ASSIGNMENT-4.2

HT.NO:2303A510I4

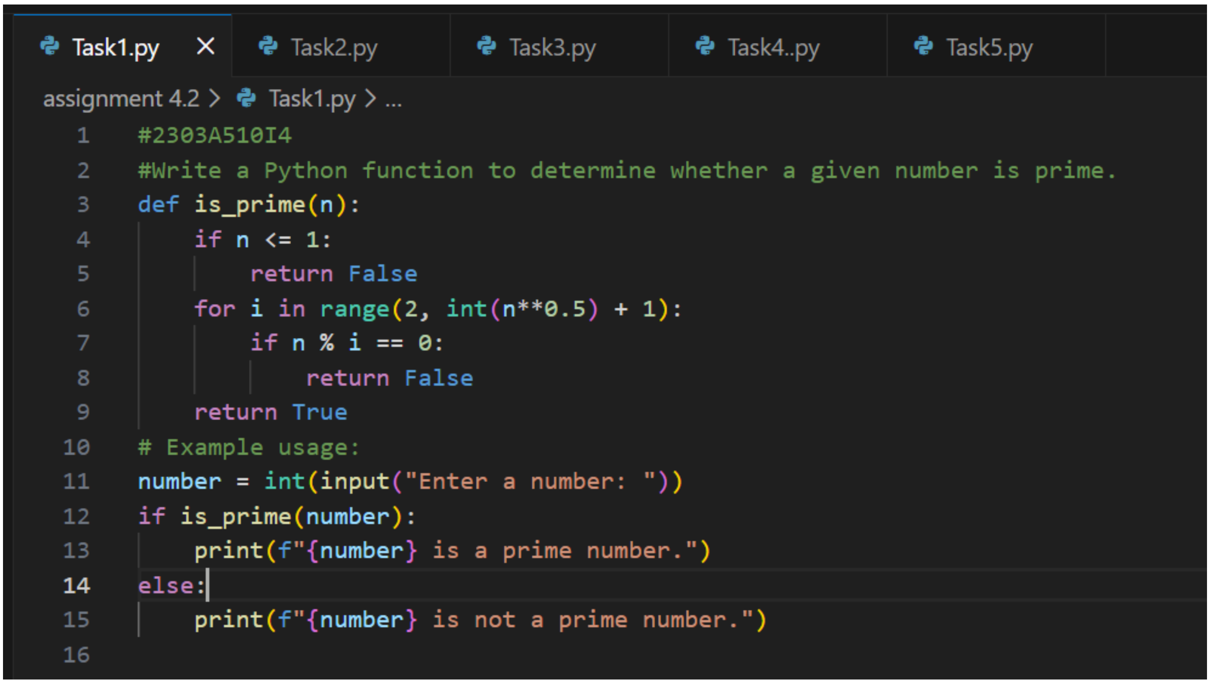
Batch.No:30

Task 1:

Prompt:

Write a Python function to determine whether a given number is prime.

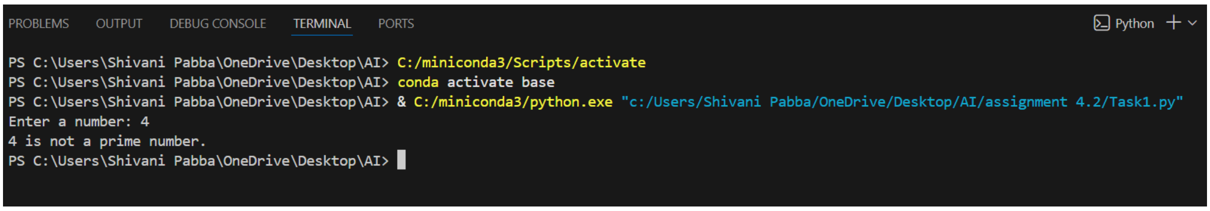
Code:



```python
#2303A510I4
#Write a Python function to determine whether a given number is prime.
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
# Example usage:
number = int(input("Enter a number: "))
if is_prime(number):
    print(f"{number} is a prime number.")
else:
    print(f"{number} is not a prime number.")

```

Output:



```
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> C:/miniconda3/Scripts/activate
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> conda activate base
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "c:/Users/Shivani Pabba/OneDrive/Desktop/AI/assignment 4.2/Task1.py"
Enter a number: 4
4 is not a prime number.
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> 
```

Observation:

Zero-shot prompting worked well because the instruction was clear. The model generated correct logic without needing examples. However, we had to explicitly call the function to see output.

Task 2:

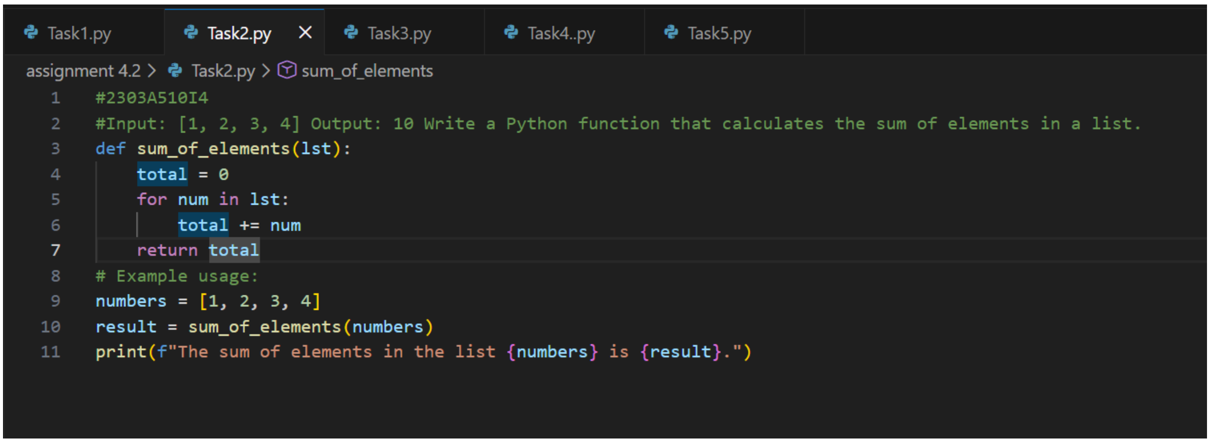
Prompt:

Input: [1, 2, 3, 4]

Output: 10

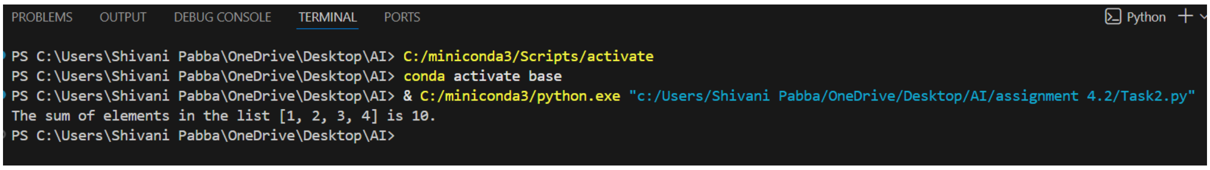
Write a Python function that calculates the sum of elements in a list.

Code:



```python
#2303A510I4
#Input: [1, 2, 3, 4] Output: 10 Write a Python function that calculates the sum of elements in a list.
def sum_of_elements(lst):
    total = 0
    for num in lst:
        total += num
    return total
# Example usage:
numbers = [1, 2, 3, 4]
result = sum_of_elements(numbers)
print(f"The sum of elements in the list {numbers} is {result}.")
```

Output:



```
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> C:/miniconda3/Scripts/activate
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> conda activate base
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "c:/Users/Shivani Pabba/OneDrive/Desktop/AI/assignment 4.2/Task2.py"
The sum of elements in the list [1, 2, 3, 4] is 10.
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI>
```

Observation:

Providing one example helped the model clearly understand that all elements in the list must be added. The single example was enough to guide the correct logic.

Task 3:

Prompt:

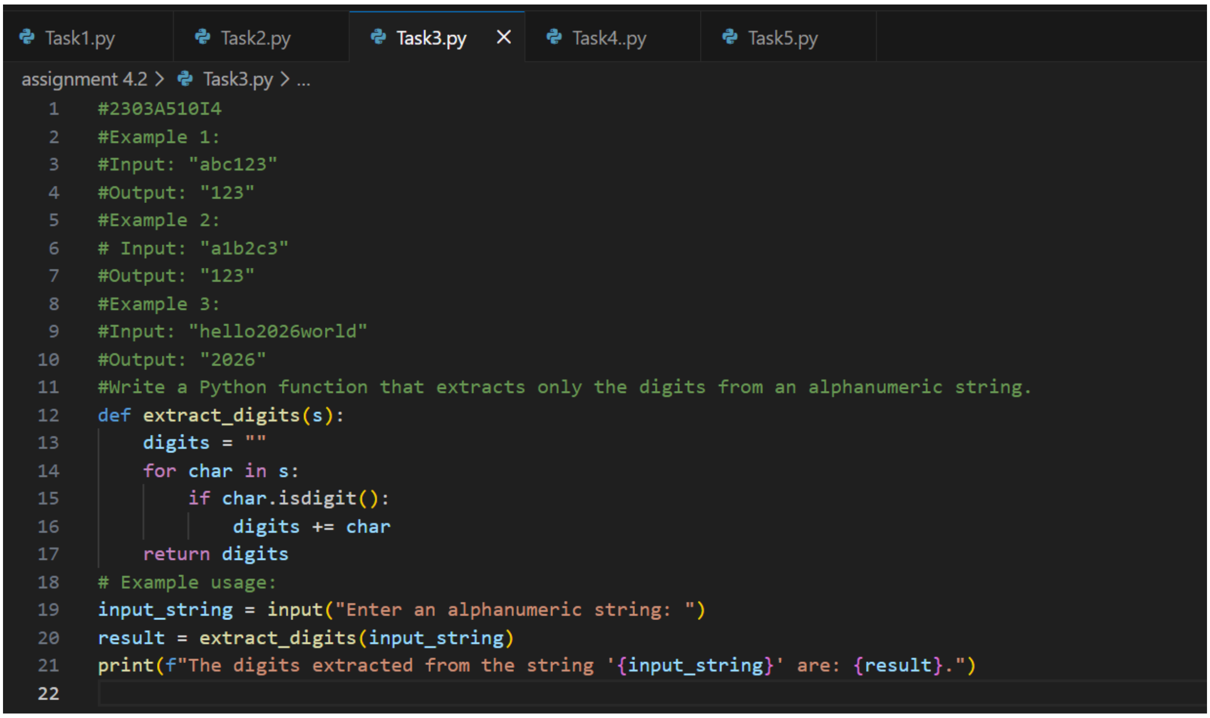
Example 1:
Input: "abc123"
Output: "123"

Example 2:
Input: "a1b2c3"
Output: "123"

Example 3:
Input: "hello2026world"
Output: "2026"

Write a Python function that extracts only the digits from an alphanumeric string.

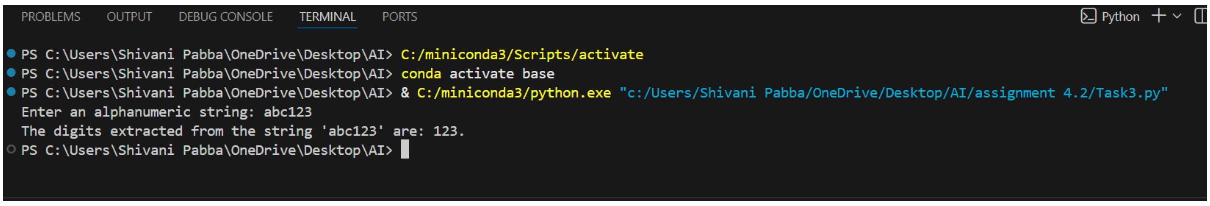
Code:



```python
#2303A510I4
#Example 1:
#Input: "abc123"
#Output: "123"
#Example 2:
# Input: "a1b2c3"
#Output: "123"
#Example 3:
#Input: "hello2026world"
#Output: "2026"
#Write a Python function that extracts only the digits from an alphanumeric string.
def extract_digits(s):
    digits = ""
    for char in s:
        if char.isdigit():
            digits += char
    return digits
# Example usage:
input_string = input("Enter an alphanumeric string: ")
result = extract_digits(input_string)
print(f"The digits extracted from the string '{input_string}' are: {result}.")

```

Output:



```
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> C:/miniconda3/Scripts/activate
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> conda activate base
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "c:/Users/Shivani Pabba/OneDrive/Desktop/AI/assignment 4.2/Task3.py"
Enter an alphanumeric string: abc123
The digits extracted from the string 'abc123' are: 123.
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI>
```

Observation:

Multiple examples made the pattern very clear. The model understood that only numeric characters should be kept and everything else ignored. Few-shot improved clarity and accuracy.

Task 4:

Prompt:

Zero-shot:

Write a Python function that counts the number of vowels in a string.

Few-shot with examples:

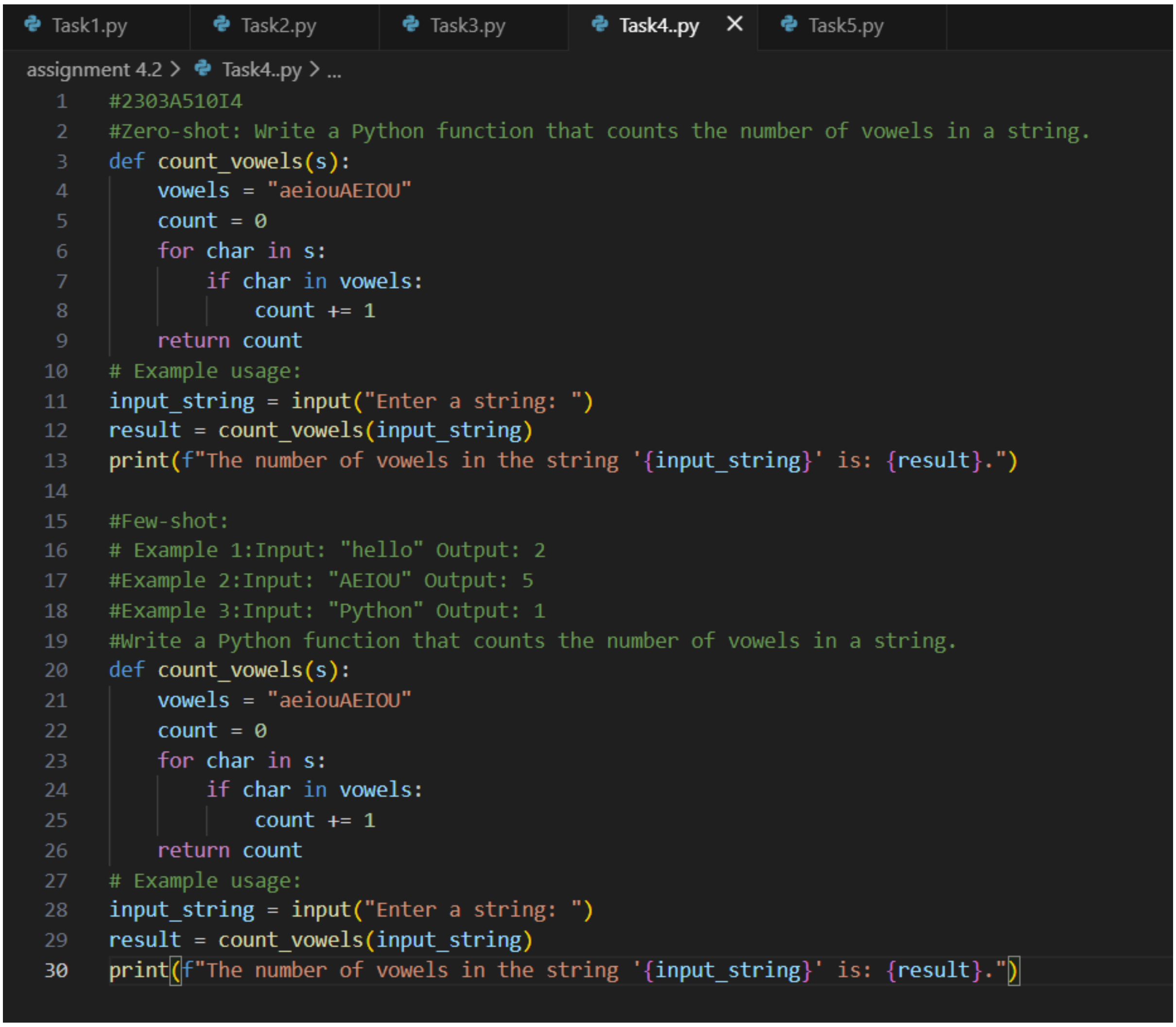
Example 1:
Input: "hello"
Output: 2

Example 2:
Input: "AEIOU"
Output: 5

Example 3:
Input: "Python"
Output: 1

Write a Python function that counts the number of vowels in a string.

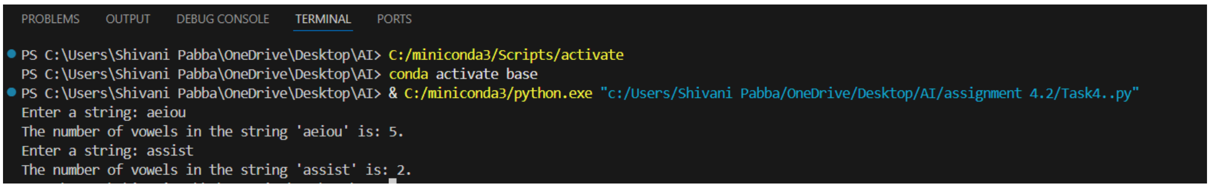
Code:



```python
#2303A510I4
#Zero-shot: Write a Python function that counts the number of vowels in a string.
def count_vowels(s):
    vowels = "aeiouAEIOU"
    count = 0
    for char in s:
        if char in vowels:
            count += 1
    return count
# Example usage:
input_string = input("Enter a string: ")
result = count_vowels(input_string)
print(f"The number of vowels in the string '{input_string}' is: {result}.")

#Few-shot:
# Example 1:Input: "hello" Output: 2
#Example 2:Input: "AEIOU" Output: 5
#Example 3:Input: "Python" Output: 1
#Write a Python function that counts the number of vowels in a string.
def count_vowels(s):
    vowels = "aeiouAEIOU"
    count = 0
    for char in s:
        if char in vowels:
            count += 1
    return count
# Example usage:
input_string = input("Enter a string: ")
result = count_vowels(input_string)
print(f"The number of vowels in the string '{input_string}' is: {result}.")
```

Output:



```
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> C:/miniconda3/Scripts/activate
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> conda activate base
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "c:/Users/Shivani Pabba/OneDrive/Desktop/AI/assignment 4.2/Task4..py"
Enter a string: aeiou
The number of vowels in the string 'aeiou' is: 5.
Enter a string: assist
The number of vowels in the string 'assist' is: 2.
```

Observation:

Zero-shot generated correct logic, but few-shot was more reliable. The examples helped ensure uppercase and lowercase vowels were handled properly. Few-shot reduced ambiguity.

Task 5:

Prompt:

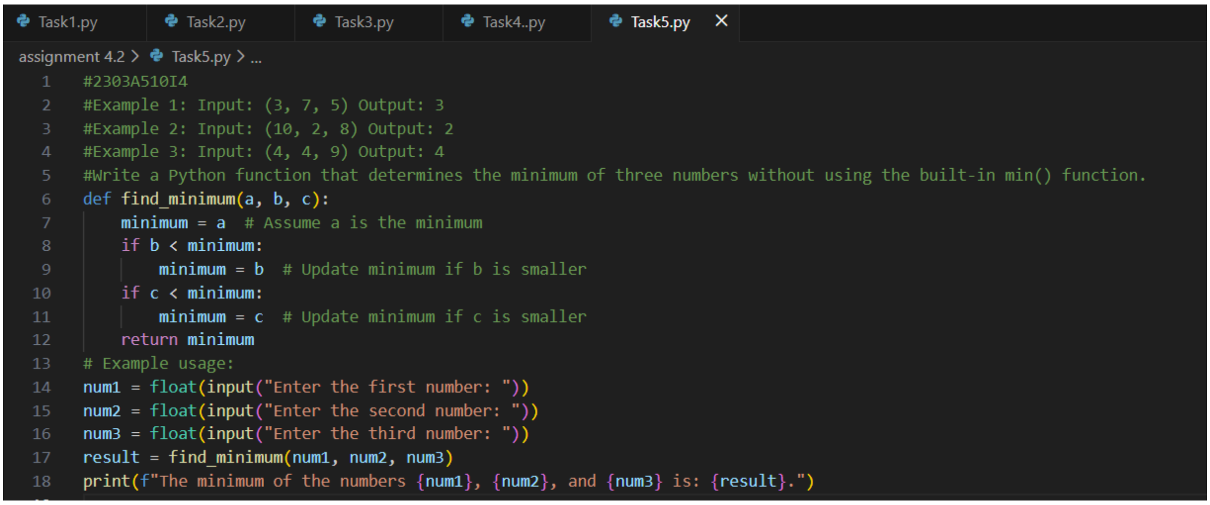
Example 1:
Input: (3, 7, 5)
Output: 3

Example 2:
Input: (10, 2, 8)
Output: 2

Example 3:
Input: (4, 4, 9)
Output: 4

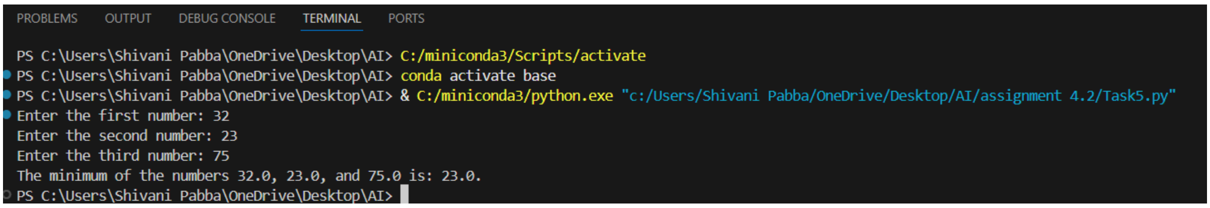
Write a Python function that determines the minimum of three numbers **without using the built-in min() function**.

Code:



```python
#2303A510I4
#Example 1: Input: (3, 7, 5) Output: 3
#Example 2: Input: (10, 2, 8) Output: 2
#Example 3: Input: (4, 4, 9) Output: 4
#Write a Python function that determines the minimum of three numbers without using the built-in min() function.
def find_minimum(a, b, c):
    minimum = a  # Assume a is the minimum
    if b < minimum:
        minimum = b  # Update minimum if b is smaller
    if c < minimum:
        minimum = c  # Update minimum if c is smaller
    return minimum
# Example usage:
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))
result = find_minimum(num1, num2, num3)
print(f"The minimum of the numbers {num1}, {num2}, and {num3} is: {result}.")
```

Output:



```
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> C:/miniconda3/Scripts/activate
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> conda activate base
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "c:/Users/Shivani Pabba/OneDrive/Desktop/AI/assignment 4.2/Task5.py"
Enter the first number: 32
Enter the second number: 23
Enter the third number: 75
The minimum of the numbers 32.0, 23.0, and 75.0 is: 23.0.
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI>
```

Observation:

The three examples helped the model understand comparison logic, including equal values. The function correctly compares numbers without using the built-in min() function.