

ASSIGNMENT-4.2

NAME- HABEEBA KHANAM

ROLL NO:2303A51474

BATCH NO:29

TASK-1: ZERO-SHOT PROMPTING

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

CODE:

The screenshot shows a Python code editor with a file named `3.1.py`. The code defines a function `is_prime(n)` to check if a given integer `n` is prime. It handles edge cases for `n < 2`, `n == 2`, and `n % 2 == 0`. For odd numbers greater than 2, it checks divisibility from 3 up to the square root of `n`. An example usage at the bottom demonstrates the function with values 17, 20, and 2.

```
File Edit Selection View Go Run ... ← → Search
3.1.py X
C:\Users\Sameera Khan> OneDrive > Desktop > 3.1.py > ...
1 def is_prime(n):
2     """
3         Determine whether a given number is prime.
4
5     Args:
6         n: An integer to check
7
8     Returns:
9         True if n is prime, False otherwise
10    """
11    if n < 2:
12        return False
13    if n == 2:
14        return True
15    if n % 2 == 0:
16        return False
17
18    for i in range(3, int(n**0.5) + 1, 2):
19        if n % i == 0:
20            return False
21    return True
22
23
24 # Example usage
25 if __name__ == "__main__":
26     print(is_prime(17)) # True
27     print(is_prime(20)) # False
28     print(is_prime(2)) # True

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Python + ... x
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
True
False
True
True
PS C:\Users\Sameera Khan>

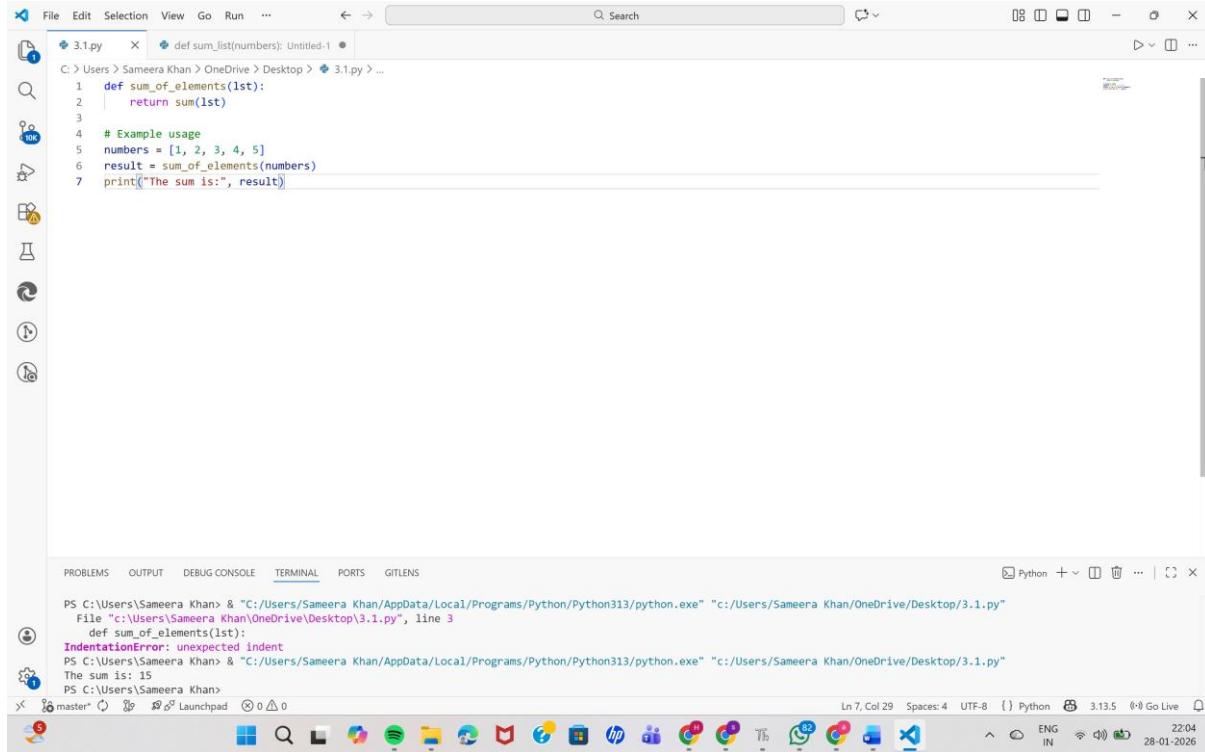
```

OBSERVATION:

The AI model successfully interprets the definition of a prime number solely from the prompt, without relying on examples or extra instructions. It applies correct mathematical logic to distinguish prime numbers from non-prime numbers. Additionally, the model produces Python code that is both syntactically correct and logically accurate, demonstrating effective zero-shot reasoning.

TASK-2

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



```
File Edit Selection View Go Run ... ⏪ ⏩ Search
3.1.py X def sum_list(numbers): Untitled-1 ...
C > Users > Sameera Khan > OneDrive > Desktop > 3.1.py > ...
1 def sum_of_elements(lst):
2     return sum(lst)
3
4 # Example usage
5 numbers = [1, 2, 3, 4, 5]
6 result = sum_of_elements(numbers)
7 print("The sum is:", result)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
File "c:/Users/Sameera Khan/OneDrive/Desktop/3.1.py", line 3
    def sum_of_elements(lst):
          ^
IndentationError: unexpected indent
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
The sum is: 15
PS C:\Users\Sameera Khan>
y\master* ⌂ ⌂ Launchpad ⌂ 0 △ 0
Ln 7, Col 29 Spaces: 4 UTF-8 { } Python 3.13.5 ⓘ Go Live 22:04
ENG IN 28-01-2026
```

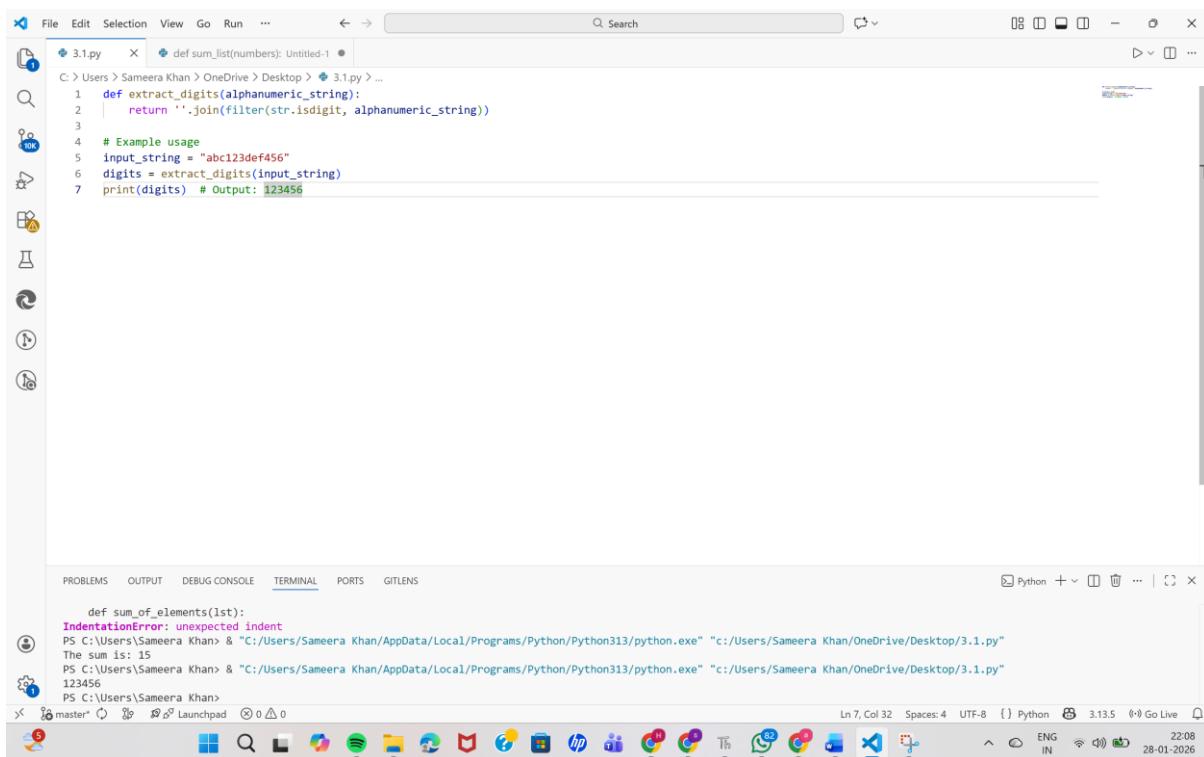
OBSERVATION:

By observing one example, the AI understands how inputs relate to outputs and correctly applies the same pattern to other lists of numbers.

TASK-3

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

CODE:



The screenshot shows a code editor interface with two tabs open: '3.1.py' and 'Untitled-1'. The '3.1.py' tab contains the following Python code:

```
File Edit Selection View Go Run ... Search
3.1.py X def sum_list(numbers): Untitled-1 ...
C: > Users > Sameera Khan > OneDrive > Desktop > 3.1.py > ...
1 def extract_digits(alphanumeric_string):
2     return ''.join(filter(str.isdigit, alphanumeric_string))
3
4 # Example usage
5 input_string = "abc123def456"
6 digits = extract_digits(input_string)
7 print(digits) # Output: 123456
```

The 'Untitled-1' tab shows a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Python + ... | x
def sum_of_elements(lst):
IndentationError: unexpected indent
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
The sum is: 15
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
123456
PS C:\Users\Sameera Khan>
```

The status bar at the bottom indicates the file is 'master' and the date is '28-01-2026'.

OBSERVATION:

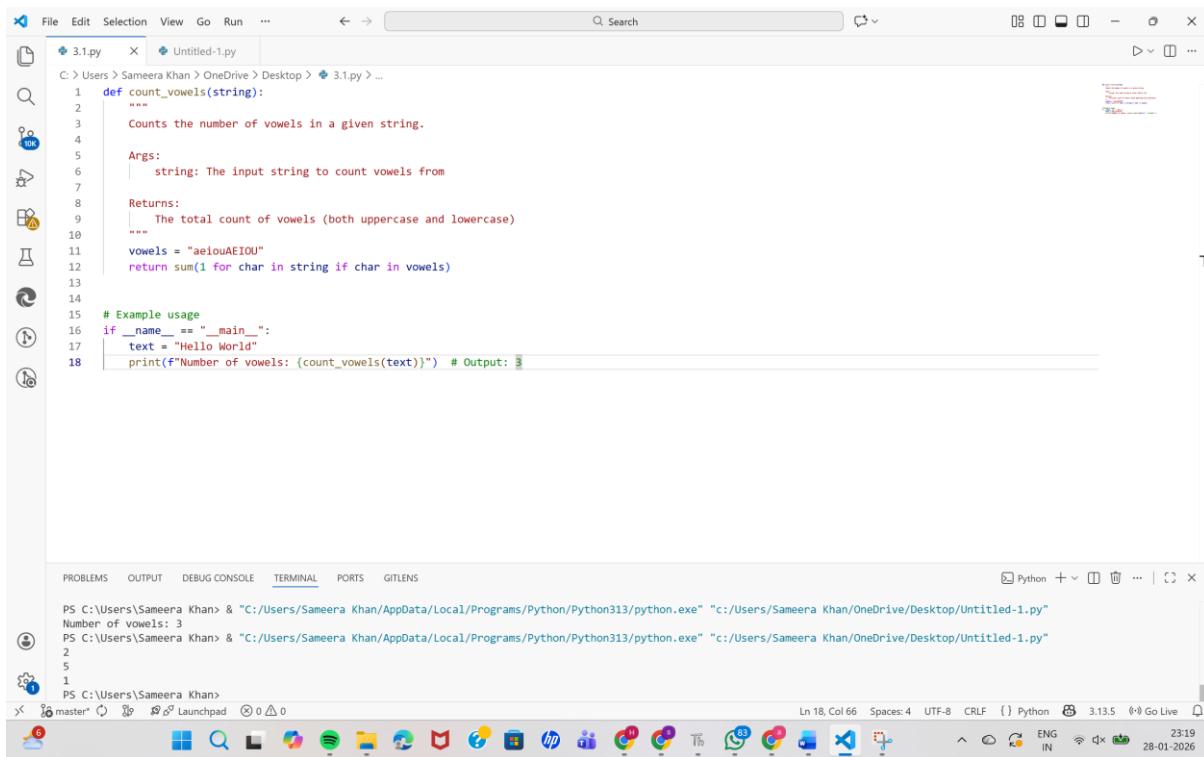
Providing multiple examples helps the AI model precisely identify the required pattern. The model correctly filters out alphabetic characters and focuses

solely on digits. This approach results in more reliable and unambiguous outputs than zero-shot and one-shot prompting.

TASK-4

PROMPT: ZERO-SHOT: Write a Python function that counts the number of vowels in a string.

FEW-SHOT: Write a Python function that counts the number of vowels in a string. Examples: Input: "hello" Output: 2 Input: "AEIOU" Output: 5 Input: "chatgpt" Output: 2



The screenshot shows the Visual Studio Code interface. The top part displays the code for a file named '3.1.py' which contains a function to count vowels in a string. The bottom part shows the terminal window with the output of running the script.

```
File Edit Selection View Go Run ... ← → Search ... File Explorer Problems Output Debug Console TERMINAL PORTS GITLENS Python + ...
```

```
C:\> Users > Sameera Khan > OneDrive > Desktop > 3.1.py > ...
1 def count_vowels(string):
2     """
3         Counts the number of vowels in a given string.
4
5     Args:
6         string: The input string to count vowels from
7
8     Returns:
9         The total count of vowels (both uppercase and lowercase)
10    """
11    vowels = "aeiouAEIOU"
12    return sum(1 for char in string if char in vowels)
13
14
15 # Example usage
16 if __name__ == "__main__":
17     text = "Hello World"
18     print(f"Number of vowels: {count_vowels(text)}") # Output: 3
```

```
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/Untitled-1.py"
Number of vowels: 3
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/Untitled-1.py"
2
5
1
PS C:\Users\Sameera Khan>
```

ZERO SHOT:

```
C:\> Users > Sameera Khan > OneDrive > Desktop > 3.1.py > ...
1 def count_vowels_zero_shot(s):
2     vowels = "aeiouAEIOU"
3     return sum(1 for char in s if char in vowels)
4
5 # Example usage
6 print(count_vowels_zero_shot("hello")) # Output: 2
7 print(count_vowels_zero_shot("AEIOU")) # Output: 5
8 print(count_vowels_zero_shot("chatgpt")) # Output: 2
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
5
1
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/Untitled-1.py"
2
5
1
```

PS C:\Users\Sameera Khan>

LN 8, COL 54 SPACES 4 UTF-8 CRLF { } PYTHON 3.13.5 ⓘ Go Live 23:21
ENG IN 28-01-2026 23:21

OBSERVATION:

FEW-SHOT OBSERVATION

The provided examples clearly define what characters should be counted as vowels

The model confidently includes both uppercase and lowercase vowels due to examples ZERO SHOT:

zero shot prompting the AI guesses the intent based on general knowledge which may vary for ambiguous tasks

TASK-5

PROMPT:

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

The screenshot shows the Visual Studio Code interface. In the top editor pane, there are two tabs: '3.1.py' and 'Untitled-1.py'. The '3.1.py' tab contains the following Python code:

```
C:\> Users > Sameera Khan > OneDrive > Desktop > 3.1.py > ...
1 def minimum_of_three(a, b, c):
2     if a < b and a < c:
3         return a
4     elif b < a and b < c:
5         return b
6     else:
7         return c
8
9 # Example usage
10 num1 = 5
11 num2 = 3
12 num3 = 8
13 print("The minimum of the three numbers is:", minimum_of_three(num1, num2, num3))
```

In the bottom terminal pane, the command `python 3.1.py` is run, and the output is:

```
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/Untitled-1.py"
2
5
1
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/3.1.py"
The minimum of the three numbers is: 3
PS C:\Users\Sameera Khan>
```

OBSERVATION:

The examples help the AI understand how to compare values to find the minimum. The model correctly accounts for equal values and implements the logic using conditional statements rather than built-in functions.

