

NAME: B.VYSHNAVI

COURSE NAME : AIAC

2303A51884

WEEK - 3

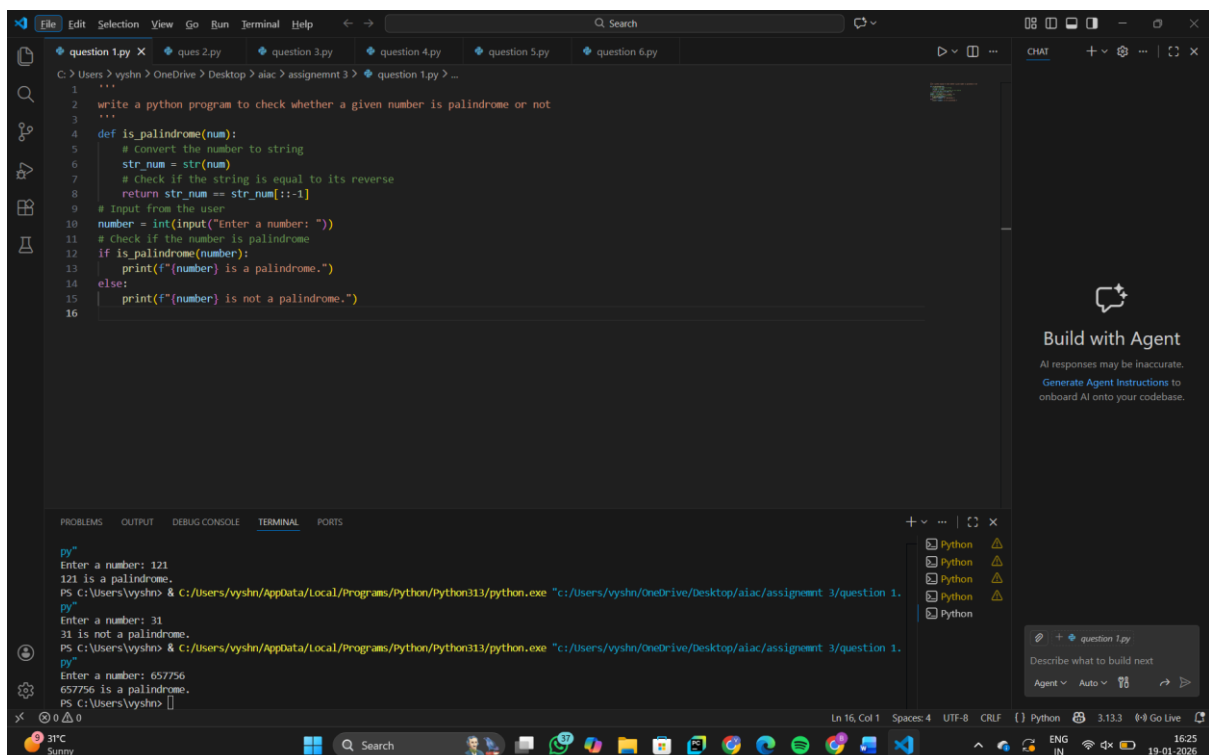
BATCH-02

### 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.



The screenshot shows a Visual Studio Code editor with a Python file named 'question 1.py'. The code defines a function 'is\_palindrome' that converts a number to a string, checks if it equals its reverse, and returns a boolean. It then prompts the user for a number and prints the result. The terminal at the bottom shows the program being executed with three test cases: 121 (palindrome), 31 (not a palindrome), and 657756 (palindrome). The status bar at the bottom indicates the file is at line 16, column 1, with 4 spaces, using UTF-8 encoding and CRLF line endings, and is running Python 3.11.3.

```
1 """
2 write a python program to check whether a given number is palindrome or not
3 """
4 def is_palindrome(num):
5     # Convert the number to string
6     str_num = str(num)
7     # Check if the string is equal to its reverse
8     return str_num == str_num[::-1]
9 # Input from the user
10 number = int(input("Enter a number: "))
11 # Check if the number is palindrome
12 if is_palindrome(number):
13     print(f"{number} is a palindrome.")
14 else:
15     print(f"{number} is not a palindrome.")
16
```

py  
Enter a number: 121  
121 is a palindrome.  
PS C:\Users\vyshn> & C:\Users\vyshn\AppData\Local\Programs\Python\Python311\python.exe "c:/Users/vyshn/OneDrive/Desktop/aiac/assignment 3/question 1.py"  
py  
Enter a number: 31  
31 is not a palindrome.  
PS C:\Users\vyshn> & C:\Users\vyshn\AppData\Local\Programs\Python\Python311\python.exe "c:/Users/vyshn/OneDrive/Desktop/aiac/assignment 3/question 1.py"  
py  
Enter a number: 657756  
657756 is a palindrome.  
PS C:\Users\vyshn>

## Question 2: One-Shot Prompting (Factorial Calculation)

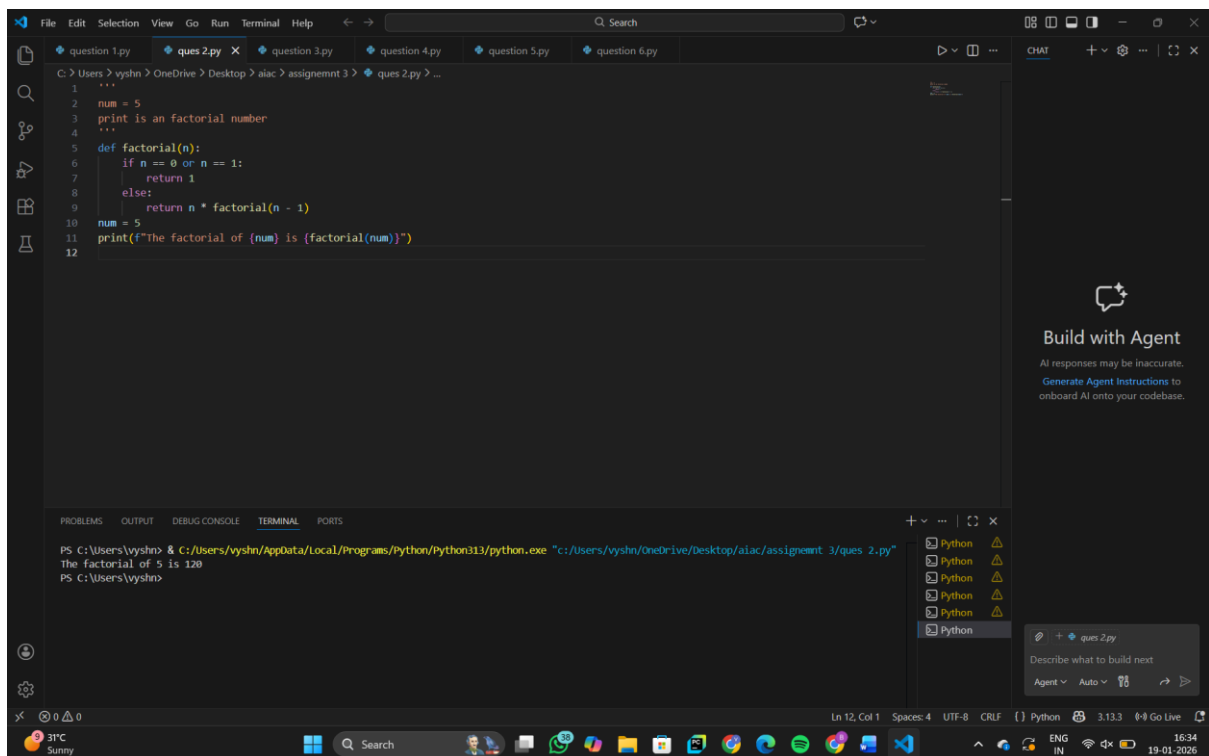
Write a one-shot prompt by providing one input-output example and ask the AI to generate a Python function to compute the factorial of a given number.

Example:

Input: 5 → Output: 120

Task:

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



The screenshot shows a Visual Studio Code editor window with a Python file named 'ques 2.py'. The code defines a recursive factorial function and prints the result for the input 5. The terminal output shows the command to run the script and the resulting output: 'The factorial of 5 is 120'.

```
1 """
2 num = 5
3 print is an factorial number
4 """
5 def factorial(n):
6     if n == 0 or n == 1:
7         return 1
8     else:
9         return n * factorial(n - 1)
10 num = 5
11 print(f"The factorial of {num} is {factorial(num)}")
12
```

Terminal Output:

```
PS C:\Users\vyshn> & C:/Users/vyshn/AppData/Local/Programs/Python/python313/python.exe "c:/Users/vyshn/OneDrive/Desktop/aia/assignment 3/ques 2.py"
The factorial of 5 is 120
PS C:\Users\vyshn>
```

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

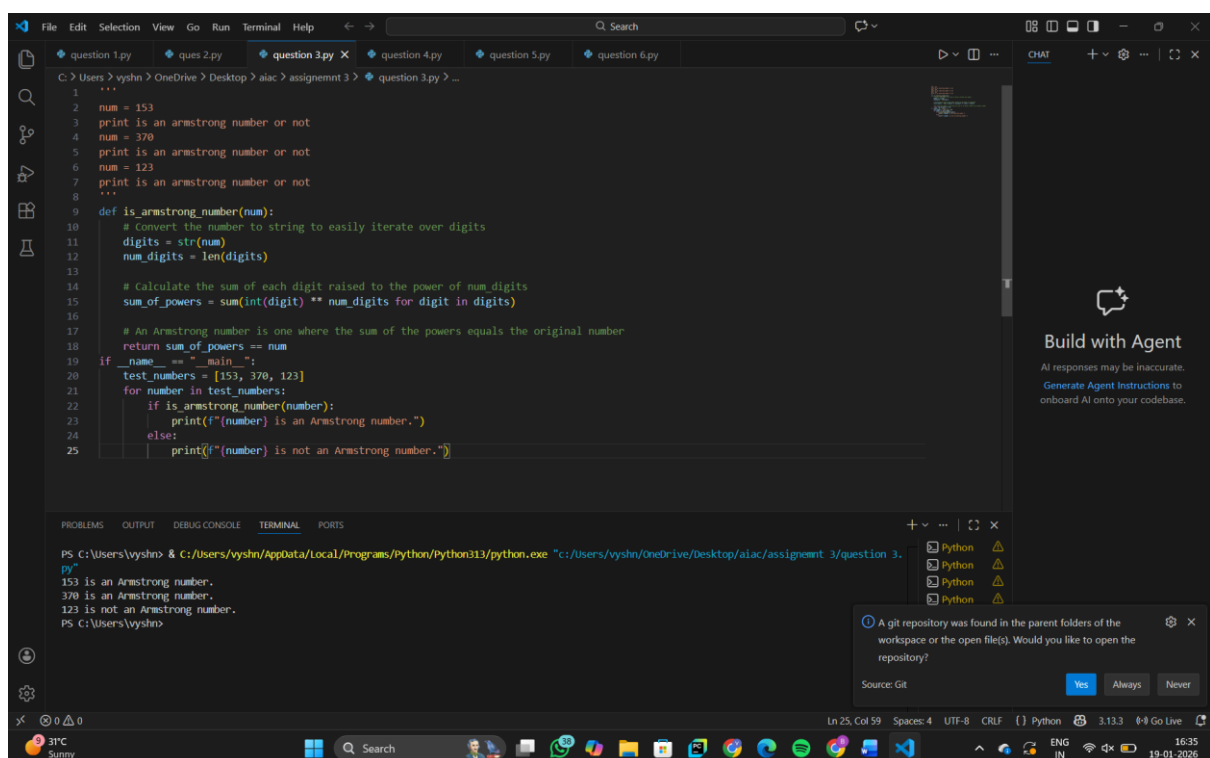
Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python function to check whether a given number is an Armstrong number.

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.



The screenshot shows a Visual Studio Code editor with a Python file named 'question 3.py'. The code defines a function `is_armstrong_number(num)` that checks if a number is an Armstrong number. It uses a few-shot prompt style, providing examples of numbers and their corresponding outputs. The function calculates the sum of each digit raised to the power of the number of digits and compares it to the original number. The terminal output shows the function being tested with the numbers 153, 370, and 123, resulting in '153 is an Armstrong number.', '370 is an Armstrong number.', and '123 is not an Armstrong number.' respectively. A 'Build with Agent' sidebar is visible on the right, and a 'git repository' notification is at the bottom right.

```
1 """
2 num = 153
3 print is an armstrong number or not
4 num = 370
5 print is an armstrong number or not
6 num = 123
7 print is an armstrong number or not
8 """
9
10 def is_armstrong_number(num):
11     # Convert the number to string to easily iterate over digits
12     digits = str(num)
13     num_digits = len(digits)
14
15     # Calculate the sum of each digit raised to the power of num_digits
16     sum_of_powers = sum(int(digit) ** num_digits for digit in digits)
17
18     # An Armstrong number is one where the sum of the powers equals the original number
19     return sum_of_powers == num
20
21 if __name__ == "__main__":
22     test_numbers = [153, 370, 123]
23     for number in test_numbers:
24         if is_armstrong_number(number):
25             print(f"{number} is an Armstrong number.")
26         else:
27             print(f"{number} is not an Armstrong number.")
```

PS C:\Users\vyshn> & C:\Users\vyshn\AppData\Local\Programs\Python\python313/python.exe "C:\Users\vyshn\OneDrive\Desktop\aiac\assignment 3\question 3.py"

153 is an Armstrong number.  
370 is an Armstrong number.  
123 is not an Armstrong number.

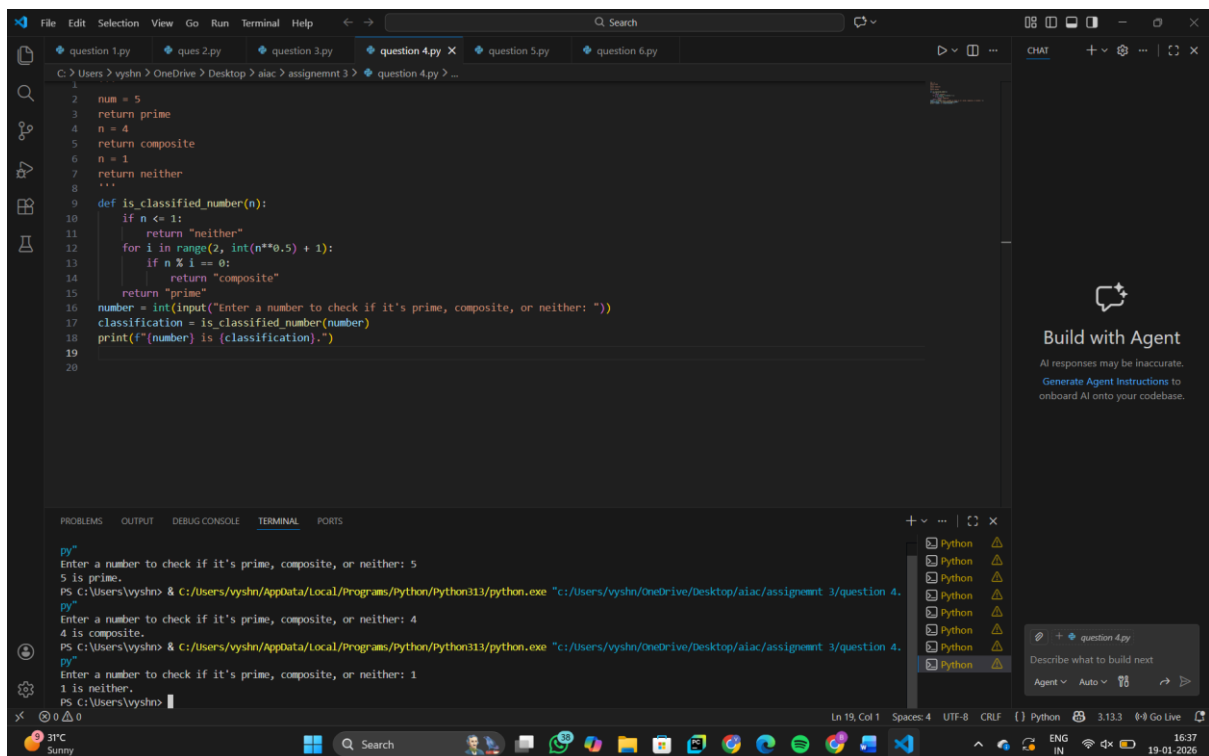
PS C:\Users\vyshn>

## 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.



The screenshot displays a Visual Studio Code editor window with a Python file named 'question 4.py'. The code defines a function `is_classified_number(n)` that checks if a number is prime, composite, or neither. It uses a loop to check divisibility up to  $\sqrt{n}$ . The main part of the script prompts the user for a number and prints the classification.

```
1 num = 5
2 return prime
3 n = 4
4 return composite
5 n = 1
6 return neither
7 ...
8
9 def is_classified_number(n):
10     if n <= 1:
11         return "neither"
12     for i in range(2, int(n**0.5) + 1):
13         if n % i == 0:
14             return "composite"
15     return "prime"
16 number = int(input("Enter a number to check if it's prime, composite, or neither: "))
17 classification = is_classified_number(number)
18 print(f"{number} is {classification}.")
19
20
```

The terminal output shows the program being executed with inputs 5, 4, and 1, resulting in the classifications 'prime', 'composite', and 'neither' respectively.

```
py
Enter a number to check if it's prime, composite, or neither: 5
5 is prime.
PS C:\Users\vyshn> & C:/Users/vyshn/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/vyshn/OneDrive/Desktop/alac/assignment 3/question 4.py"
py
Enter a number to check if it's prime, composite, or neither: 4
4 is composite.
PS C:\Users\vyshn> & C:/Users/vyshn/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/vyshn/OneDrive/Desktop/alac/assignment 3/question 4.py"
py
Enter a number to check if it's prime, composite, or neither: 1
1 is neither.
PS C:\Users\vyshn>
```

## 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.

The screenshot shows a Visual Studio Code editor with a Python file named 'question 5.py'. The code defines a function `is_perfect_number(num)` that calculates the sum of divisors of a number and checks if it equals the number itself. The main block prompts the user to enter a number and prints the result.

```
1 ...
2 write a python program to check whether a given number is perfect number or not.
3 ...
4 def is_perfect_number(num):
5     sum_of_divisors = 0
6     for i in range(1, num):
7         if num % i == 0:
8             sum_of_divisors += i
9     return sum_of_divisors == num
10
11 if __name__ == "__main__":
12     number = int(input("Enter a number: "))
13     if is_perfect_number(number):
14         print(f"{number} is a perfect number.")
15     else:
16         print(f"{number} is not a perfect number.")
```

The terminal window shows the program being executed with three test cases: 25 (not perfect), 1 (not perfect), and 7 (perfect).

```
py
Enter a number: 25
25 is not a perfect number.
PS C:\Users\vyshn> & c:\Users\vyshn\AppData\Local\Programs\Python\Python313\python.exe "c:\Users\vyshn\OneDrive\Desktop\aiac\assignment 3\question 5.py"
py
Enter a number: 1
1 is not a perfect number.
PS C:\Users\vyshn> & c:\Users\vyshn\AppData\Local\Programs\Python\Python313\python.exe "c:\Users\vyshn\OneDrive\Desktop\aiac\assignment 3\question 5.py"
py
Enter a number: 7
7 is not a perfect number.
PS C:\Users\vyshn>
```

The sidebar on the right displays 'Build with Agent' and a list of Python files generated by the AI agent, including 'question 5.py'.

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

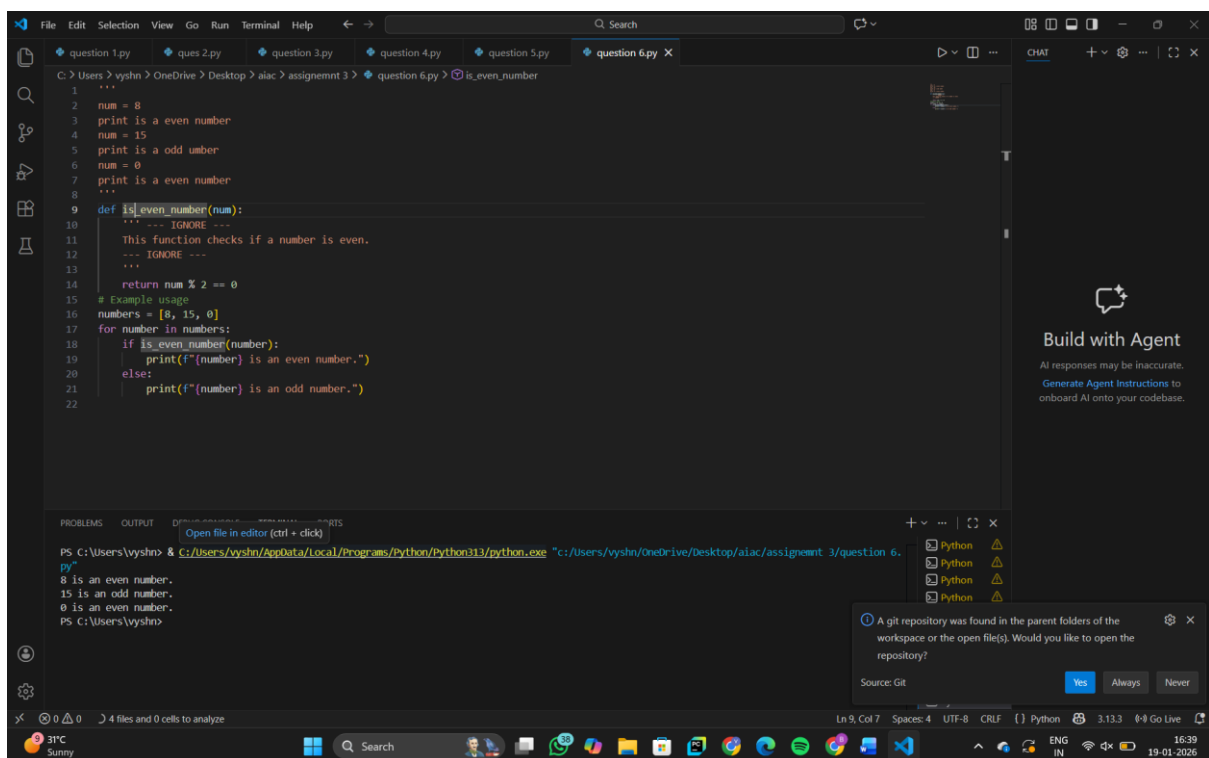
Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python program that determines whether a given number is even or odd, including proper input validation.

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.



```
File Edit Selection View Go Run Terminal Help
question 1.py question 2.py question 3.py question 4.py question 5.py question 6.py X
C:\Users\vyshn> OneDrive > Desktop > aiaac > assignment 3 > question 6.py > is_even_number
1 """
2 num = 8
3 print is a even number
4 num = 15
5 print is a odd umber
6 num = 0
7 print is a even number
8 """
9 def is_even_number(num):
10     """ --- IGNORE ---
11     This function checks if a number is even.
12     --- IGNORE ---
13     """
14     return num % 2 == 0
15 # Example usage
16 numbers = [8, 15, 0]
17 for number in numbers:
18     if is_even_number(number):
19         print(f"{number} is an even number.")
20     else:
21         print(f"{number} is an odd number.")
22
```

PROBLEMS OUTPUT

PS C:\Users\vyshn> & c:\Users\vyshn\AppData\Local\Programs\Python\python313/python.exe "c:\Users\vyshn\OneDrive\Desktop\aiac\assignment 3\question 6.py"

8 is an even number.  
15 is an odd number.  
0 is an even number.  
PS C:\Users\vyshn>

Build with Agent  
AI responses may be inaccurate.  
Generate Agent instructions to onboard AI onto your codebase.

A git repository was found in the parent folders of the workspace or the open file(s). Would you like to open the repository?  
Source: Git [Yes] [Always] [Never]

Ln 9, Col 7 Spaces: 4 UTF-8 CRLF Python 3.13.3 Go Live