Name: Rithu goud

Hall ticket: 2303A51641

Batch: 29

Assessment: 4.2

Task 1: Zero-shot: Prompt AI with only the instruction.

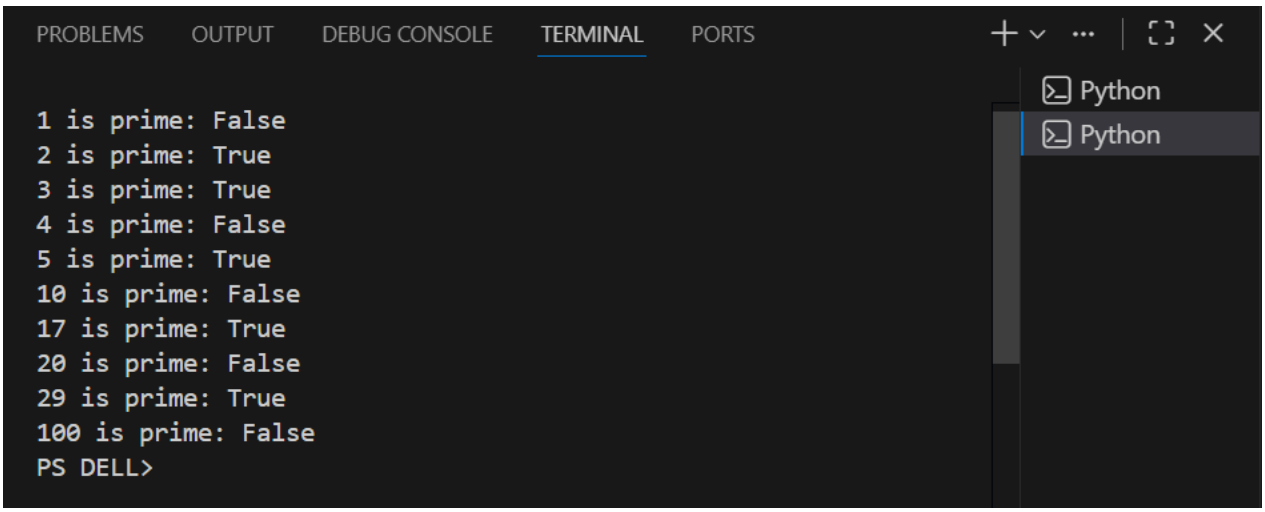
Prompt:

Generate a zero shot prompt to determine whether the given number is prime or not.

Code:

```python
def is_prime(n):
    if n < 2:
        return False

    if n == 2:
        return True

    if n % 2 == 0:
        return False

    i = 3
    while i * i <= n:
        if n % i == 0:
            return False
        i += 2

    return True


if __name__ == "__main__":
    test_numbers = [1, 2, 3, 4, 5, 10, 17, 20, 29, 100]

    for num in test_numbers:
        result = is_prime(num)
        print(f"{num} is prime: {result}")
```

Output:




Significance:

This outcome shows that zero-shot prompting works well when the problem is simple and clearly defined. Even without any examples or hints, the AI was able to understand the idea of prime numbers and apply the correct logic on its own. This indicates that the AI can rely on its basic knowledge and reasoning skills to solve standard programming problems. As a result, zero-shot prompting is useful for quick solutions and for checking an AI's natural problem-solving ability without providing extra guidance.

Task 2: One-shot: Provide one example:

Prompt:

Generate a function that calculates the sum of elements in a list.

Code:

TASK ai.py

C: > Users > KAVYA SRI > Downloads > ai.py > ...

```python
def calculate_sum(lst):
    return sum(lst)

if __name__ == "__main__":
    input_list = [1, 2, 3, 4]
    output = calculate_sum(input_list)

    print(f"Input: {input_list}")
    print(f"Output: {output}")

    print("\nAdditional test cases:")
    print(f"calculate_sum([5, 10, 15]) = {calculate_sum([5, 10, 1
    print(f"calculate_sum([0]) = {calculate_sum([0])}")
    print(f"calculate_sum([-1, 2, -3, 4]) = {calculate_sum([-1, 2
```

Output:

```
Input: [1, 2, 3, 4]
Output: 10

Additional test cases:
calculate_sum([5, 10, 15]) = 30
calculate_sum([0]) = 0
calculate_sum([-1, 2, -3, 4]) = 2
```

Significance:

A single input–output example was given to help the AI understand the task. This example made it clear that the goal was to add all elements in the list. As a result, the generated function correctly calculates the total for the given input. One-shot prompting minimized ambiguity and led to a more accurate solution. This shows how even one clear example can strongly influence and improve the AI's output.

Task 3: Few-shot: Give 2–3 examples

Prompt:

Generate function that extracts digits from an alphanumeric string

Ex:1 Input: "a1b2c3d4" -> output: "1234"

Ex:2 Input: "abc" -> output: ""

Ex:3 Input: "12345" -> output: "12345"

Code:

```
# Generate function that extracts digits from an alphanumeric string
# Ex:1 Input: "a1b2c3d4" -> output: "1234"
# Ex:2 Input: "abc" -> output: ""
# Ex:3 Input: "12345" -> output: "12345"
def extract_digits(s):
    return ''.join(filter(str.isdigit, s))
# Example usage:
input_str1 = "a1b2c3d4"
output1 = extract_digits(input_str1)
print("Extracted digits from '{}': {}".format(input_str1, output1))  #

input_str2 = "abc"
output2 = extract_digits(input_str2)
print("Extracted digits from '{}': {}".format(input_str2, output2))  #
input_str3 = "12345"
output3 = extract_digits(input_str3)
print("Extracted digits from '{}': {}".format(input_str3, output3))  #
# Output: Extracted digits from 'a1b2c3d4': 1234
# Output: Extracted digits from 'abc':
# Output: Extracted digits from '12345': 12345
```

Output:

```
Extracted digits from 'a1b2c3d4': 1234
Extracted digits from 'abc':
Extracted digits from '12345': 12345
```

Significance:

Several examples were provided to explain how digits should be separated from letters. These examples covered different cases, including mixed characters, inputs with no digits, and inputs containing only digits. By observing these patterns, the AI was able to understand the rule clearly and apply it correctly in its solution. Few-shot prompting helped improve accuracy and ensured better handling of edge cases, showing the value of giving multiple examples to guide the AI.

## Task 4: zero-shot vs few-shot prompting

Prompt:

Generate a function that counts the number of vowels in a string.

Code:

```python
#generate a function that counts the number of vowels in a string.
def count_vowels(s):
    vowels = 'aeiouAEIOU'
    return sum(1 for char in s if char in vowels)
# Example usage:
input_str = "Hello World"
vowel_count = count_vowels(input_str)
print("Number of vowels in '{}': {}".format(input_str, vowel_count))  # Output
```

Output:

```
Number of vowels in 'Hello World': 3
```

Prompt:

Generate a function that counts the number of vowels in a string.

Ex: Input: "Hello World" -> output: 3

Ex: Input: "Python" -> output: 1

Ex: Input: "AEIOU" -> output: 5

Code:

```python
#Generate a function that counts the number of vowels in a string.
# Ex: Input: "Hello World" -> output: 3
# Ex: Input: "Python" -> output: 1
# Ex: Input: "AEIOU" -> output: 5
def count_vowels(s):
    vowels = 'aeiouAEIOU'
    return sum(1 for char in s if char in vowels)
# Example usage:
input_str = "Hello World"
vowel_count = count_vowels(input_str)
print("Number of vowels in '{}': {}".format(input_str, vowel_count))  # Output: Number of vowels in 'Hello World': 3
# Output: Number of vowels in 'Hello World': 3

#Exanple usage:
input_str2 = "Python"
vowel_count2 = count_vowels(input_str2)
print("Number of vowels in '{}': {}".format(input_str2, vowel_count2))  # Output: Number of vowels in 'Python': 1

# Output: Number of vowels in 'Python': 1
input_str3 = "AEIOU"
vowel_count3 = count_vowels(input_str3)
print("Number of vowels in '{}': {}".format(input_str3, vowel_count3))
# Output: Number of vowels in 'AEIOU': 5
```

```python
#Generate a function that counts the number of vowels in a string.
# Ex: Input: "Hello World" -> output: 3
def count_vowels(s):
    vowels = 'aeiouAEIOU'
    return sum(1 for char in s if char in vowels)
# Example usage:
input_str = "Hello World"
vowel_count = count_vowels(input_str)
print("Number of vowels in '{}': {}".format(input_str, vowel_count))  # Output: Number of vowels in 'Hello World': 3
# Output: Number of vowels in 'Hello World': 3
```

Output:

```
Number of vowels in 'Hello World': 3
Number of vowels in 'Python': 1
Number of vowels in 'AEIOU': 5
```

Comparison Table:

```
# Comparsion table for both zero shot and few shot prompting for counting the number of vowels in a string. just a table not cpode

| Aspect                | Zero Shot Prompting                                          | Few Shot Prompting                                          |
|-----------------------|--------------------------------------------------------------|-------------------------------------------------------------|
| Definition            | Directly asks the model to count vowels without examples | Provides examples of input-output pairs for counting vowels |
| Examples Provided     | None                                                         | Multiple examples demonstrating the task                    |
| Clarity of Task       | May be less clear, relies on model's understanding           |
| More explicit due to examples provided                 |
| Accuracy              | May vary, depends on model's prior knowledge                 | Generally higher accuracy due
| Contextual Understanding | Relies on model's general knowledge                       | Enhanced by specific examples provided                      |
| Response Time         | Potentially faster as no examples need to be processed | May take slightly longer due to processing examples         |
| Use Case              | Quick tasks where examples are not feasible                  | Tasks requiring higher accuracy and clarity                 |
| Flexibility           | More flexible, can handle a wider range of inputs            | More rigid, may perform best on inputs similar to examples |
| Adaptability          | Adapts based on model's training data                        | Adapts based on provided examples                           |
```

Significance:

Zero-shot prompting relies on the AI's overall understanding of language, whereas few-shot prompting sets clear expectations by providing examples. When examples are included, the task becomes easier for the AI to interpret correctly. As a result, few-shot prompting produces more consistent and reliable outputs. This task clearly shows that examples help the AI understand requirements better and that well-structured input leads to higher-quality results.

Task 5: Use few-shot prompting with 3 sample inputs

Prompt:

Generate a function that determines the minimum of three numbers without using the built-in min () function. with 3 or more examples.

Ex 1: Input: (10, 5, 8) -> output: 5

Ex 2: Input: (-1, -5, -3) -> output: -5

Ex 3: Input: (0, 0, 1) -> output: 0

Code:

```python
#Generate a function that determines the minimum of three numbers without using the built-in min() function. with 3 or more examples.
#Ex 1: Input: (10, 5, 8) -> output: 5
#Ex 2: Input: (-1, -5, -3) -> output: -5
#Ex 3: Input: (0, 0, 1) -> output: 0
def minimum_of_three(a, b, c):
    if a <= b and a <= c:
        return a
    elif b <= a and b <= c:
        return b
    else:
        return c
# Example usage:
num1 = 10
num2 = 5
num3 = 8
min_value = minimum_of_three(num1, num2, num3)
print("Minimum of {}, {}, and {} is: {}".format(num1, num2, num3, min_value))
# Output: Minimum of 10, 5, and 8 is: 5
# Example 2
num4 = -1
num5 = -5
num6 = -3
min_value2 = minimum_of_three(num4, num5, num6)
print("Minimum of {}, {}, and {} is: {}".format(num4, num5, num6, min_value2))  #
# Output: Minimum of -1, -5, and -3 is: -5
# Example 3
num7 = 0
num8 = 0
num9 = 1
min_value3 = minimum_of_three(num7, num8, num9)
print("Minimum of {}, {}, and {} is: {}".format(num7, num8, num9, min_value3))  #
# Output: Minimum of 0, 0, and 1 is: 0
```

Output:


Significance:

Three different sample cases were provided to guide the AI's understanding. These examples included positive numbers, negative numbers, and repeated values, covering all the important conditions. Based on this guidance, the AI generated logic that correctly finds the smallest number without using built-in functions. Few-shot prompting helped the AI follow a clear and consistent reasoning process. This confirms the effectiveness of using examples to drive accurate AI outputs.