ASSIGNMENT:04

HALLTICKET:2303A51598

BATCH:29

SHAIK NEHA FARIYAL

Task Description-1

• Zero-shot: Prompt AI with only the instruction. Write a Python function to determine whether a given number is prime

PROMPT:

Write a python code to determine whether a given number is prime

```
#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 = 29
if is_prime(number):
    print(f"{number} is a prime number.")
else:
    print(f"{number} is not a prime number.")
```

OUTPUT:

```
PS C:\Users\nehaf\OneDrive\Desktop\java files> & C:/Users/nehaf/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/nehaf/OneDrive/Desktop/java files/la
b1.py"
29 is a prime number.
PS C:\Users\nehaf\OneDrive\Desktop\java files> & C:/Users/nehaf/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/nehaf/OneDrive/Desktop/java files/la
b1.py"
29 is a prime number.
PS C:\Users\nehaf\OneDrive\Desktop\java files>
```

JUSTIFICATION:

The In this task, zero-shot prompting was used, where only the instruction was provided to the AI without any examples. The prompt directly asked the AI to write a Python function to check whether a given number is prime.

Since no examples or additional context were given, the AI relied entirely on its internal knowledge of prime numbers and basic programming logic. The generated function correctly checks divisibility and handles edge cases such as numbers less than or equal to 1. This demonstrates that zero-shot prompting is effective for simple and well-known problems, but it may lack optimization or detailed explanations.

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.

PROMPT:

generate a Python function that provide one example input[1,2,3,4] and output 10 calculates the sum of elements in a list

```
[8]
0s
def count_chars(input_string):

    vowels = "aeiouAEIOU"
    consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"
    digits = "0123456789"

    vowel_count = 0
    consonant_count = 0
    digit_count = 0

    for char in input_string:
        if char in vowels:
            vowel_count += 1
        elif char in consonants:
            consonant_count += 1
        elif char in digits:
            digit_count += 1

    return {
        "vowels": vowel_count,
        "consonants": consonant_count,
        "digits": digit_count
    }
```

OUTPUT:

```
PS C:\Users\nehaf\OneDrive\Desktop\java files>
PS C:\Users\nehaf\OneDrive\Desktop\java files> & C:/Users/nehaf/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/nehaf/OneDrive/Desktop/java files/lab1.py"
The sum of elements in [1, 2, 3, 4] is 10.
PS C:\Users\nehaf\OneDrive\Desktop\java files>
```

Open Website Generate Commit Message Ln 24, Col 62 Spaces: 2 UTF-8 CRLF { } Python 3.13.1 Go Live BLACKB

JUSTIFICATION:

This task one-shot prompting was applied by providing one input-output example:

Input: [1, 2, 3, 4]

Output: 10

This single example helped the AI clearly understand that the task was to calculate the sum of all elements in a list. The AI generated a correct Python function using a loop or built-in operations based on the given example. Compared to zero-shot prompting, the presence of one example improved clarity and reduced ambiguity. This shows that one-shot prompting is useful when a small hint is enough to guide the model.

Task Description-3

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

alphanumeric string.

PROMPT:

Create a function that extracts digits from an alphanumeric string with 3 examples

```python
#Create a function that extracts digits from an alphanumeric string with 3 examples
def extract_digits(input_string):
    return ''.join(filter(str.isdigit, input_string))
#Example usage
example1 = "abc123def"
example2 = "a1b2c3"
example3 = "no_digits_here!"
print(f"Digits in '{example1}': {extract_digits(example1)}")
print(f"Digits in '{example2}': {extract_digits(example2)}")
print(f"Digits in '{example3}': {extract_digits(example3)}")
    #print(f"{number} is not a prime number.")

def sum_of_elements(input_list):
    return sum(input_list)
#Example usage
example_input = [1, 2, 3, 4]
result = sum_of_elements(example_input)
print(f"The sum of elements in {example_input} is {result}.")
```

OUTPUT:

```
PS C:\Users\nehaf\OneDrive\Desktop\java files> & C:/Users/nehaf/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/nehaf/OneDrive/Desktop/java files/lab1.py"
Digits in 'abc123def': 123
Digits in 'a1b2c3': 123
Digits in 'no_digits_here!':
The sum of elements in [1, 2, 3, 4] is 10.
PS C:\Users\nehaf\OneDrive\Desktop\java files>
```

JUSTIFICATION:

In this task, few-shot prompting was used by providing two to three examples showing how digits should be extracted from alphanumeric strings. These examples clearly demonstrated the expected pattern and output format.

With multiple examples, the AI was able to understand that:

Only numeric characters should be extracted

All other alphabetic or special characters should be ignored

As a result, the generated function accurately returns only the digits from the input string. This task proves that few-shot prompting is very effective for pattern-based problems, where examples help the AI generalize the solution correctly.

Task Description-4

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

PROMPT:

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

CODE:

```python
#Compare zero-shot vs few-shot prompting for generating a function that counts the number of vowels in a string.
def count_vowels(input_string):
    vowels = 'aeiouAEIOU'
    return sum(1 for char in input_string if char in vowels)
#Example usage
example_string = "Hello, World!"
vowel_count = count_vowels(example_string)
print(f"The number of vowels in '{example_string}' is {vowel_count}.")
def sum_of_elements(input_list):
    return sum(input_list)
#Example usage
example_input = [1, 2, 3, 4]
result = sum_of_elements(example_input)
print(f"The sum of elements in {example_input} is {result}.")
```

Output:

```
PS C:\Users\nehaf\OneDrive\Desktop\java files> & C:/Users/nehaf/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/nehaf/OneDrive/Desktop/java files/la
b1.py"
The number of vowels in 'Hello, World!' is 3.
The sum of elements in [1, 2, 3, 4] is 10.
PS C:\Users\nehaf\OneDrive\Desktop\java files>
```

JUSTIFICATION:

This task compares zero-shot prompting and few-shot prompting for generating a function that counts vowels in a string.

In the zero-shot approach, the AI generated a basic solution that counted vowels correctly but lacked clarity in handling uppercase letters or explaining the logic.

In the few-shot approach, examples showed how vowels are counted in different strings. This helped the AI generate a more accurate and robust function, often including case-insensitive handling.

The comparison shows that examples significantly improve AI performance, making the output more reliable and easier to understand. Few-shot prompting reduces errors and improves logical completeness.

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

Prompt:

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

```
#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()
def min_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
minimum = min_of_three(num1, num2, num3)
print(f"The minimum of {num1}, {num2}, and {num3} is {minimum}.")
```

Output:

```
PS C:\Users\nehaf\OneDrive\Desktop\java files> & C:/Users/nehaf/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/nehaf/OneDrive/Desktop/java files/la
b1.py"
The minimum of 10, 5, and 8 is 5.
PS C:\Users\nehaf\OneDrive\Desktop\java files>
```

Open Website Generate Commit Message Ln 59, Col 1 Spaces: 2 UTF-8 CRLF { } Python 3.13.1 Go Live BLACKBO

JUSTIFICATIONS:

In this task, few-shot prompting with three sample inputs was used to generate a function that finds the minimum of three numbers without using the built-in min() function.

The provided examples helped the AI identify comparison patterns between three values and implement conditional logic correctly. The generated function successfully handles all possible cases, such as equal values and negative numbers. This task highlights how few-shot prompting helps the AI learn decision-making logic from examples rather than relying on built-in shortcuts.