AI Assisted Coding

Assignment – 4.2

Name: **L. Sharan Sai Varshith**

HtNo: **2303A51450**
Batch: **21**

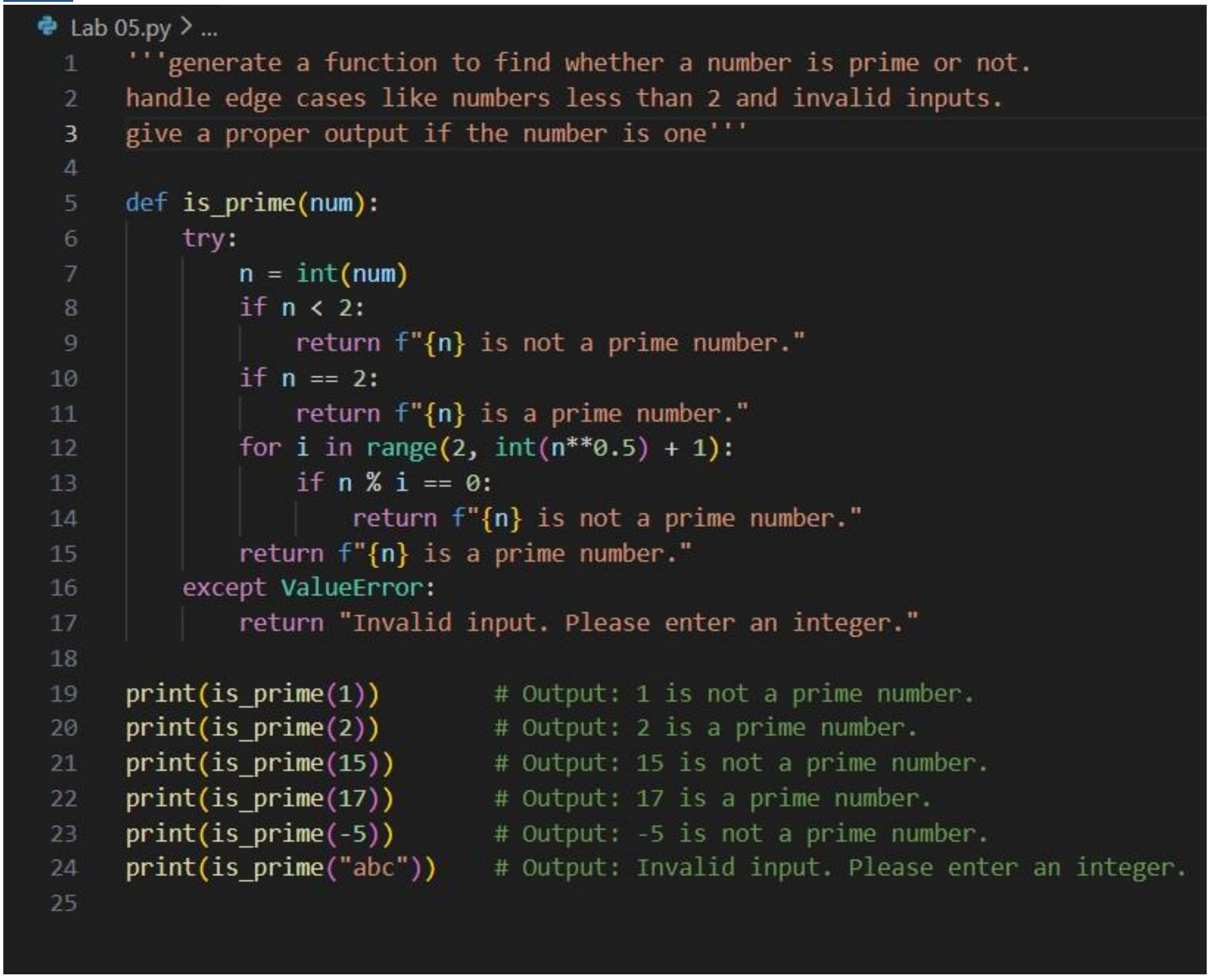
Task Description-1

• 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.

Code:



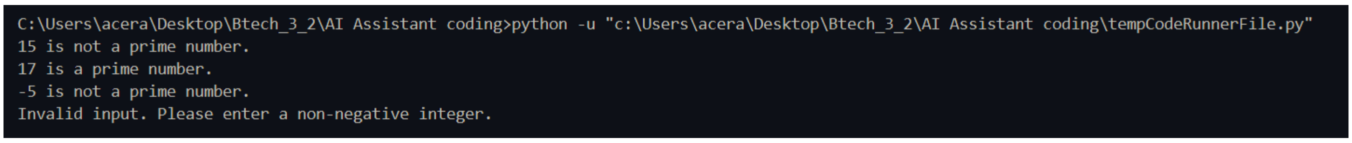
Lab 05.py > ...

```python
'''generate a function to find whether a number is prime or not.
handle edge cases like numbers less than 2 and invalid inputs.
give a proper output if the number is one'''

def is_prime(num):
    try:
        n = int(num)
        if n < 2:
            return f"{n} is not a prime number."
        if n == 2:
            return f"{n} is a prime number."
        for i in range(2, int(n**0.5) + 1):
            if n % i == 0:
                return f"{n} is not a prime number."
        return f"{n} is a prime number."
    except ValueError:
        return "Invalid input. Please enter an integer."

print(is_prime(1))          # Output: 1 is not a prime number.
print(is_prime(2))          # Output: 2 is a prime number.
print(is_prime(15))         # Output: 15 is not a prime number.
print(is_prime(17))         # Output: 17 is a prime number.
print(is_prime(-5))         # Output: -5 is not a prime number.
print(is_prime("abc"))      # Output: Invalid input. Please enter an integer.
```

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python -u "c:\Users\acera\Desktop\Btech_3_2\AI Assistant coding\tempCodeRunnerFile.py"
15 is not a prime number.
17 is a prime number.
-5 is not a prime number.
Invalid input. Please enter a non-negative integer.
```

Final Observation:

The is_prime(n) function checks whether a number is prime by first making sure the number is greater than 1, since 0, 1, and negative numbers are not prime. It then goes through the numbers from 2 to n−1 to see if any of them divide the given number exactly. If it finds even one such number, it returns False, meaning the number is not prime. If no divisor is found, it returns True, confirming that the number is prime. Because this solution was generated using zero-shot prompting, without giving any examples, it shows that the AI can understand the concept and produce a correct but basic prime-checking logic.

Task Description-2

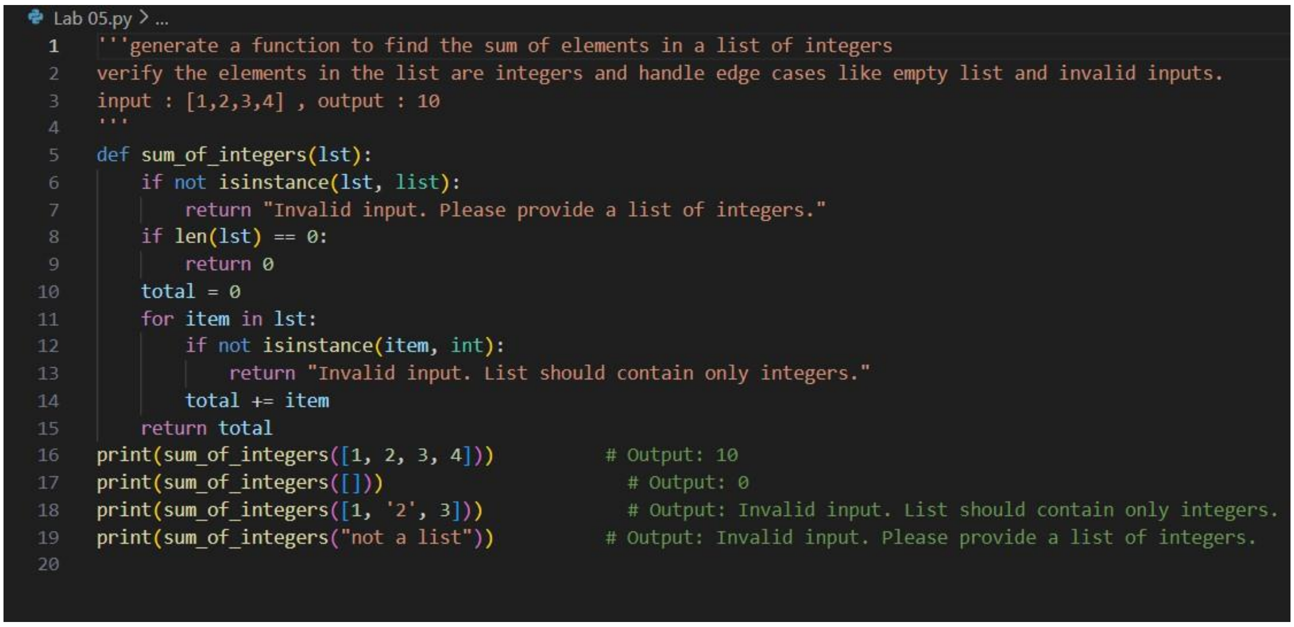
• One-shot: Provide one example: Input: [1, 2, 3, 4], Output: 10 to help AI generate a

function that calculates the sum of elements in a list.

Expected Output-2

• A correct conversion function guided by the single example.

Code:



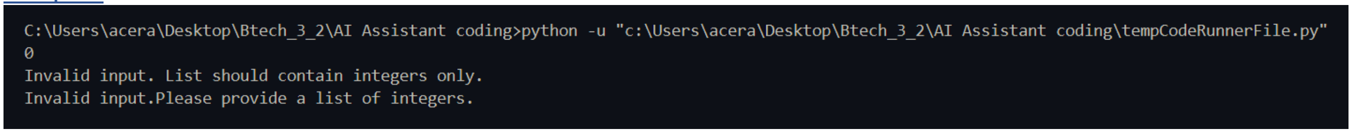
Lab 05.py > ...

```python
'''generate a function to find the sum of elements in a list of integers
verify the elements in the list are integers and handle edge cases like empty list and invalid inputs.
input : [1,2,3,4] , output : 10
'''
def sum_of_integers(lst):
    if not isinstance(lst, list):
        return "Invalid input. Please provide a list of integers."
    if len(lst) == 0:
        return 0
    total = 0
    for item in lst:
        if not isinstance(item, int):
            return "Invalid input. List should contain only integers."
        total += item
    return total
print(sum_of_integers([1, 2, 3, 4]))          # Output: 10
print(sum_of_integers([]))                      # Output: 0
print(sum_of_integers([1, '2', 3]))             # Output: Invalid input. List should contain only integers.
print(sum_of_integers("not a list"))          # Output: Invalid input. Please provide a list of integers.

```

•

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python -u "c:\Users\acera\Desktop\Btech_3_2\AI Assistant coding\tempCodeRunnerFile.py"
0
Invalid input. List should contain integers only.
Invalid input.Please provide a list of integers.
```

Final Observation:

The sum_of_elements(arr) function calculates the total sum of all elements in a list by starting with a variable total set to zero and then adding each number in the list one by one. After the loop finishes, the final value of total is returned as the sum of the list. Since this function was generated using one-shot prompting, where a single input-output example was provided, the example helped guide the AI to clearly understand the task and produce a correct and simple function for summing list elements.

Task Description-3

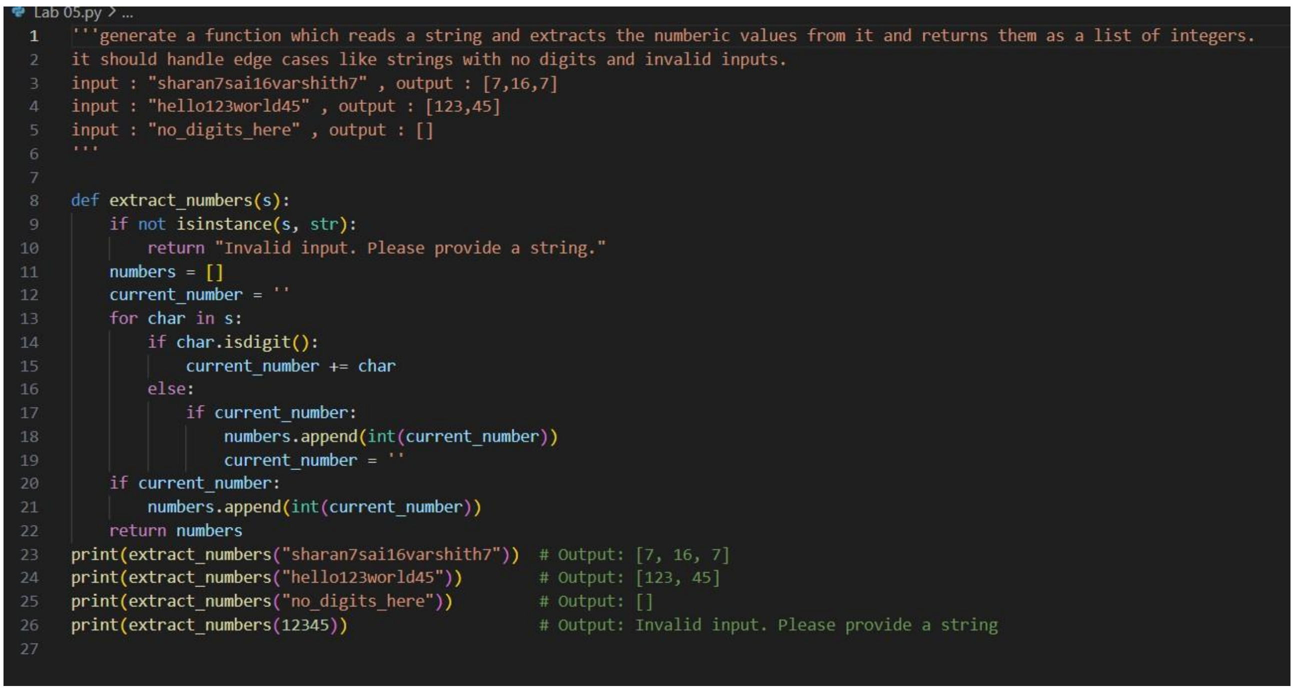
• Few-shot: Give 2–3 examples to create a function that extracts digits from an

alphanumeric string.

Expected Output-3

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

Code:



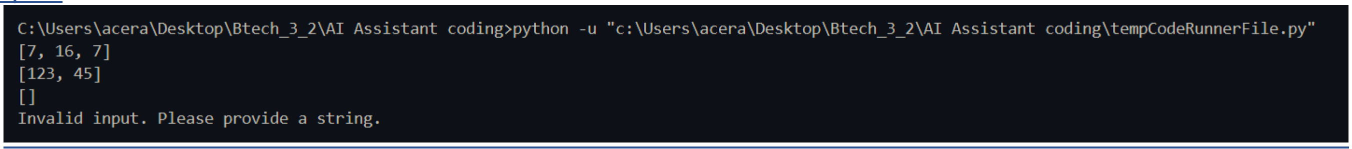
Lab 05.py > ...

```python
'''generate a function which reads a string and extracts the numberic values from it and returns them as a list of integers.
it should handle edge cases like strings with no digits and invalid inputs.
input : "sharan7sai16varshith7" , output : [7,16,7]
input : "hello123world45" , output : [123,45]
input : "no_digits_here" , output : []
'''

def extract_numbers(s):
    if not isinstance(s, str):
        return "Invalid input. Please provide a string."
    numbers = []
    current_number = ''
    for char in s:
        if char.isdigit():
            current_number += char
        else:
            if current_number:
                numbers.append(int(current_number))
                current_number = ''
    if current_number:
        numbers.append(int(current_number))
    return numbers
print(extract_numbers("sharan7sai16varshith7"))  # Output: [7, 16, 7]
print(extract_numbers("hello123world45"))        # Output: [123, 45]
print(extract_numbers("no_digits_here"))         # Output: []
print(extract_numbers(12345))                    # Output: Invalid input. Please provide a string
```

•

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python -u "c:\Users\acera\Desktop\Btech_3_2\AI Assistant coding\tempCodeRunnerFile.py"
[7, 16, 7]
[123, 45]
[]
Invalid input. Please provide a string.
```

Final Observation:

The extract_digits(text) function goes through each character in the given string and checks whether it is a digit using the isdigit() method. If the character is a digit, it is added to the result string. After the loop finishes, the function returns a string that contains only the digits from the original input. Since this was generated using few-shot prompting, the multiple examples clearly guided the AI to recognize the pattern of removing letters and keeping only numbers, resulting in an accurate and reliable function.

Task Description-4

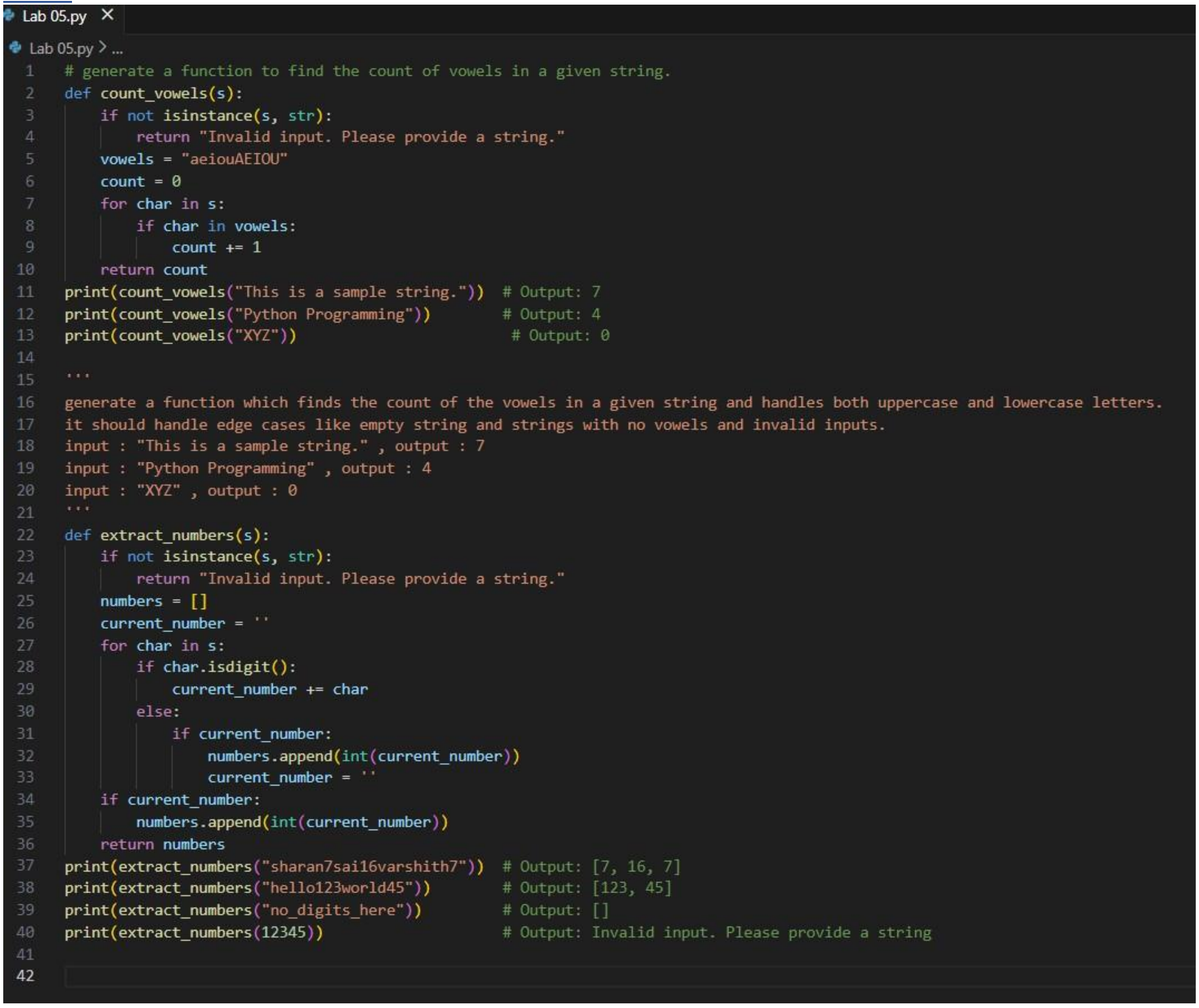
• Compare zero-shot vs few-shot prompting for generating a function that counts the number of vowels in a string.

Expected Output-4

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

Zero Shot:

Code:

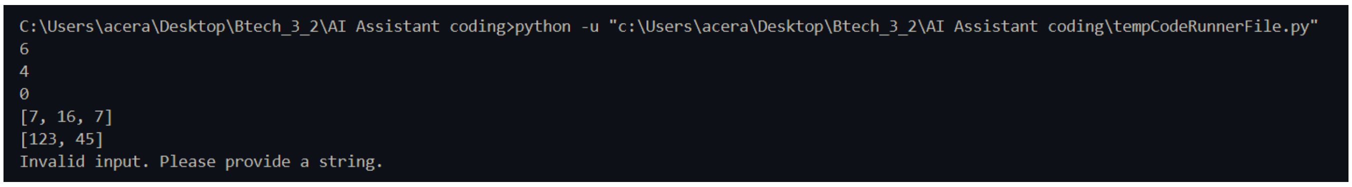


```python
# generate a function to find the count of vowels in a given string.
def count_vowels(s):
    if not isinstance(s, str):
        return "Invalid input. Please provide a string."
    vowels = "aeiouAEIOU"
    count = 0
    for char in s:
        if char in vowels:
            count += 1
    return count
print(count_vowels("This is a sample string."))  # Output: 7
print(count_vowels("Python Programming"))        # Output: 4
print(count_vowels("XYZ"))                        # Output: 0

'''
generate a function which finds the count of the vowels in a given string and handles both uppercase and lowercase letters.
it should handle edge cases like empty string and strings with no vowels and invalid inputs.
input : "This is a sample string." , output : 7
input : "Python Programming" , output : 4
input : "XYZ" , output : 0
'''
def extract_numbers(s):
    if not isinstance(s, str):
        return "Invalid input. Please provide a string."
    numbers = []
    current_number = ''
    for char in s:
        if char.isdigit():
            current_number += char
        else:
            if current_number:
                numbers.append(int(current_number))
                current_number = ''
    if current_number:
        numbers.append(int(current_number))
    return numbers
print(extract_numbers("sharan7sai16varshith7"))  # Output: [7, 16, 7]
print(extract_numbers("hello123world45"))        # Output: [123, 45]
print(extract_numbers("no_digits_here"))         # Output: []
print(extract_numbers(12345))                    # Output: Invalid input. Please provide a string

```

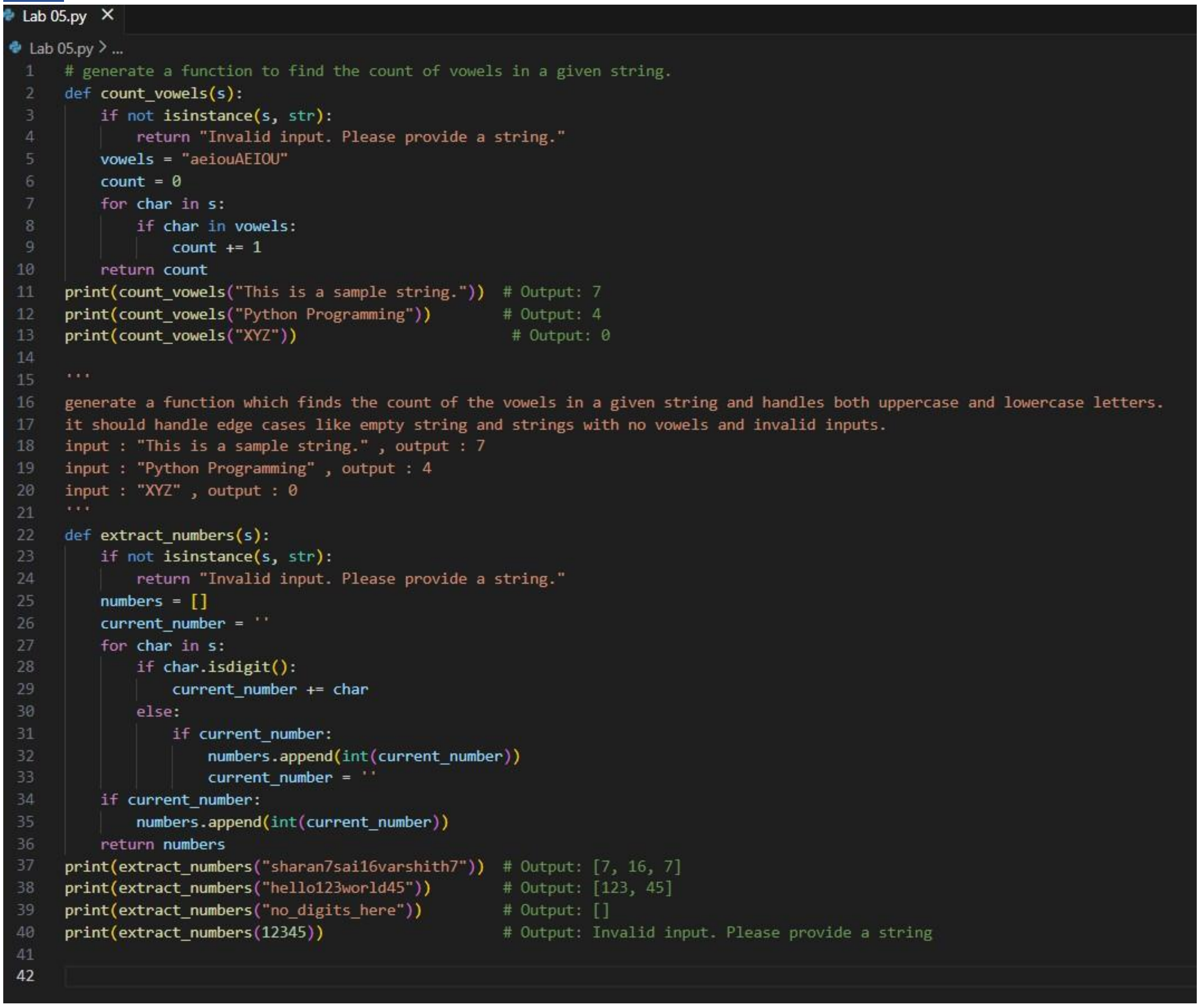
Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python -u "c:\Users\acera\Desktop\Btech_3_2\AI Assistant coding\tempCodeRunnerFile.py"
6
4
0
[7, 16, 7]
[123, 45]
Invalid input. Please provide a string.
```

Few Shot:

Code



Lab 05.py ×

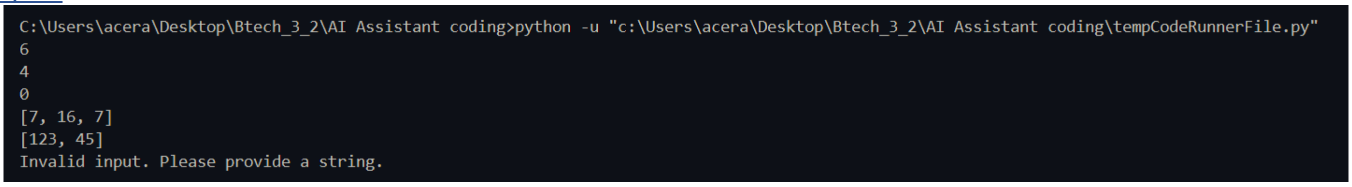
Lab 05.py > ...

```python
# generate a function to find the count of vowels in a given string.
def count_vowels(s):
    if not isinstance(s, str):
        return "Invalid input. Please provide a string."
    vowels = "aeiouAEIOU"
    count = 0
    for char in s:
        if char in vowels:
            count += 1
    return count
print(count_vowels("This is a sample string."))  # Output: 7
print(count_vowels("Python Programming"))        # Output: 4
print(count_vowels("XYZ"))                        # Output: 0

'''
generate a function which finds the count of the vowels in a given string and handles both uppercase and lowercase letters.
it should handle edge cases like empty string and strings with no vowels and invalid inputs.
input : "This is a sample string." , output : 7
input : "Python Programming" , output : 4
input : "XYZ" , output : 0
'''
def extract_numbers(s):
    if not isinstance(s, str):
        return "Invalid input. Please provide a string."
    numbers = []
    current_number = ''
    for char in s:
        if char.isdigit():
            current_number += char
        else:
            if current_number:
                numbers.append(int(current_number))
                current_number = ''
    if current_number:
        numbers.append(int(current_number))
    return numbers
print(extract_numbers("sharan7sai16varshith7"))  # Output: [7, 16, 7]
print(extract_numbers("hello123world45"))        # Output: [123, 45]
print(extract_numbers("no_digits_here"))         # Output: []
print(extract_numbers(12345))                    # Output: Invalid input. Please provide a string

```

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python -u "c:\Users\acera\Desktop\Btech_3_2\AI Assistant coding\tempCodeRunnerFile.py"
6
4
0
[7, 16, 7]
[123, 45]
Invalid input. Please provide a string.
```

Output Comparison:

Zero-shot version:
The function is clear, simple, and correct, using a loop and counter variable.

Few-shot version:
The function is more refined and compact. The examples helped the model clearly understand that both uppercase and lowercase vowels must be counted, leading to a more confident and optimized implementation.

Final Observation:

In the zero-shot approach, the AI generated a basic vowel-counting function based only on the instruction, which resulted in a straightforward loop-based solution. In the few-shot approach, the given examples clearly showed what should be considered as vowels and how the output should look. These examples helped the AI better understand the pattern and expectations, which led to a cleaner and slightly optimized solution. This comparison shows that providing examples improves clarity and often results in better-structured and more accurate outputs.

Task Description-5

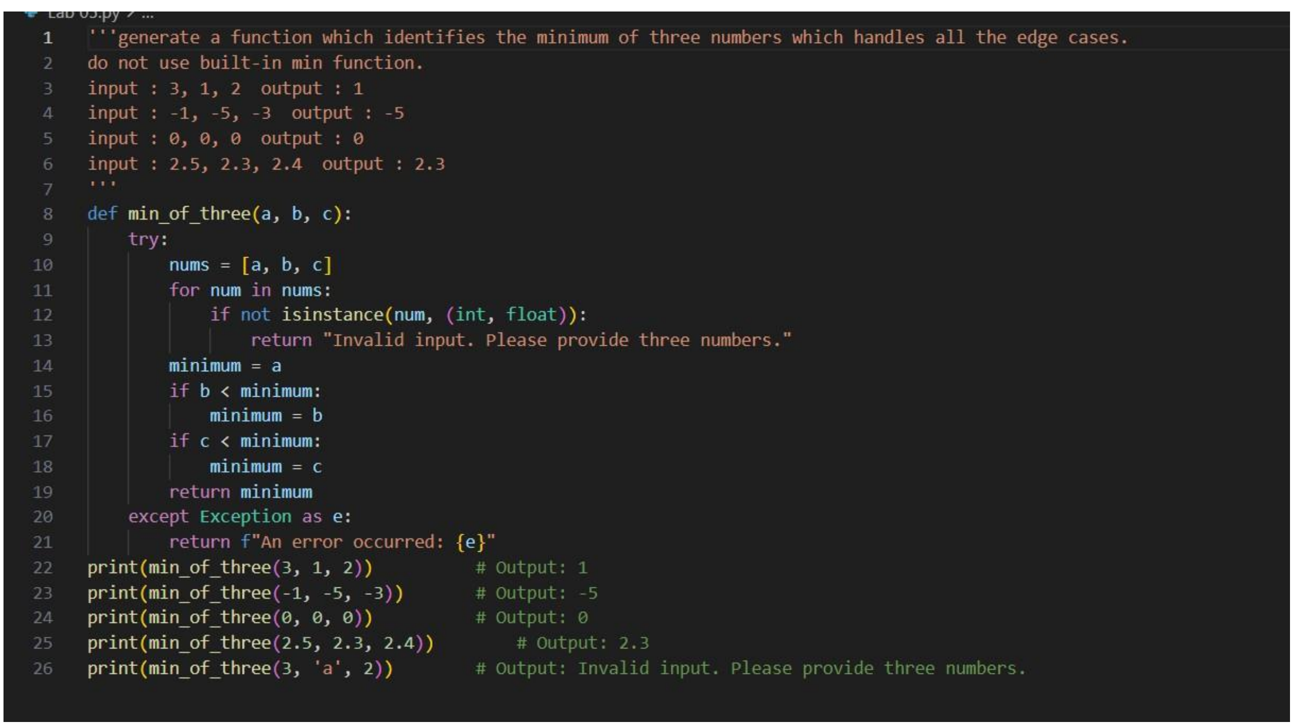
• Use few-shot prompting with 3 sample inputs to generate a function that determines

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.

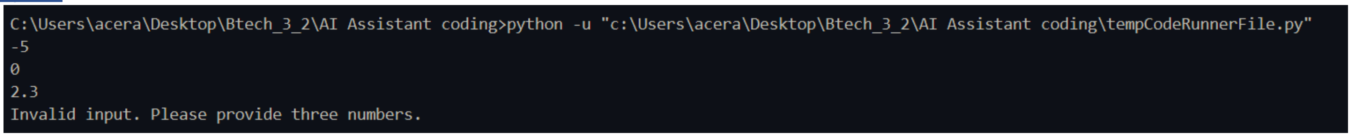
Code:



```
'''generate a function which identifies the minimum of three numbers which handles all the edge cases.
do not use built-in min function.
input : 3, 1, 2  output : 1
input : -1, -5, -3  output : -5
input : 0, 0, 0  output : 0
input : 2.5, 2.3, 2.4  output : 2.3
'''
def min_of_three(a, b, c):
    try:
        nums = [a, b, c]
        for num in nums:
            if not isinstance(num, (int, float)):
                return "Invalid input. Please provide three numbers."
        minimum = a
        if b < minimum:
            minimum = b
        if c < minimum:
            minimum = c
        return minimum
    except Exception as e:
        return f"An error occurred: {e}"
print(min_of_three(3, 1, 2))          # Output: 1
print(min_of_three(-1, -5, -3))       # Output: -5
print(min_of_three(0, 0, 0))          # Output: 0
print(min_of_three(2.5, 2.3, 2.4))        # Output: 2.3
print(min_of_three(3, 'a', 2))        # Output: Invalid input. Please provide three numbers.
```

•

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python -u "c:\Users\acera\Desktop\Btech_3_2\AI Assistant coding\tempCodeRunnerFile.py"
-5
0
2.3
Invalid input. Please provide three numbers.
```

Final Observation:

The find_minimum(a, b, c) function compares the three given numbers using conditional statements. It first checks whether a is smaller than or equal to both b and c. If not, it then checks whether b is smaller than or equal to the other two. If neither of these conditions is true, the function returns c as the minimum. Since this function was generated using few-shot prompting, the three examples clearly showed the expected pattern of comparing values and returning the smallest one. These examples helped the AI design logic that correctly handles all cases, including negative numbers.