Lab-2.2

Course : AI Assisted Coding

Topic:- Exploring Additional AI Coding Tools beyond Copilot – Gemini (Colab) and Cursor AI

Student Details:

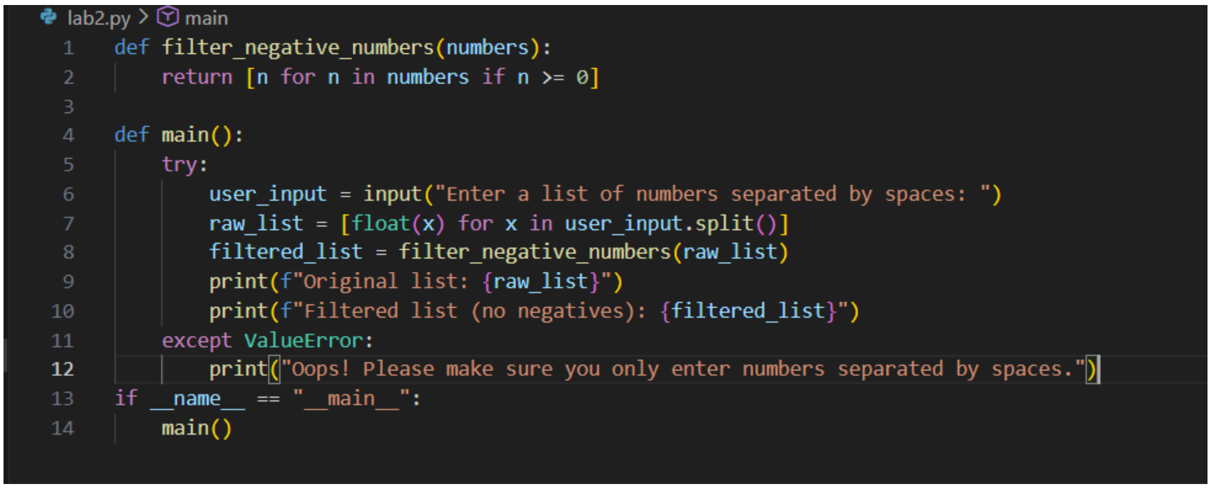
Name	:	Mohammed Sabir
Hall Ticket No	:	2303A51506
Batch	:	22

Task-01

Final Optimal Prompt:-

write a python program to filter all negative number in list as in function taker user input

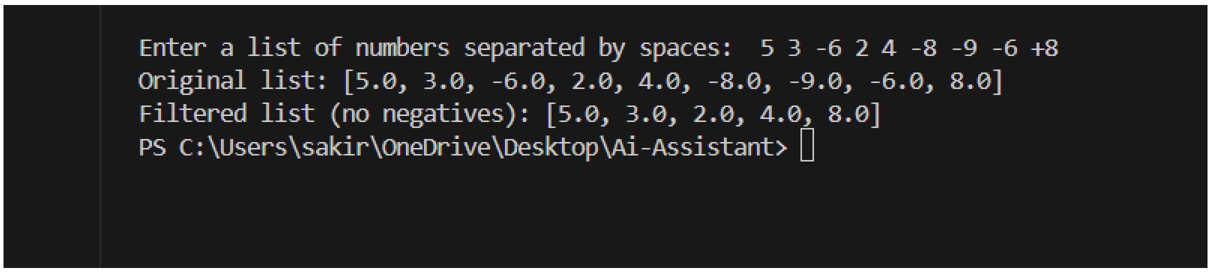
Code Screenshot :-



lab2.py > main

```python
def filter_negative_numbers(numbers):
    return [n for n in numbers if n >= 0]

def main():
    try:
        user_input = input("Enter a list of numbers separated by spaces: ")
        raw_list = [float(x) for x in user_input.split()]
        filtered_list = filter_negative_numbers(raw_list)
        print(f"Original list: {raw_list}")
        print(f"Filtered list (no negatives): {filtered_list}")
    except ValueError:
        print("Oops! Please make sure you only enter numbers separated by spaces.")
if __name__ == "__main__":
    main()
```

Output Screenshot:-



```
Enter a list of numbers separated by spaces:  5 3 -6 2 4 -8 -9 -6 +8
Original list: [5.0, 3.0, -6.0, 2.0, 4.0, -8.0, -9.0, -6.0, 8.0]
Filtered list (no negatives): [5.0, 3.0, 2.0, 4.0, 8.0]
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> 
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence) :-

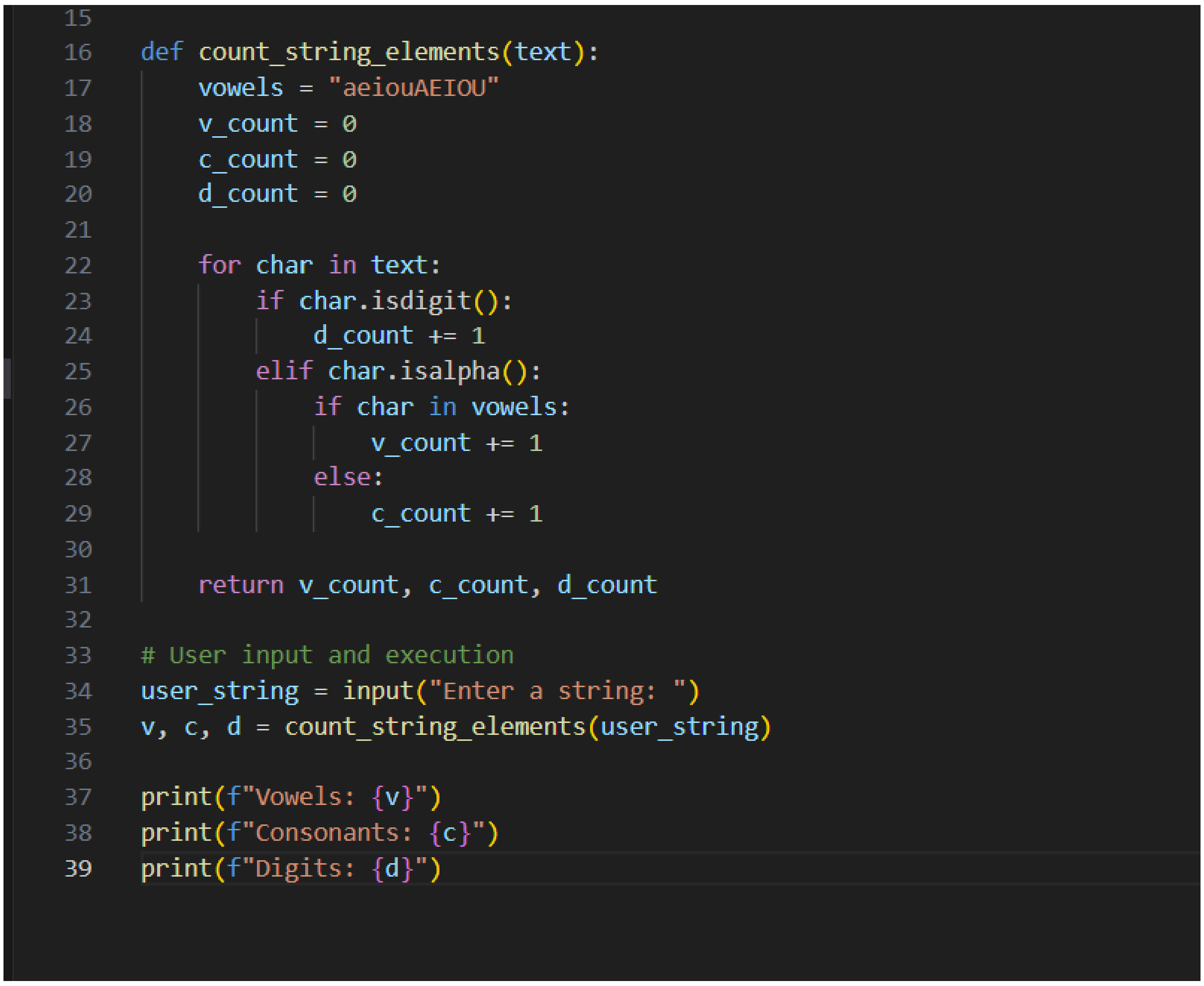
The program begins by capturing the user's input as a single string and splitting it into individual components based on the spaces between them. These individual strings are then converted into numerical values, typically floats or integers, so that the computer can interpret them as actual numbers rather than just text characters. To perform the filtering, the code uses a list comprehension that examines every number in the sequence and checks if it is greater than or equal to zero. If the number satisfies this non-negative condition, it is added to a new list, while any negative values are excluded from the final result. The process concludes by displaying this refined list to the user, providing a clean output that contains only the positive values and zero

Task-02

Final Optimal Prompt:-

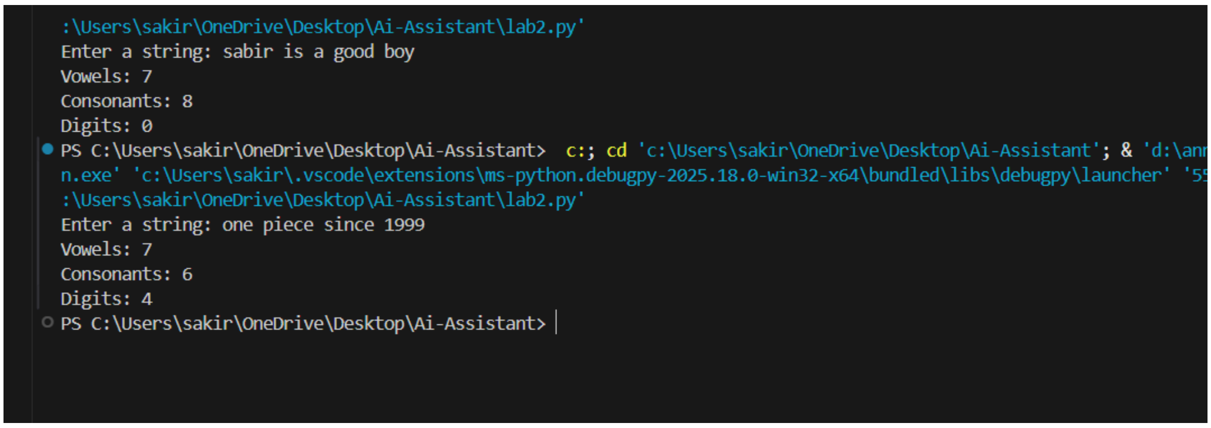
write a python program in function that counts vowel, consonant , and digits in string

Code Screenshot :-



```python

def count_string_elements(text):
    vowels = "aeiouAEIOU"
    v_count = 0
    c_count = 0
    d_count = 0

    for char in text:
        if char.isdigit():
            d_count += 1
        elif char.isalpha():
            if char in vowels:
                v_count += 1
            else:
                c_count += 1

    return v_count, c_count, d_count

# User input and execution
user_string = input("Enter a string: ")
v, c, d = count_string_elements(user_string)

print(f"Vowels: {v}")
print(f"Consonants: {c}")
print(f"Digits: {d}")
```

Output Screenshot:-



```
:\Users\sakir\OneDrive\Desktop\Ai-Assistant\lab2.py'
Enter a string: sabir is a good boy
Vowels: 7
Consonants: 8
Digits: 0
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant>  c:; cd 'c:\Users\sakir\OneDrive\Desktop\Ai-Assistant'; & 'd:\ann
n.exe' 'c:\Users\sakir\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55
:\Users\sakir\OneDrive\Desktop\Ai-Assistant\lab2.py'
Enter a string: one piece since 1999
Vowels: 7
Consonants: 6
Digits: 4
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> 
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence) :-

It starts by setting up three counters at zero and creating a reference list of all the vowels so it knows what to look for. As it moves through your string, it first asks if a character is a number; if so, it adds one to the digit count. If the character is a letter, it then checks if that letter is in its vowel list; if it finds a match, the vowel count goes up, and if not, it assumes the letter is a consonant. Because it uses these specific checks, things like spaces or punctuation marks are naturally ignored, and once the program reaches the end of your string, it simply hands you back the final totals for each category.

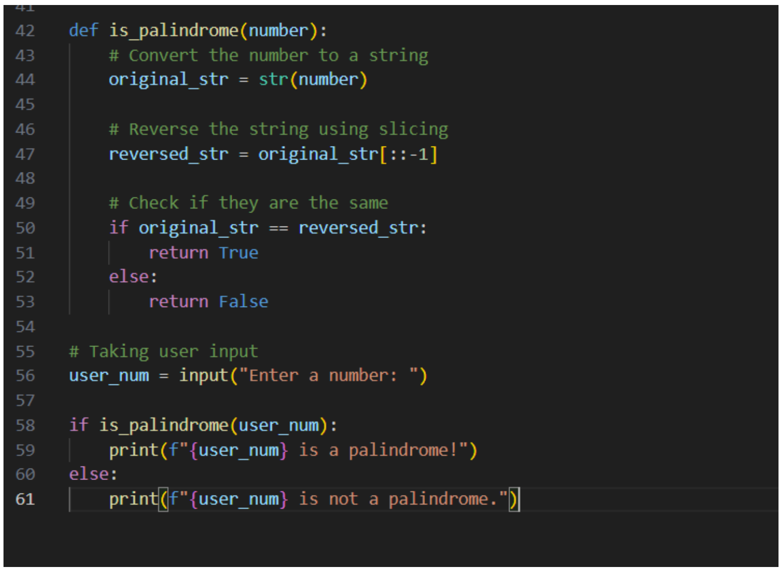
Task-03

Final Optimal Prompt:-

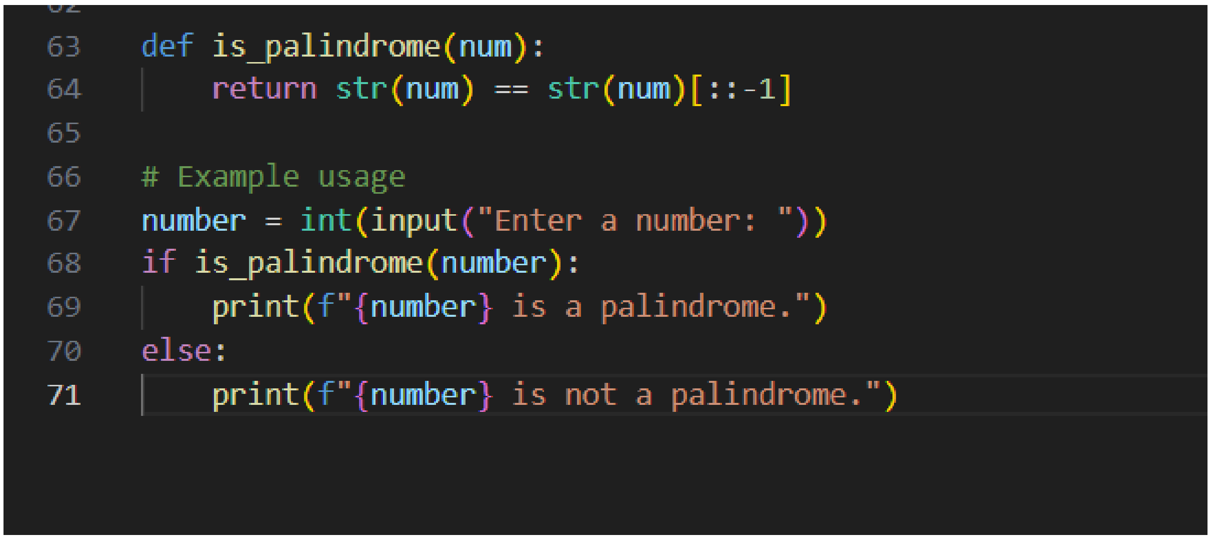
write a python program to check whether the number is a palindrome or not?

write a python program to check whether the number is a palindrome or not?

Code Screenshot :



```python
def is_palindrome(number):
    # Convert the number to a string
    original_str = str(number)

    # Reverse the string using slicing
    reversed_str = original_str[::-1]

    # Check if they are the same
    if original_str == reversed_str:
        return True
    else:
        return False

# Taking user input
user_num = input("Enter a number: ")

if is_palindrome(user_num):
    print(f"{user_num} is a palindrome!")
else:
    print(f"{user_num} is not a palindrome.")
```



```python
def is_palindrome(num):
    return str(num) == str(num)[::-1]

# Example usage
number = int(input("Enter a number: "))
if is_palindrome(number):
    print(f"{number} is a palindrome.")
else:
    print(f"{number} is not a palindrome.")
```

Output Comparision:

Feature	First Code (Detailed)	Second Code (Concise)
Logic Style	Step-by-Step: Breaks every action into a new line.	Compact: Combines conversion and check in one line.
Return Type	Uses explicit return True and return False statements.	Returns the result of a comparison directly (Boolean).
Input Type	Handles input as a String (works for words and numbers).	Converts input to an Integer (strict numerical use).
Readability	High for beginners; easy to follow the flow.	High for experienced users; less "clutter" in the code.
Error Handling	Flexible; won't crash if letters are typed.	Will crash if the user enters non-numeric text.
Code Length	~10 lines of logic.	~2 lines of logic.

Explanation/Justification/Observation (100 words / 5 – 6 sentence) :-

The first code is designed to be highly descriptive by breaking the task into clear, sequential steps that make it easy for a beginner to trace how the data is being converted and reversed. In contrast, the second code follows a professional "Pythonic" style by condensing the entire logical check into a single, efficient line that returns the result of the comparison immediately. While the first version is more flexible because it treats the input as a string—allowing it to check both words and numbers—the second version is more restrictive because it converts the input into an integer, which causes an error if a user types anything other than a digit. The first approach relies on a manual if-else structure to return a result, whereas the second approach uses the inherent logic of an equality check to produce a true or false value automatically. Ultimately, choosing between them is a matter of prioritizing either the ease of reading for a human student or the compact efficiency preferred in modern software development.

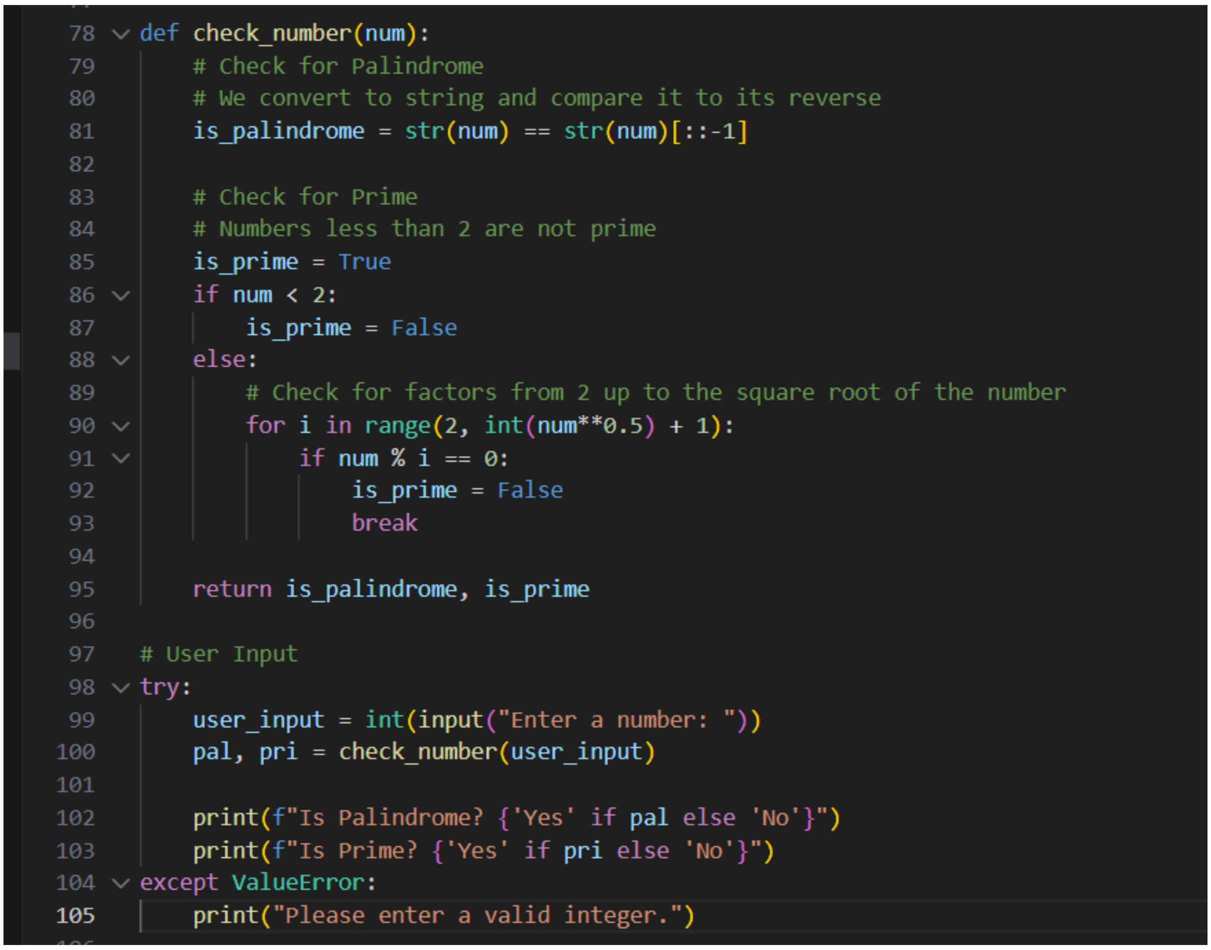
Task-04

Final Optimal Prompt:-

write a python program to check the number given in prime or palindrome in function

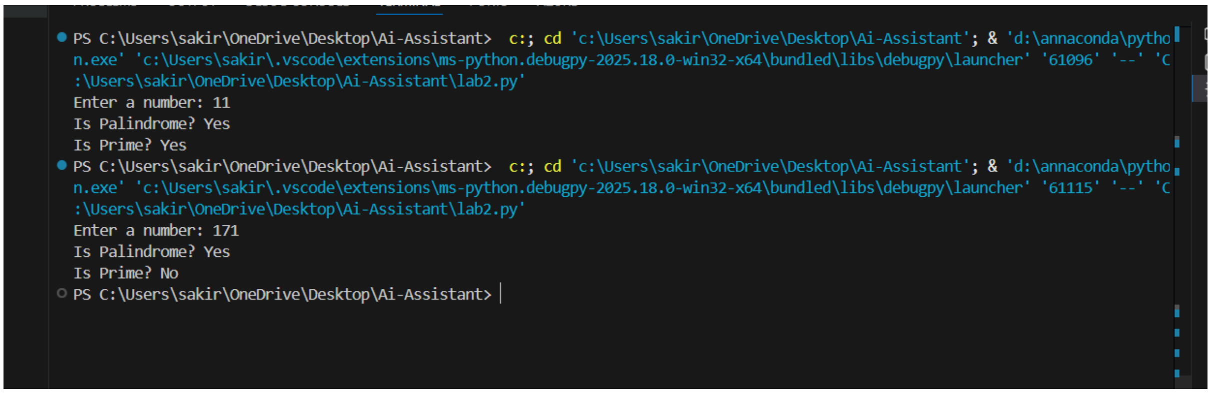
and explain

Code Screenshot :-



```python
def check_number(num):
    # Check for Palindrome
    # We convert to string and compare it to its reverse
    is_palindrome = str(num) == str(num)[::-1]

    # Check for Prime
    # Numbers less than 2 are not prime
    is_prime = True
    if num < 2:
        is_prime = False
    else:
        # Check for factors from 2 up to the square root of the number
        for i in range(2, int(num**0.5) + 1):
            if num % i == 0:
                is_prime = False
                break

    return is_palindrome, is_prime

# User Input
try:
    user_input = int(input("Enter a number: "))
    pal, pri = check_number(user_input)

    print(f"Is Palindrome? {'Yes' if pal else 'No'}")
    print(f"Is Prime? {'Yes' if pri else 'No'}")
except ValueError:
    print("Please enter a valid integer.")
```

Output Screenshot:-



```
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant>  c:; cd 'c:\Users\sakir\OneDrive\Desktop\Ai-Assistant'; & 'd:\annaconda\pytho
n.exe' 'c:\Users\sakir\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61096' '--' 'c
:\Users\sakir\OneDrive\Desktop\Ai-Assistant\lab2.py'
Enter a number: 11
Is Palindrome? Yes
Is Prime? Yes
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant>  c:; cd 'c:\Users\sakir\OneDrive\Desktop\Ai-Assistant'; & 'd:\annaconda\pytho
n.exe' 'c:\Users\sakir\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61115' '--' 'c
:\Users\sakir\OneDrive\Desktop\Ai-Assistant\lab2.py'
Enter a number: 171
Is Palindrome? Yes
Is Prime? No
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant>
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence):-

The program works by passing your number into a function that handles two different mathematical tests. For the palindrome check, it converts the number to a string and uses Python's slicing trick to see if it matches its mirrored version.

For the prime check, the program first rules out any numbers less than 2. It then uses a for loop to try dividing your number by every integer starting from 2 up to the square root of the number; if any of these divisions result in a remainder of zero, the program flags the number as "not prime" and stops looking. Finally, it returns both results to the main part of the code, which prints a clear "Yes" or "No" for each category.
