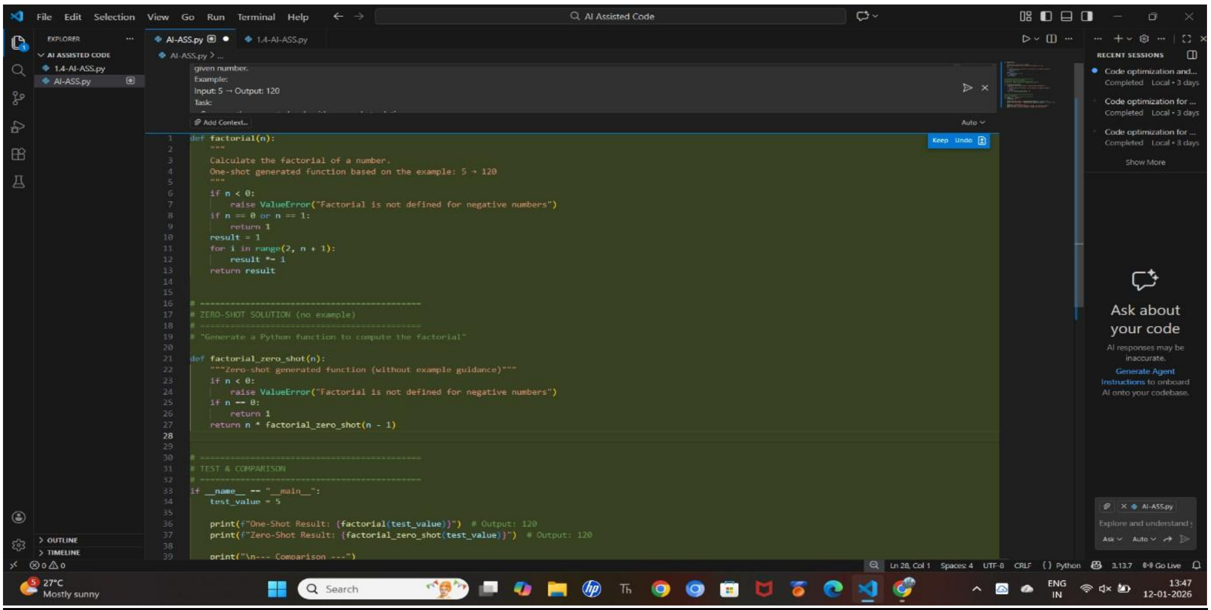
Example: Input: 5

→ Output: 120

Task:

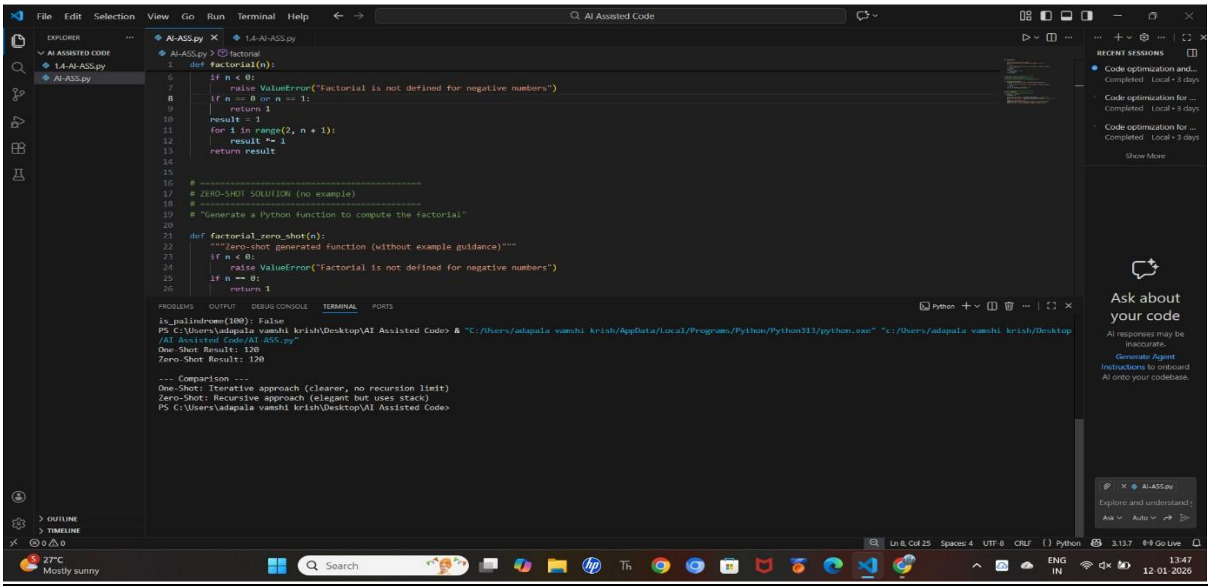
- Compare the generated code with a zero-shot solution.
- Examine improvements in clarity and correctness. **PROMPT:** **Factorial Calculation giving some instructions input data**

CODE:



given number.
Example:
Input: 5 → Output: 120
Task:

```python
def factorial(n):
    """
    Calculate the factorial of a number.
    One-shot generated function based on the example: 5 → 120
    """
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    if n == 0 or n == 1:
        return 1
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result


# ---------------------------------------------
# ZERO-SHOT SOLUTION (no example)
# ---------------------------------------------
# "Generate a Python function to compute the factorial"

def factorial_zero_shot(n):
    """Zero-shot generated function (without example guidance)"""
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    if n == 0:
        return 1
    return n * factorial_zero_shot(n - 1)


# ---------------------------------------------
# TEST & COMPARISON
# ---------------------------------------------
if __name__ == "__main__":
    test_value = 5

    print(f"One-Shot Result: {factorial(test_value)}")  # Output: 120
    print(f"Zero-Shot Result: {factorial_zero_shot(test_value)}")  # Output: 120

    print("\n--- Comparison ---")
```



```python
def factorial(n):
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    if n == 0 or n == 1:
        return 1
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result


# ---------------------------------------------
# ZERO-SHOT SOLUTION (no example)
# ---------------------------------------------
# "Generate a Python function to compute the factorial"

def factorial_zero_shot(n):
    """Zero-shot generated function (without example guidance)"""
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    if n == 0:
        return 1
```

```
is_palindrome(100): False
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code> & "C:/Users/adapala vamshi krish/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/adapala vamshi krish/Desktop
/AI Assisted Code/AI-ASS.py"
One-Shot Result: 120
Zero-Shot Result: 120

--- Comparison ---
One-Shot: Iterative approach (clearer, no recursion limit)
Zero-Shot: Recursive approach (elegant but uses stack)
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code>
```

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 • Input: 123 → Output: Not an Armstrong Number

Task:

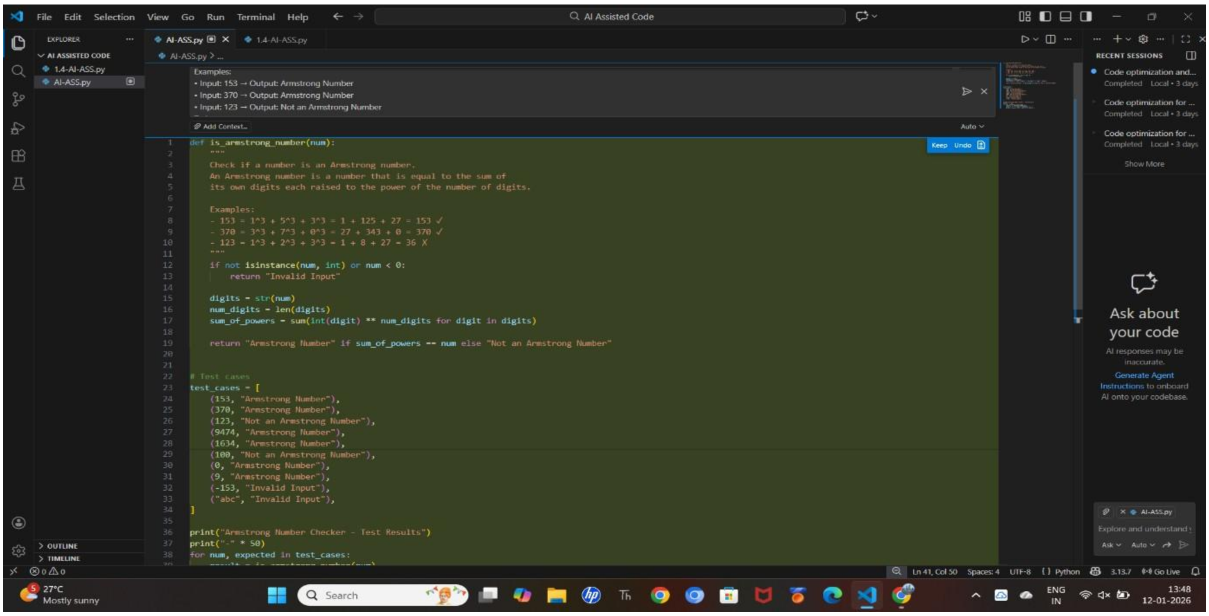
- Analyze how multiple examples influence code structure and accuracy.
- Test the function with boundary values and invalid inputs.

PROMPT: **Armstrong Number Check**

- Input: 153 → Output: Armstrong Number

- Input: 370 → Output: Armstrong Number
- Input: 123 → Output: Not an Armstrong Number

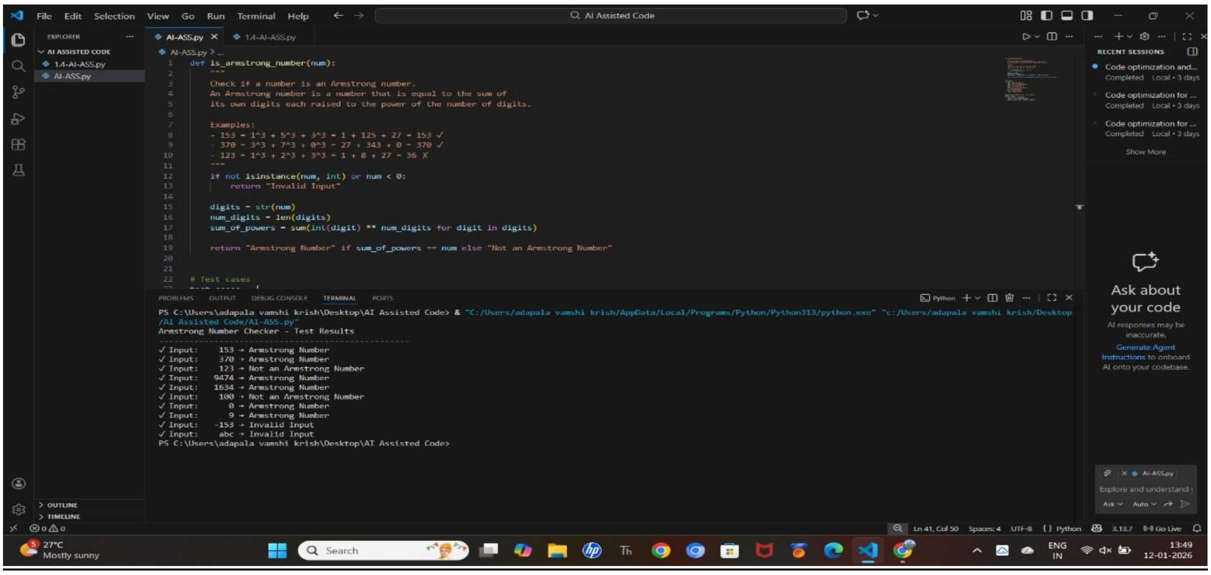
CODE:



File Edit Selection View Go Run Terminal Help
AI Assisted Code
EXPLORER
AI ASSISTED CODE
1.4-AI-ASS.py
AI-ASS.py
AI-ASS.py
Add Context...
Keep Undo
RECENT SESSIONS
Code optimization and...
Code optimization for ...
Code optimization for ...
Show More
Ask about your code
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
OUTLINE
TIMELINE
Ln 41, Col 50 Spaces: 4 UTF-8 Python 3.13.7 Go Live
27°C Mostly sunny
Search
ENG IN
13:48
12-01-2026

Examples:
- Input: 153 → Output: Armstrong Number
- Input: 370 → Output: Armstrong Number
- Input: 123 → Output: Not an Armstrong Number

```python
def is_armstrong_number(num):
    """
    Check if a number is an Armstrong number.
    An Armstrong number is a number that is equal to the sum of
    its own digits each raised to the power of the number of digits.

    Examples:
    - 153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153 ✓
    - 370 = 3^3 + 7^3 + 0^3 = 27 + 343 + 0 = 370 ✓
    - 123 = 1^3 + 2^3 + 3^3 = 1 + 8 + 27 = 36 X
    """
    if not isinstance(num, int) or num < 0:
        return "Invalid Input"

    digits = str(num)
    num_digits = len(digits)
    sum_of_powers = sum(int(digit) ** num_digits for digit in digits)

    return "Armstrong Number" if sum_of_powers == num else "Not an Armstrong Number"


# Test cases
test_cases = [
    (153, "Armstrong Number"),
    (370, "Armstrong Number"),
    (123, "Not an Armstrong Number"),
    (9474, "Armstrong Number"),
    (1634, "Armstrong Number"),
    (100, "Not an Armstrong Number"),
    (0, "Armstrong Number"),
    (9, "Armstrong Number"),
    (-153, "Invalid Input"),
    ("abc", "Invalid Input"),
]

print("Armstrong Number Checker - Test Results")
print("-" * 50)
for num, expected in test_cases:
```



File Edit Selection View Go Run Terminal Help
AI Assisted Code
EXPLORER
AI ASSISTED CODE
1.4-AI-ASS.py
AI-ASS.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
RECENT SESSIONS
Code optimization and...
Code optimization for ...
Code optimization for ...
Show More
Ask about your code
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
OUTLINE
TIMELINE
Ln 41, Col 50 Spaces: 4 UTF-8 Python 3.13.7 Go Live
27°C Mostly sunny
Search
ENG IN
13:49
12-01-2026

```python
def is_armstrong_number(num):
    """
    Check if a number is an Armstrong number.
    An Armstrong number is a number that is equal to the sum of
    its own digits each raised to the power of the number of digits.

    Examples:
    - 153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153 ✓
    - 370 = 3^3 + 7^3 + 0^3 = 27 + 343 + 0 = 370 ✓
    - 123 = 1^3 + 2^3 + 3^3 = 1 + 8 + 27 = 36 X
    """
    if not isinstance(num, int) or num < 0:
        return "Invalid Input"

    digits = str(num)
    num_digits = len(digits)
    sum_of_powers = sum(int(digit) ** num_digits for digit in digits)

    return "Armstrong Number" if sum_of_powers == num else "Not an Armstrong Number"


# Test cases
```

```
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code> & "C:/Users/adapala vamshi krish/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/adapala vamshi krish/Desktop
/AI Assisted Code/AI-ASS.py"
Armstrong Number Checker - Test Results
--------------------------------------------------
✓ Input:    153 → Armstrong Number
✓ Input:    370 → Armstrong Number
✓ Input:    123 → Not an Armstrong Number
✓ Input:   9474 → Armstrong Number
✓ Input:   1634 → Armstrong Number
✓ Input:    100 → Not an Armstrong Number
✓ Input:      0 → Armstrong Number
✓ Input:      9 → Armstrong Number
✓ Input:   -153 → Invalid Input
✓ Input:    abc → Invalid Input
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code>
```

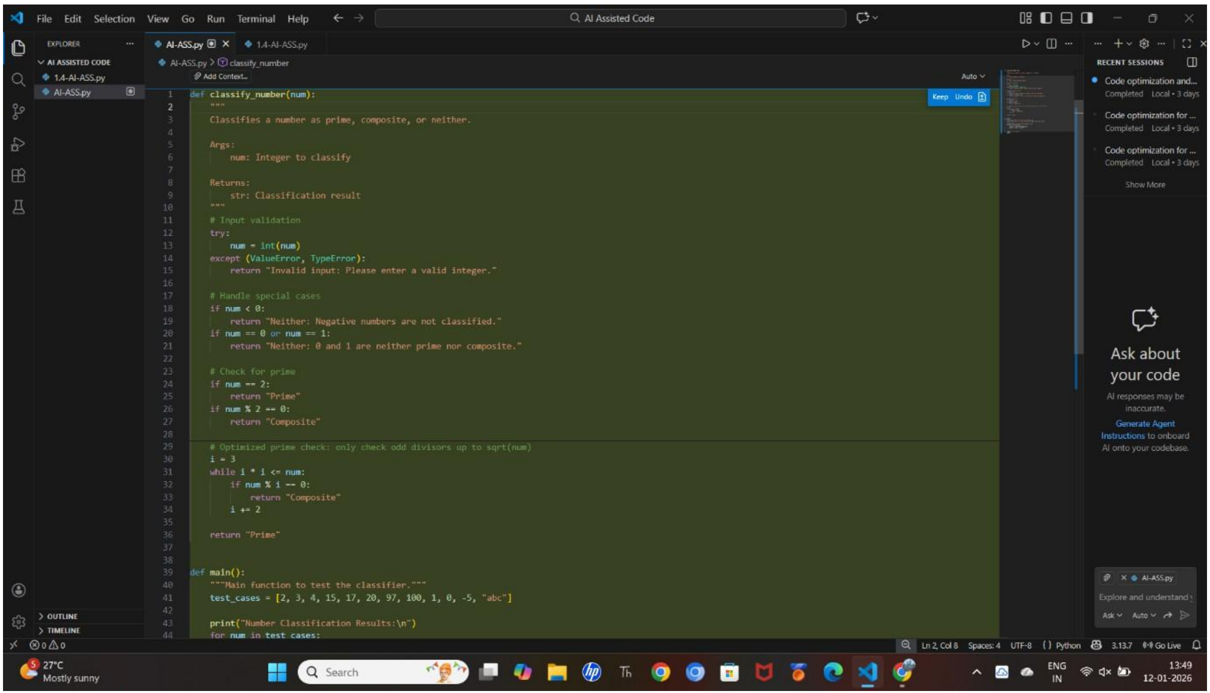
Question 4: Context-Managed Prompting (Optimized Number Classification)

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.

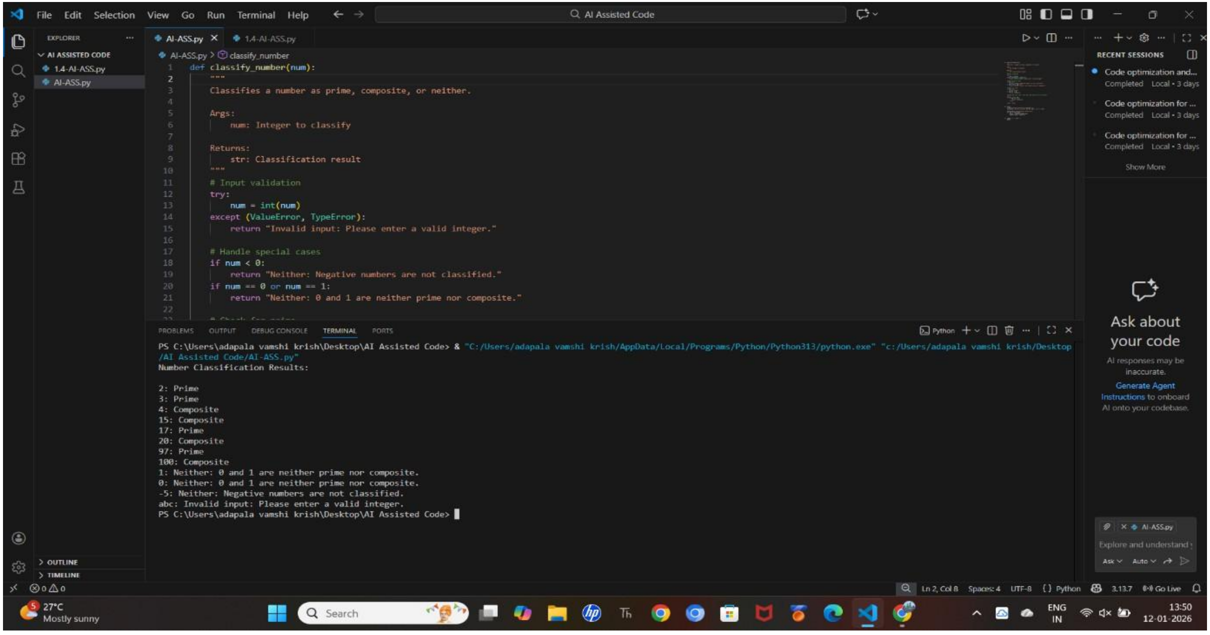
Task:

- Ensure proper input validation.
- Optimize the logic for efficiency.
- Compare the output with earlier prompting strategies.

PROMPT: **Optimized Number Classification CODE:**



```python
def classify_number(num):
    """
    Classifies a number as prime, composite, or neither.

    Args:
        num: Integer to classify

    Returns:
        str: Classification result
    """
    # Input validation
    try:
        num = int(num)
    except (ValueError, TypeError):
        return "Invalid input: Please enter a valid integer."

    # Handle special cases
    if num < 0:
        return "Neither: Negative numbers are not classified."
    if num == 0 or num == 1:
        return "Neither: 0 and 1 are neither prime nor composite."

    # Check for prime
    if num == 2:
        return "Prime"
    if num % 2 == 0:
        return "Composite"

    # Optimized prime check: only check odd divisors up to sqrt(num)
    i = 3
    while i * i <= num:
        if num % i == 0:
            return "Composite"
        i += 2

    return "Prime"


def main():
    """Main function to test the classifier."""
    test_cases = [2, 3, 4, 15, 17, 20, 97, 100, 1, 0, -5, "abc"]

    print("Number Classification Results:\n")
    for num in test_cases:
```



```python
def classify_number(num):
    """
    Classifies a number as prime, composite, or neither.

    Args:
        num: Integer to classify

    Returns:
        str: Classification result
    """
    # Input validation
    try:
        num = int(num)
    except (ValueError, TypeError):
        return "Invalid input: Please enter a valid integer."

    # Handle special cases
    if num < 0:
        return "Neither: Negative numbers are not classified."
    if num == 0 or num == 1:
        return "Neither: 0 and 1 are neither prime nor composite."
```

```
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code> & "C:/Users/adapala vamshi krish/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/adapala vamshi krish/Desktop
/AI Assisted Code/AI-ASS.py"
Number Classification Results:

2: Prime
3: Prime
4: Composite
15: Composite
17: Prime
20: Composite
97: Prime
100: Composite
1: Neither: 0 and 1 are neither prime nor composite.
0: Neither: 0 and 1 are neither prime nor composite.
-5: Neither: Negative numbers are not classified.
abc: Invalid input: Please enter a valid integer.
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code>
```

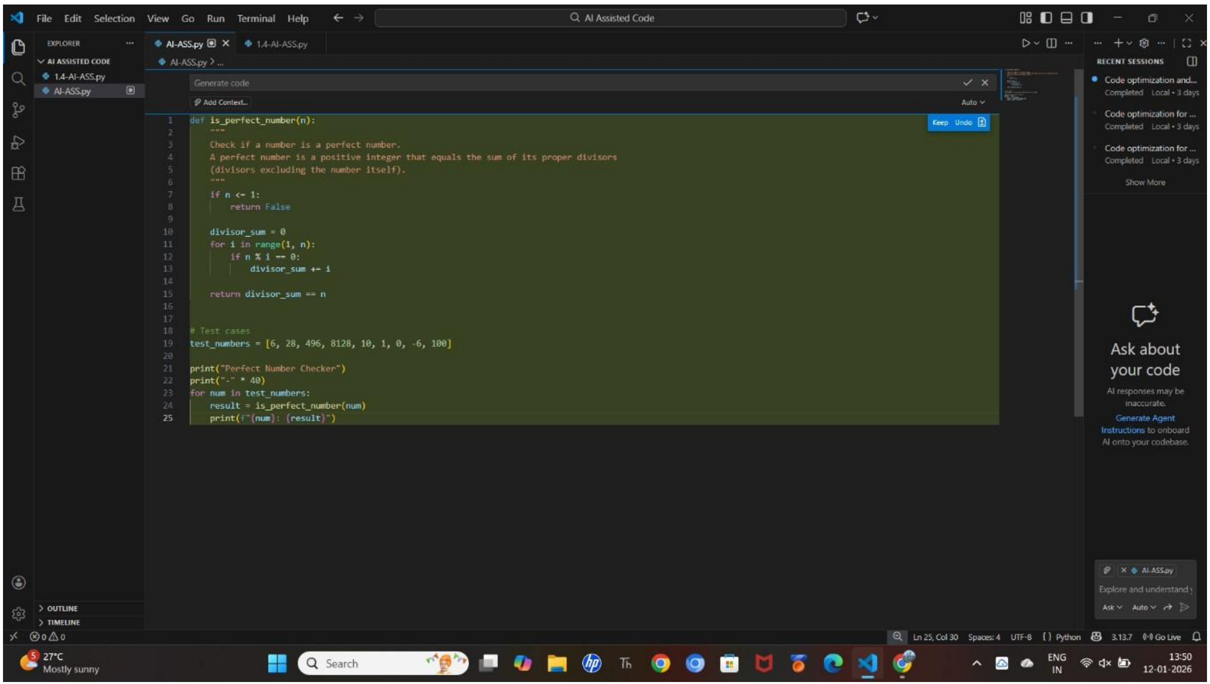
Question 5: Zero-Shot Prompting (Perfect Number Check)

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

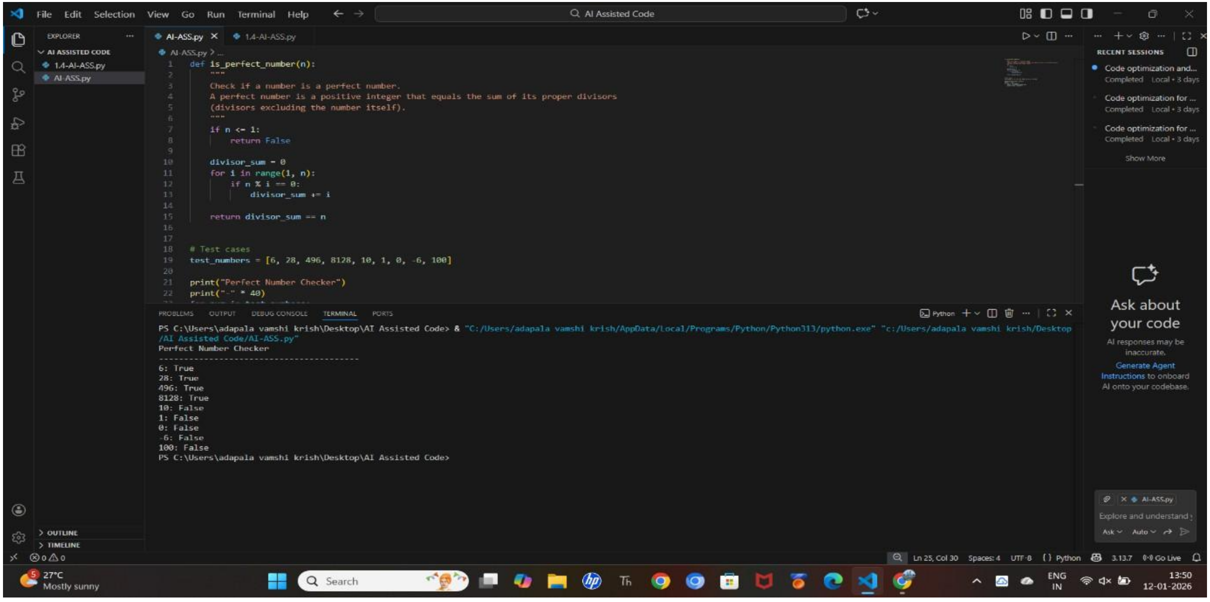
Task:

- Record the AI-generated code.
- Test the program with multiple inputs.
- Identify any missing conditions or inefficiencies in the logic.

<u>PROMPT:</u> Perfect Number Check without providing any examples <u>CODE:</u>



```python
def is_perfect_number(n):
    """
    Check if a number is a perfect number.
    A perfect number is a positive integer that equals the sum of its proper divisors
    (divisors excluding the number itself).
    """
    if n <= 1:
        return False

    divisor_sum = 0
    for i in range(1, n):
        if n % i == 0:
            divisor_sum += i

    return divisor_sum == n


# Test cases
test_numbers = [6, 28, 496, 8128, 10, 1, 0, -6, 100]

print("Perfect Number Checker")
print("-" * 40)
for num in test_numbers:
    result = is_perfect_number(num)
    print(f"{num}: {result}")
```



```
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code> & "C:/Users/adapala vamshi krish/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/adapala vamshi krish/Desktop
/AI Assisted Code/AI-ASS.py"
Perfect Number Checker
----------------------------------------
6: True
28: True
496: True
8128: True
10: False
1: False
0: False
-6: False
100: False
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code>
```

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

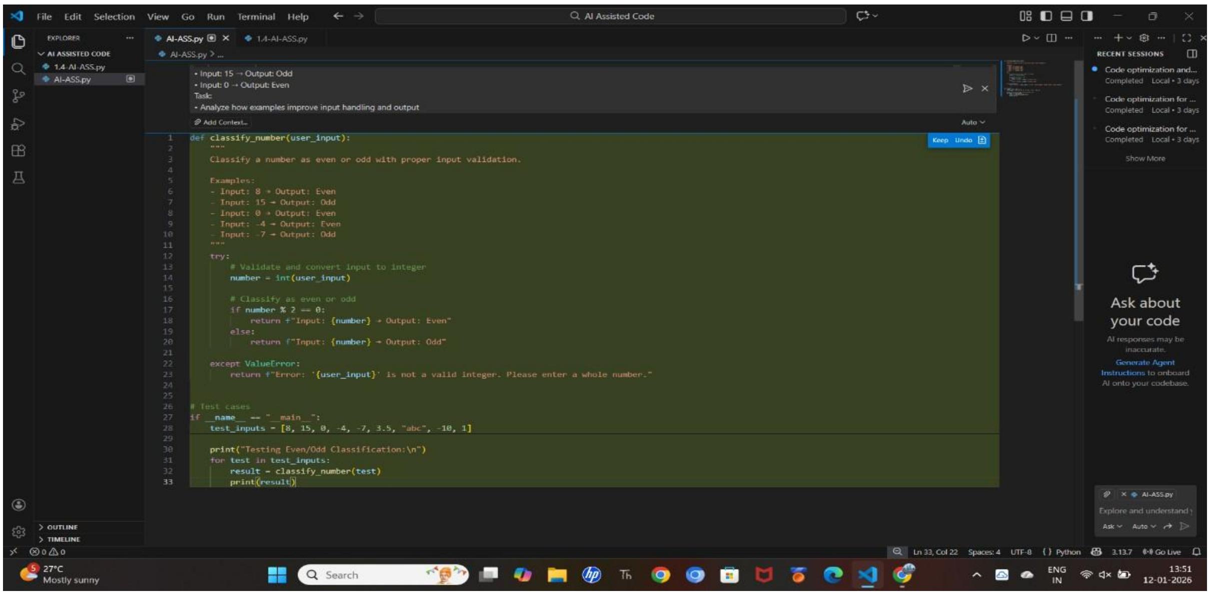
Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python program that determines whether a given number is even or odd, including proper input validation.
Examples:
- Input: 8 → Output: Even
- Input: 15 → Output: Odd • Input: 0 → Output: Even **Task:**
- Analyze how examples improve input handling and output clarity.
- Test the program with negative numbers and non-integer inputs.

PROMPT:

Even or Odd Classification with Validation

- Input: 8 → Output: Even
- Input: 15 → Output: Odd
- Input: 0 → Output: Even

CODE:

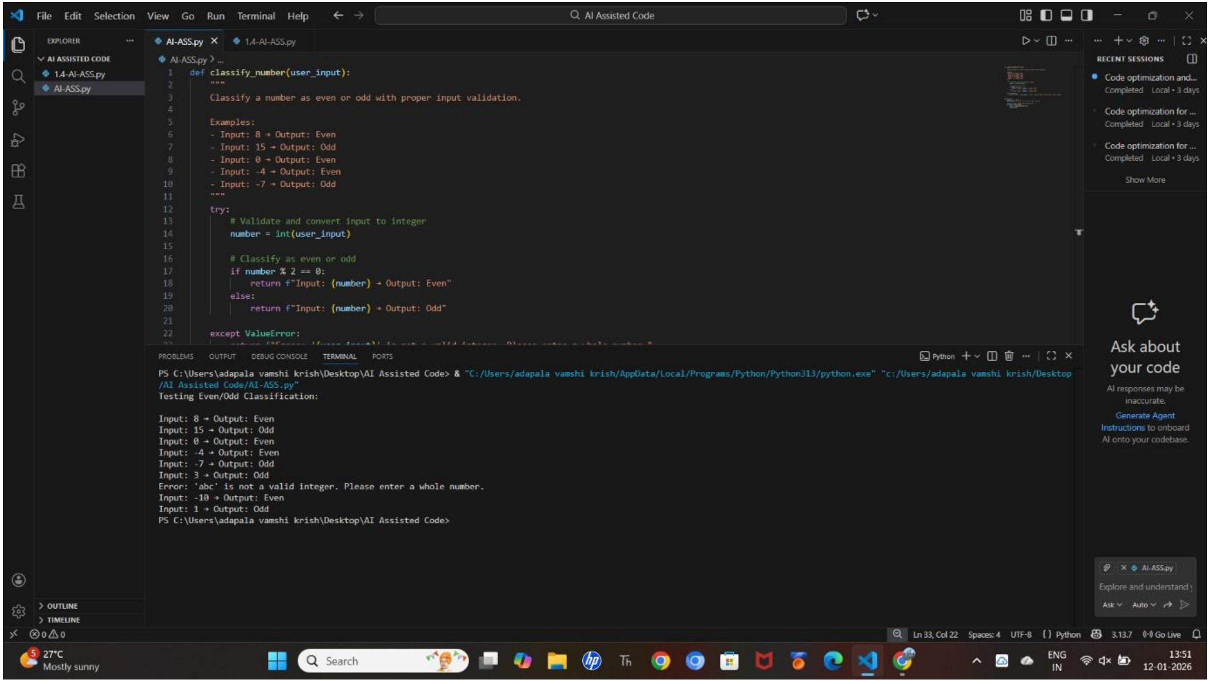


- Input: 15 → Output: Odd
- Input: 0 → Output: Even

Task:

- Analyze how examples improve input handling and output

```python
def classify_number(user_input):
    """
    Classify a number as even or odd with proper input validation.

    Examples:
    - Input: 8 → Output: Even
    - Input: 15 → Output: Odd
    - Input: 0 → Output: Even
    - Input: -4 → Output: Even
    - Input: -7 → Output: Odd
    """
    try:
        # Validate and convert input to integer
        number = int(user_input)

        # Classify as even or odd
        if number % 2 == 0:
            return f"Input: {number} → Output: Even"
        else:
            return f"Input: {number} → Output: Odd"

    except ValueError:
        return f"Error: '{user_input}' is not a valid integer. Please enter a whole number."


# Test cases
if __name__ == "__main__":
    test_inputs = [8, 15, 0, -4, -7, 3.5, "abc", -10, 1]

    print("Testing Even/Odd Classification:\n")
    for test in test_inputs:
        result = classify_number(test)
        print(result)
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Ask about your code
Ln 33, Col 22 Spaces: 4 UTF-8 Python 3.13.7 Go Live
13:51 12-01-2026

```
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code> & "C:/Users/adapala vamshi krish/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/adapala vamshi krish/Desktop
/AI Assisted Code/AI-ASS.py"
Testing Even/Odd Classification:

Input: 8 → Output: Even
Input: 15 → Output: Odd
Input: 0 → Output: Even
Input: -4 → Output: Even
Input: -7 → Output: Odd
Input: 3 → Output: Odd
Error: 'abc' is not a valid integer. Please enter a whole number.
Input: -10 → Output: Even
Input: 1 → Output: Odd
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code>
```