Assignment-4.2

Name: Adepu Tejaswini

Ht.no: 2303A51423

Batch. No: 21

Task Description-1

• Zero-shot: Prompt AI with only the instruction. Write a Python function to determine

whether a given number is prime

Expected Output-1

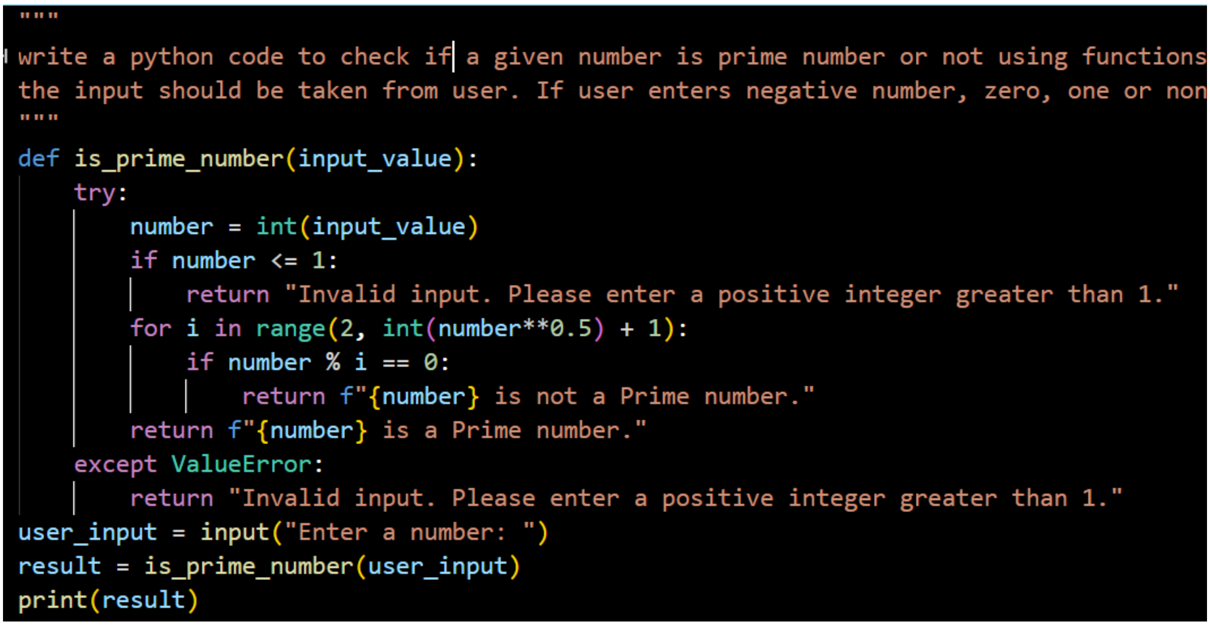
• A basic Python function to check if a number is prime, demonstrating correct logical

conditions without relying on examples or additional context

Prompt Used:

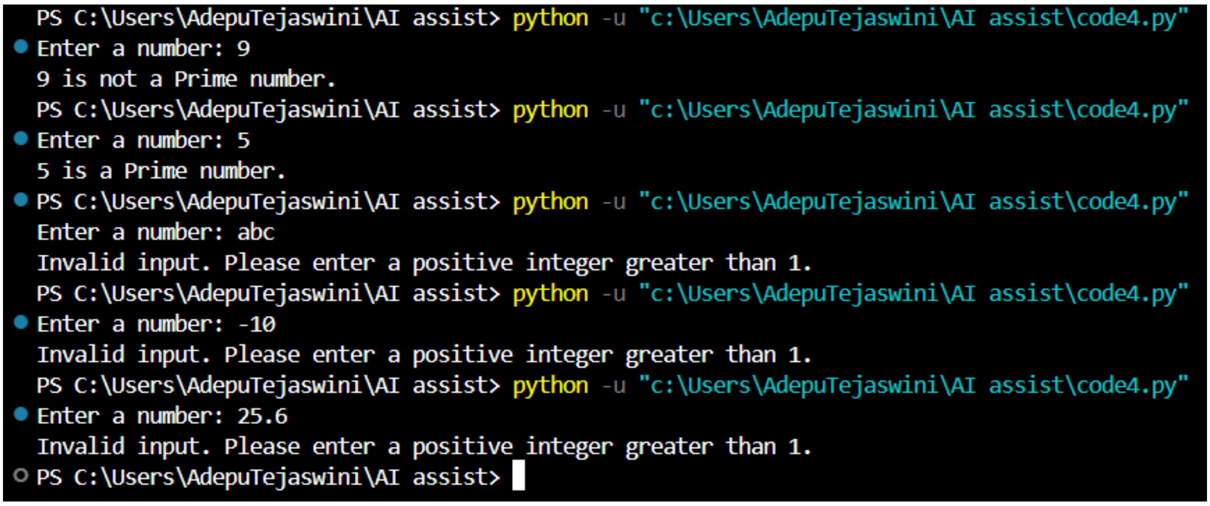
write a python code to check if a given number is prime number or not using functions , the input should be taken from user. If user enters negative number, zero, one or non-integer value, display a message "Invalid input. Please enter a positive integer greater than 1."

Code:



```
"""
write a python code to check if a given number is prime number or not using functions
the input should be taken from user. If user enters negative number, zero, one or non
"""
def is_prime_number(input_value):
    try:
        number = int(input_value)
        if number <= 1:
            return "Invalid input. Please enter a positive integer greater than 1."
        for i in range(2, int(number**0.5) + 1):
            if number % i == 0:
                return f"{number} is not a Prime number."
        return f"{number} is a Prime number."
    except ValueError:
        return "Invalid input. Please enter a positive integer greater than 1."
user_input = input("Enter a number: ")
result = is_prime_number(user_input)
print(result)
```

Output:



```
  PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
● Enter a number: 9
  9 is not a Prime number.
  PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
● Enter a number: 5
  5 is a Prime number.
● PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
  Enter a number: abc
  Invalid input. Please enter a positive integer greater than 1.
  PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
● Enter a number: -10
  Invalid input. Please enter a positive integer greater than 1.
  PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
● Enter a number: 25.6
  Invalid input. Please enter a positive integer greater than 1.
○ PS C:\Users\AdepuTejaswini\AI assist> █
```

Justification:

The code correctly uses a user-defined function to check whether a number is prime while taking input from the user and handling errors using a try-except block. It properly treats negative numbers, zero, one, and non-integer values as invalid and displays the required error message. The output confirms correct behavior by accurately identifying prime numbers, non-prime numbers, and invalid inputs as expected.

Task Description-2

- One-shot: Provide one example: Input: [1, 2, 3, 4], Output: 10 to help AI generate a

function that calculates the sum of elements in a list.

Expected Output-2

- A correct conversion function guided by the single example.

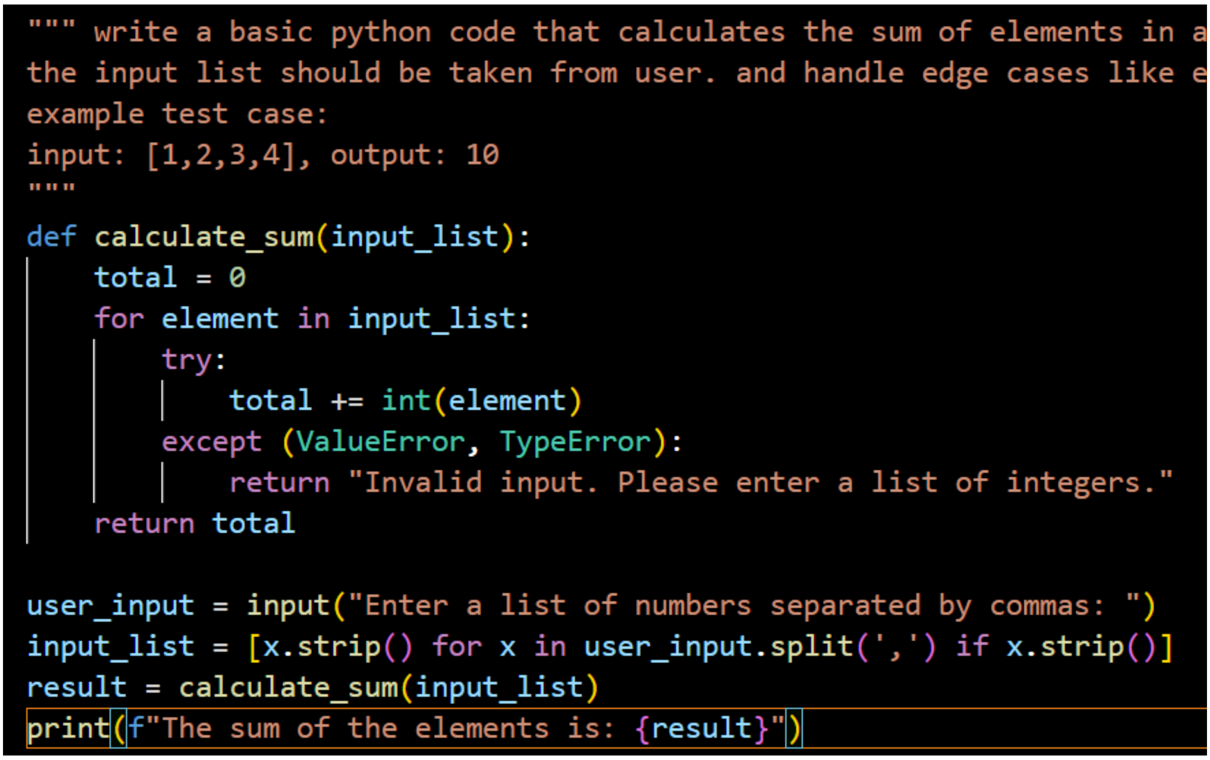
Prompt Used:

write a basic python code that calculates the sum of elements in a list using functions and loops. the input list should be taken from user. and handle edge cases like empty list and non-integer inputs.

example test case:

input: [1,2,3,4], output: 10

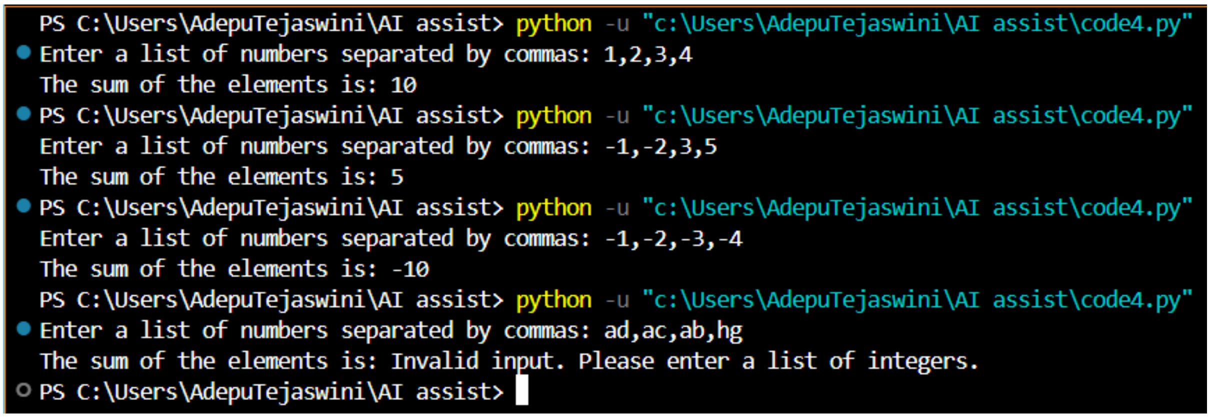
Code:



```python
""" write a basic python code that calculates the sum of elements in a
the input list should be taken from user. and handle edge cases like e
example test case:
input: [1,2,3,4], output: 10
"""
def calculate_sum(input_list):
    total = 0
    for element in input_list:
        try:
            total += int(element)
        except (ValueError, TypeError):
            return "Invalid input. Please enter a list of integers."
    return total

user_input = input("Enter a list of numbers separated by commas: ")
input_list = [x.strip() for x in user_input.split(',') if x.strip()]
result = calculate_sum(input_list)
print(f"The sum of the elements is: {result}")
```

Output:



```
PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
Enter a list of numbers separated by commas: 1,2,3,4
The sum of the elements is: 10
PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
Enter a list of numbers separated by commas: -1,-2,3,5
The sum of the elements is: 5
PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
Enter a list of numbers separated by commas: -1,-2,-3,-4
The sum of the elements is: -10
PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
Enter a list of numbers separated by commas: ad,ac,ab,hg
The sum of the elements is: Invalid input. Please enter a list of integers.
PS C:\Users\AdepuTejaswini\AI assist> 
```

Justification:

The code correctly uses a function and a loop to calculate the sum of elements in a user-provided list, following the one-shot example given. It handles edge cases such as non-integer inputs by using exception handling and displaying a clear error message. The outputs confirm correct summation for valid inputs and proper handling of invalid data.

Task Description-3

- Few-shot: Give 2–3 examples to create a function that extracts digits from an

alphanumeric string.

Expected Output-3

- Accurate function that returns only the digits from alphanumeric string.

Prompt Used:

write a python code that extracts and returns only digits from a give alphanumeric string using functions.

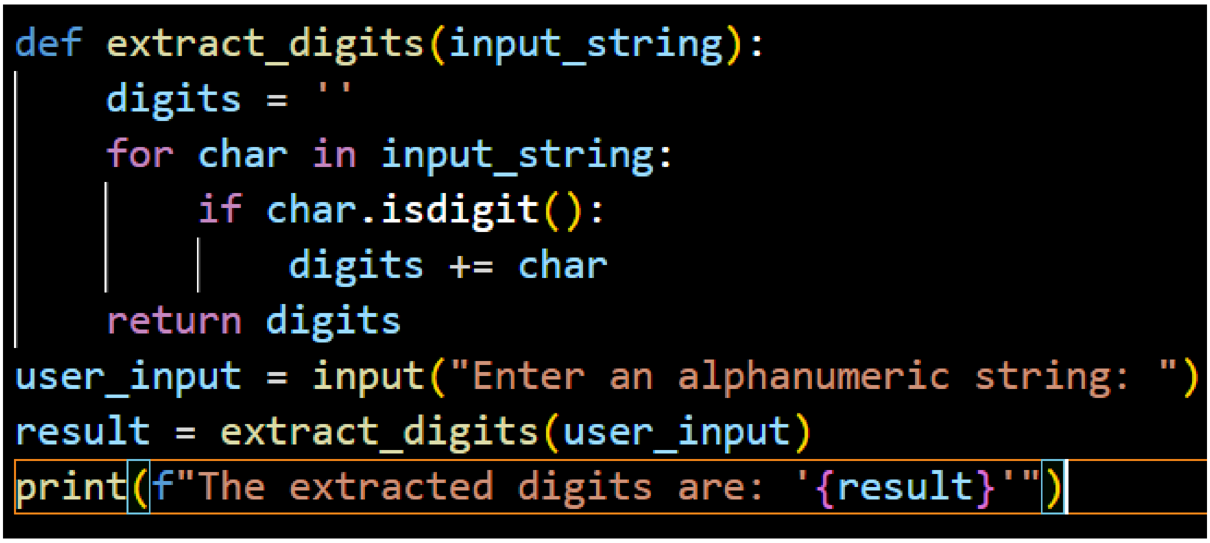
example test case:

input: "a1b2c3", output: "123"

input: "abc2024xyz" → Output: "2024"

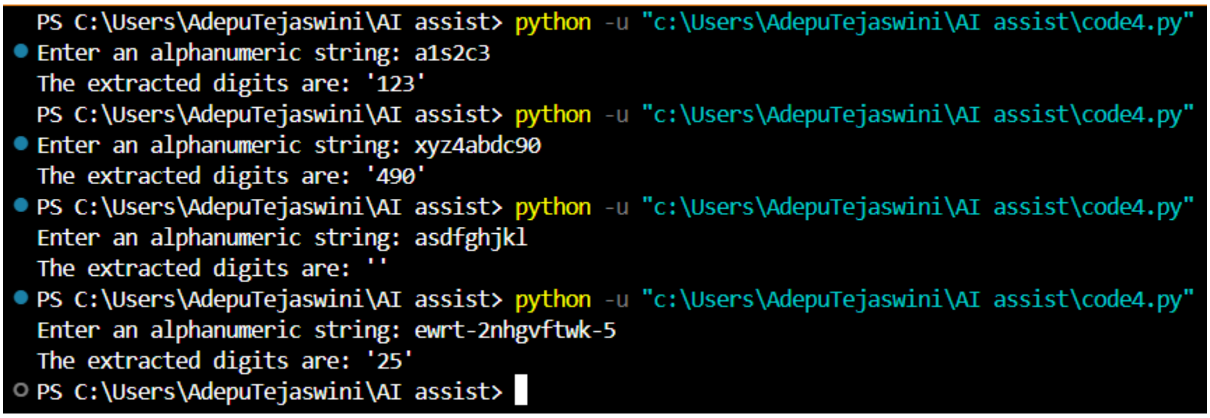
Input: "no_digits" → Output: ""

Code:



```
def extract_digits(input_string):
    digits = ''
    for char in input_string:
        if char.isdigit():
            digits += char
    return digits
user_input = input("Enter an alphanumeric string: ")
result = extract_digits(user_input)
print(f"The extracted digits are: '{result}'")
```

Output:



```
  PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
● Enter an alphanumeric string: a1s2c3
  The extracted digits are: '123'
  PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
● Enter an alphanumeric string: xyz4abdc90
  The extracted digits are: '490'
● PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
  Enter an alphanumeric string: asdfghjkl
  The extracted digits are: ''
● PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
  Enter an alphanumeric string: ewrt-2nhgvftwk-5
  The extracted digits are: '25'
○ PS C:\Users\AdepuTejaswini\AI assist> █
```

Justification:

The code correctly implements a user-defined function that iterates through each character of the input string and uses isdigit() to extract only numeric characters, which aligns with the few-shot examples provided. It preserves the original order of digits and safely ignores all non-digit characters without causing errors. The outputs confirm correct behavior by returning only digits when present and an empty string when no digits exist.

Task Description-4

- Compare zero-shot vs few-shot prompting for generating a function that counts the

number of vowels in a string.

Expected Output-4

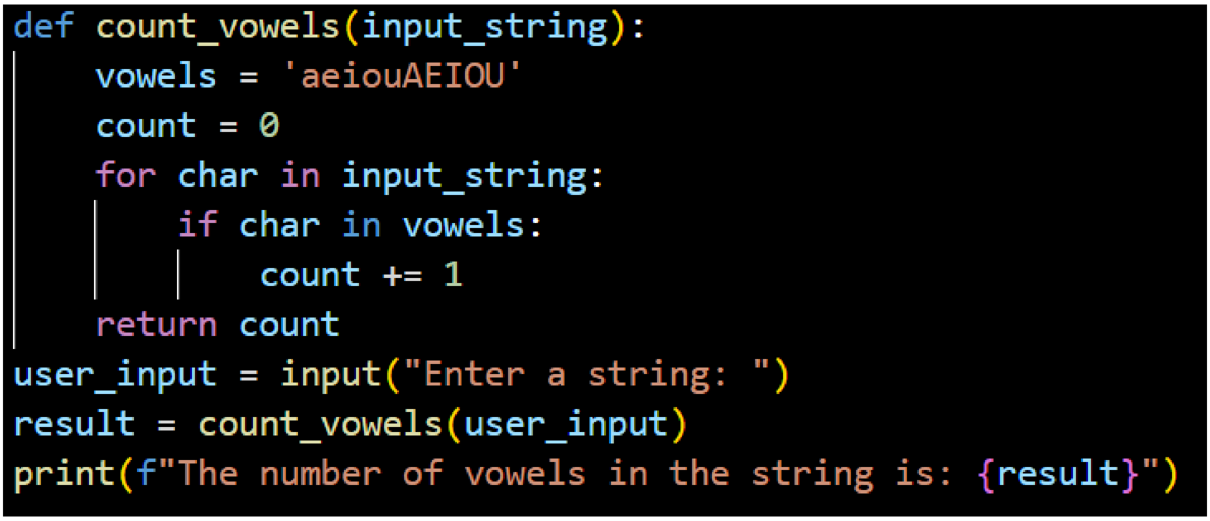
- Output comparison + student explanation on how examples helped the model.

Zero-shot:

Prompt Used:

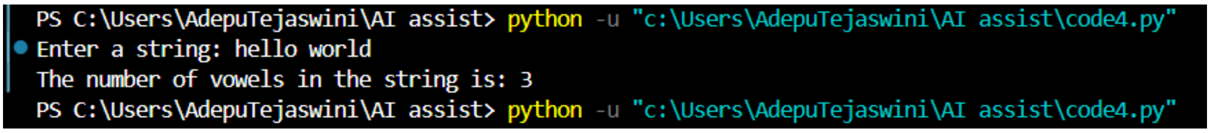
write a python function that count the number of vowels in a string.

Code:



```python
def count_vowels(input_string):
    vowels = 'aeiouAEIOU'
    count = 0
    for char in input_string:
        if char in vowels:
            count += 1
    return count
user_input = input("Enter a string: ")
result = count_vowels(user_input)
print(f"The number of vowels in the string is: {result}")
```

Output:



```
PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
Enter a string: hello world
The number of vowels in the string is: 3
PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
```

Few-shot:

Prompt Used:

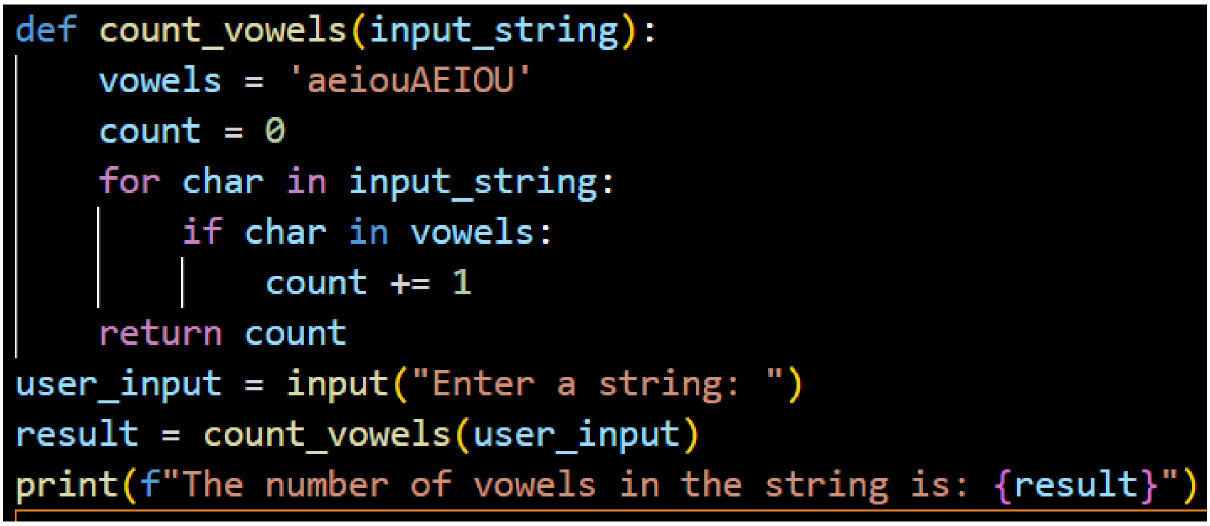
write a python function to count the number of vowels (including lower and upper letters) in a string.

Example test cases:

Input: "Hello World" → Output: 3

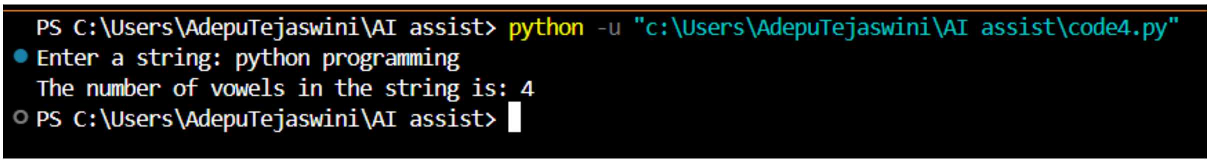
Input: "Python Programming" → Output: 4

Code:



```python
def count_vowels(input_string):
    vowels = 'aeiouAEIOU'
    count = 0
    for char in input_string:
        if char in vowels:
            count += 1
    return count
user_input = input("Enter a string: ")
result = count_vowels(user_input)
print(f"The number of vowels in the string is: {result}")
```

Output:



```
PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI assist\code4.py"
Enter a string: python programming
The number of vowels in the string is: 4
PS C:\Users\AdepuTejaswini\AI assist> 
```

Justification:

Zero-shot version:

The function is straightforward and works correctly by using basic logic such as looping through the string and counting vowels. However, without examples, the handling of uppercase and lowercase vowels may rely on assumptions made by the model.

Few-shot version:

The function is more accurate and polished because the examples clearly show that both uppercase and lowercase vowels must be considered. This guidance helps the model produce a more confident, optimized, and unambiguous implementation.

Task Description-5

- Use few-shot prompting with 3 sample inputs to generate a function that determines

the minimum of three numbers without using the built-in min() function.

Expected Output-5

- A function that handles all cases with correct logic based on example patterns.

Prompt Used:

Write a Python function that determines the minimum of three numbers without using the built-in min() function.

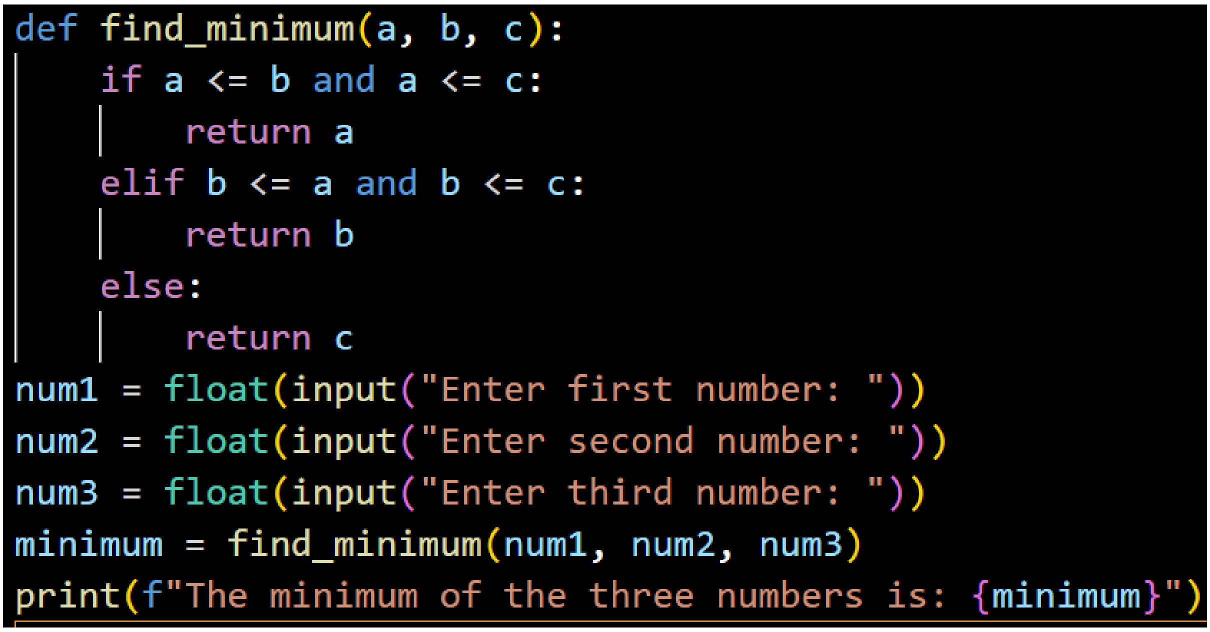
Examples:

Input: 3, 7, 5 → Output: 3

Input: 10, 2, 8 → Output: 2

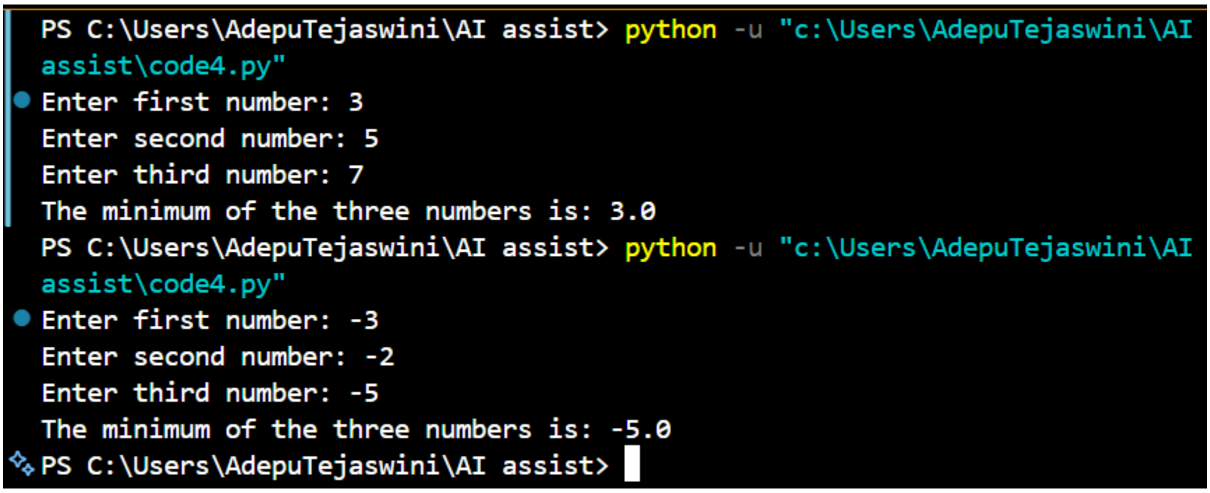
Input: 4, 4, 9 → Output: 4

Code:



```python
def find_minimum(a, b, c):
    if a <= b and a <= c:
        return a
    elif b <= a and b <= c:
        return b
    else:
        return c
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
num3 = float(input("Enter third number: "))
minimum = find_minimum(num1, num2, num3)
print(f"The minimum of the three numbers is: {minimum}")
```

Output:



```
PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI
assist\code4.py"
Enter first number: 3
Enter second number: 5
Enter third number: 7
The minimum of the three numbers is: 3.0
PS C:\Users\AdepuTejaswini\AI assist> python -u "c:\Users\AdepuTejaswini\AI
assist\code4.py"
Enter first number: -3
Enter second number: -2
Enter third number: -5
The minimum of the three numbers is: -5.0
PS C:\Users\AdepuTejaswini\AI assist>
```

Justification:

The few-shot prompt provides clear patterns through multiple examples, helping the model understand how to compare three values using logical conditions instead of built-in functions. These examples guide the function to correctly handle different scenarios, including equal values. As a result, the generated function applies consistent and correct logic across all input cases.