AI ASSISTED CODING

LAB-3.1

Katta Lasya

2303A51724

Batch-11

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.

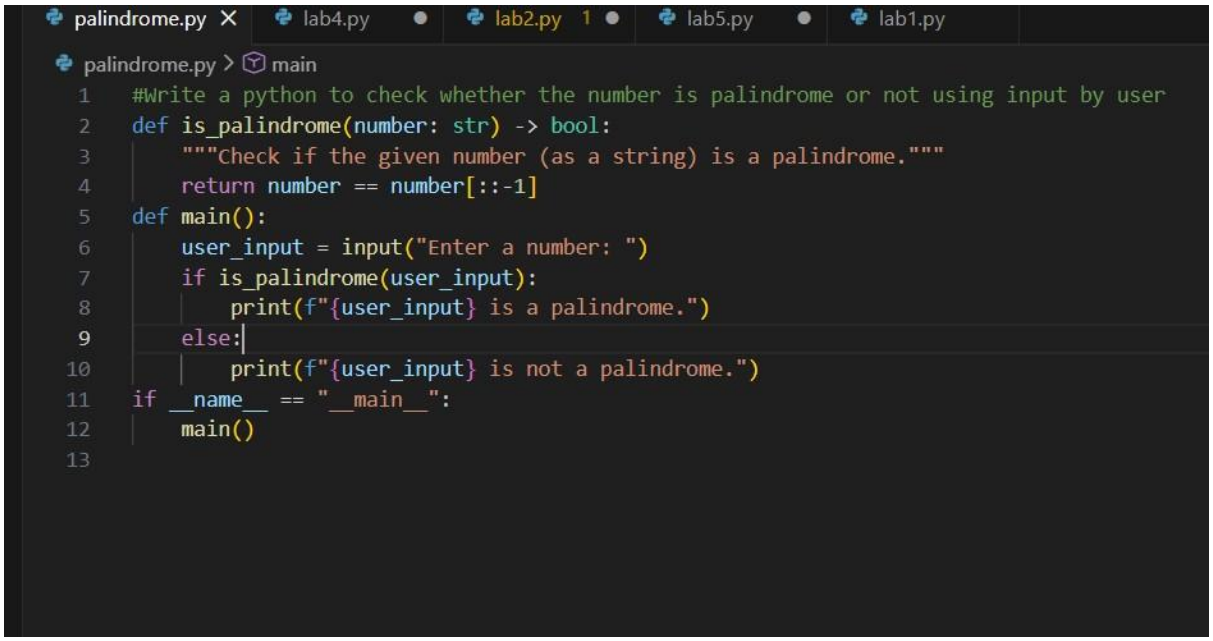
Task:

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

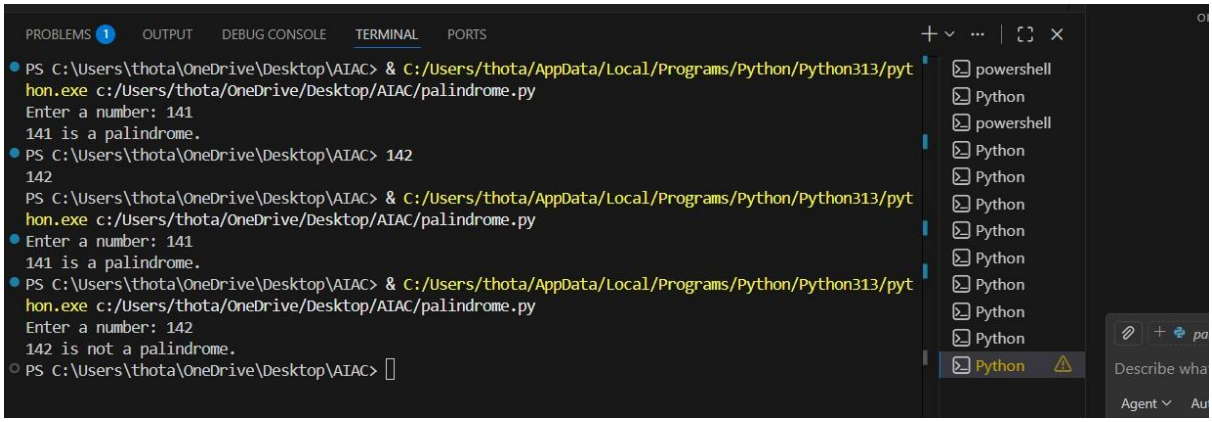
PROMPT:

#Write a python to check whether the number is palindrome or not using input by user

CODE:



OUTPUT:



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:

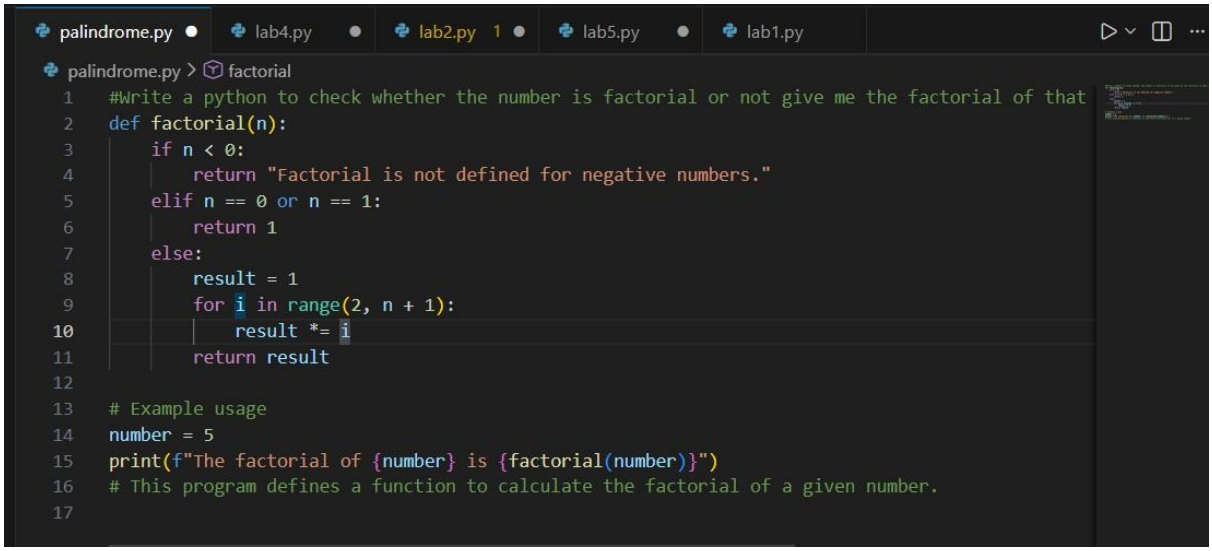
Input: 5 → Output: 120

Task:

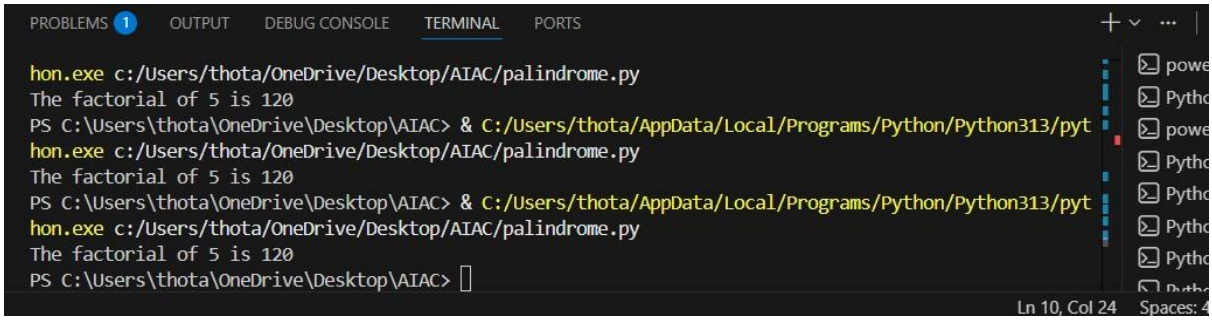
- Compare the generated code with a zero-shot solution.
- Examine improvements in clarity and correctness.

PROMPT:

#Write a python to check whether the number is factorial or not give me the factorial of that number example input=5 and output=120 **CODE:**



OUTPUT:



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:

#write a code to generate armstrong number checking functions with comments

CODE:

palindrome.py | lab4.py | lab2.py 1 | lab5.py | lab1.py

lab1.py > is_armstrong_number

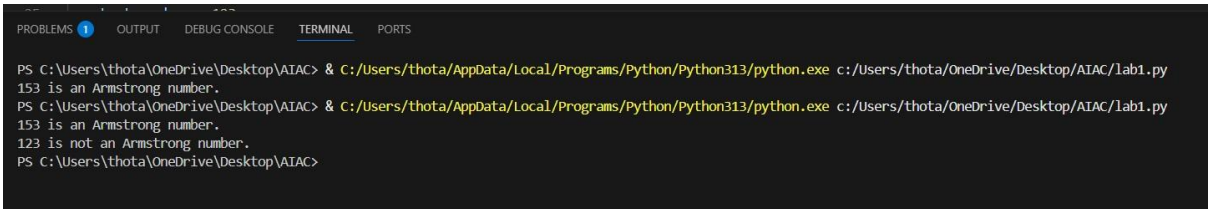
```python
#write a code to generate armstrong number checking functions with comments
def is_armstrong_number(num):
    """
    This function checks if a number is an Armstrong number.
    An Armstrong number is a number that is equal to the sum of its own digits
    each raised to the power of the number of digits.
    For example, 153 is an Armstrong number because 1^3 + 5^3 + 3^3 = 153.
    """
    # Convert the number to string to easily iterate over digits
    digits = str(num)
    num_digits = len(digits)

    # Calculate the sum of each digit raised to the power of num_digits
    sum_of_powers = sum(int(digit) ** num_digits for digit in digits)

    # Check if the sum of powers is equal to the original number
    return sum_of_powers == num
# Example usage
if __name__ == "__main__":
    test_number = 153
    if is_armstrong_number(test_number):
        print(f"{test_number} is an Armstrong number.")
    else:
        print(f"{test_number} is not an Armstrong number.")
    test_number = 123
    if is_armstrong_number(test_number):
        print(f"{test_number} is an Armstrong number.")
    else:
        print(f"{test_number} is not an Armstrong number.")

```

OUTPUT:



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:

#Write a python code to check whether the number classifies as prime,composite or neither using input by user **CODE:**

```python
#Write a python code to check whether the number classifies as prime,composite or neither using input by user
def classify_number(num):
    """
    This function classifies a number as 'prime', 'composite', or 'neither'.
    A prime number is greater than 1 and has no divisors other than 1 and itself.
    A composite number is greater than 1 and has more than two divisors.
    Numbers less than or equal to 1 are classified as 'neither'.
    """
    if num <= 1:
        return "neither"
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return "composite"
    return "prime"
# Example usage
if __name__ == "__main__":
    user_input = int(input("Enter a number to classify: "))
    classification = classify_number(user_input)
    print(f"The number {user_input} is classified as: {classification}")
# This program defines a function to classify a number as prime, composite, or neither.

```

OUTPUT:

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

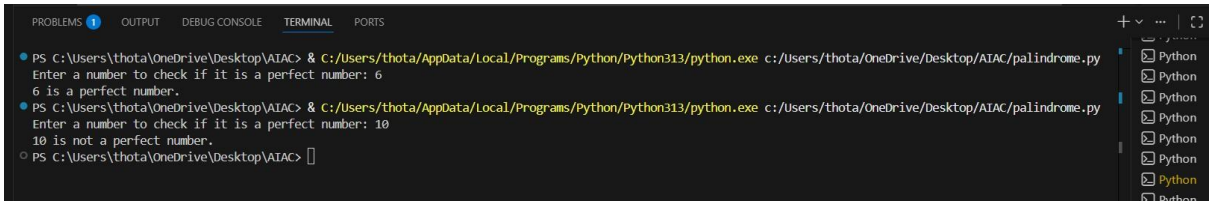
PROMPT:

#Write a python code to check whether the number is perfect number or not with inputbyuser **CODE:**

```python
#Write a python code to check whether the number is perfect number or not with inputbyuser
def is_perfect_number(n: int) -> bool:
    """Check if a number is a perfect number."""
    if n < 1:
        return False
    divisors_sum = sum(i for i in range(1, n) if n % i == 0)
    return divisors_sum == n
def main():

    try:
        number = int(input("Enter a number to check if it is a perfect number: "))
        if is_perfect_number(number):
            print(f"{number} is a perfect number.")
        else:
            print(f"{number} is not a perfect number.")
    except ValueError:
        print("Invalid input. Please enter a positive integer.")
if __name__ == "__main__":
    main()
```

OUTPUT:



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:

"""Write a python code to check whether the number is even or odd

Examples Input:8,Output:Even

Input:15,Output:odd

Input:0,Output:Even"""

CODE:

palindrome.py | lab4.py | lab2.py 1 | lab5.py | lab1.py

palindrome.py > ...

```python
"""Write a python code to check whether the number is even or odd
Examples Input:8,Output:Even
Input:15,Output:odd
Input:0,Output:Even"""
def check_even_odd(number: int) -> str:
    if number % 2 == 0:
        return "Even"
    else:
        return "Odd"
if __name__ == "__main__":
    test_numbers = [8, 15, 0]
    for num in test_numbers:
        result = check_even_odd(num)
        print(f"Input:{num}, Output:{result}")
# This program checks whether a given number is even or odd and prints the result.
# Examples:
```

OUTPUT: