SR UNIVERSITY

AI Assistant Coding
Assignment-4

A.HARSHITH
2303A52422
B-37

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

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

Code:

```
def is_prime(n: int) -> bool:
    """Return True if n is a prime number, otherwise False."""
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False

    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True


if __name__ == "__main__":
    num = int(input("Enter a number: "))
    print("Prime" if is_prime(num) else "Not prime")
```

s

Output:

```
Enter a number: 5
Prime
```

Justification:
In zero-shot prompting, only the instruction is provided without examples. The AI relies on its prior knowledge to generate the solution. This demonstrates the model's ability to understand the problem and apply correct logical conditions independently.

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.

Prompt:
Example:
Input: [1, 2, 3, 4]
Output: 10
Write a Python function to calculate the sum of elements in a list.

Code:

```
def sum_list(values: list[int | float]) -> int | float:
    """Return the sum of all elements in the list."""
    total: int | float = 0
    for v in values:
        total += v
    return total


if __name__ == "__main__":
    print("sum_list([1, 2, 3, 4]) ->", sum_list([1, 2, 3, 4]))
```

Output:

```
sum_list([1, 2, 3, 4]) -> 10
```

Justification:
One-shot prompting provides a single example to guide the AI. The example helps clarify the expected operation and output format. This improves accuracy by reducing ambiguity compared to zero-shot prompting.

SR UNIVERSITY

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.

Prompt:

Example1:Input:"a1b2c3"→Output:"123"
Example2:Input:"abc2025xyz"→Output:"2025"
Example 3: Input: "x9y" → Output: "9"
Write a Python function to extract digits from an alphanumeric string.

Code:

```
def extract_digits(text: str) -> str:
    """Extract and return all digit characters from an alphanumeric string."""
    return "".join(ch for ch in text if ch.isdigit())


if __name__ == "__main__":
    print('extract_digits("a1b2c3") ->', extract_digits("a1b2c3"))
    print('extract_digits("abc2025xyz") ->', extract_digits("abc2025xyz"))
    print('extract_digits("x9y") ->', extract_digits("x9y"))
```

Output:

```
extract_digits("a1b2c3") -> 123
extract_digits("abc2025xyz") -> 2025
extract_digits("x9y") -> 9
```

Justification:

Few-shot prompting uses multiple examples to show clear input–output patterns. These examples help the AI identify the exact rule for extracting digits. As a result, the generated function is more precise and reliable.

SR UNIVERSITY

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.

Prompt:

Zero-Shot Prompt

Write a Python function to count the number of vowels in a string.

Few-Shot Prompt

Example 1: "hello" → 2
Example 2: "AI Tools" → 4
Example 3: "xyz" → 0
Write a Python function to count vowels in a string.

Code:

Zero-Shot Prompt

```
def count_vowels(text: str) -> int:
    """Return the number of vowels (a, e, i, o, u) in a string."""
    vowels = {"a", "e", "i", "o", "u"}
    return sum(1 for ch in text.lower() if ch in vowels)


if __name__ == "__main__":
    print("Vowels -> ", count_vowels("hello"))
    print("Vowels -> ", count_vowels("AI Code"))
```

Few-Shot Prompt

```
def count_vowels(s):
    vowels = set("aeiouAEIOU")
    return sum(1 for ch in s if ch in vowels)

if __name__ == "__main__":
    print('count_vowels("hello") ->', count_vowels("hello"))
    print('count_vowels("AI Tools") ->', count_vowels("AI Tools"))
    print('count_vowels("xyz") ->', count_vowels("xyz"))
```

Output:

Zero-Shot Prompt

```
Vowels ->  2
Vowels ->  4
```

Few-Shot Prompt

```
count_vowels("hello") -> 2
count_vowels("AI Tools") -> 4
count_vowels("xyz") -> 0
```

Comparison & Explanation:

Aspect	Zero-Shot	Few-Shot
Logic clarity	Basic loop	Optimized logic
Accuracy	Correct	Correct
Code quality	Simple	Cleaner & concise
Guidance	No examples	Clear behavior from examples

Justification:

Few-shot prompting improves output quality because examples clarify expected behavior, edge cases, and efficiency. The model generates more refined and optimized solutions when patterns are visible.

SR UNIVERSITY

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.

Prompt:

Example 1: (3, 7, 5) → 3
Example 2: (10, 2, 8) → 2
Example 3: (4, 4, 9) → 4
Write a Python function to find the minimum of three numbers without using min()

Code:

```
def min_of_three(a: int | float, b: int | float, c: int | float) -> int | float:
    """Return the minimum of three numbers without using min()."""
    smallest = a
    if b < smallest:
        smallest = b
    if c < smallest:
        smallest = c
    return smallest

if __name__ == "__main__":
    print("min_of_three(3, 7, 5) ->", min_of_three(3, 7, 5))
    print("min_of_three(10, 2, 8) ->", min_of_three(10, 2, 8))
    print("min_of_three(4, 4, 9) ->", min_of_three(4, 4, 9))
```

OUTPUT:

```
min_of_three(3, 7, 5) -> 3
min_of_three(10, 2, 8) -> 2
min_of_three(4, 4, 9) -> 4
```

JUSTIFICATION:

Few-shot examples help the AI understand comparison logic and edge cases without using built-in functions. The model learns the decision pattern from examples and applies it correctly. This results in accurate handling of all possible inputs.