

Assignment – 4.2

AI Assisted Coding

Name: Mohammed Rahil

Batch: 21

HtNo: 2303A51433

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:

```
1  '''Write a Python function that checks if a number is prime.
2  | with example usage.
3  | with ouptput and explanation.'''
4
5
6  def is_prime(n):
7      """Check if a number is prime."""
8      if n <= 1:
9          return False
10     for i in range(2, int(n**0.5) + 1):
11         if n % i == 0:
12             return False
13     return True
14 # Example usage:
15 number = 29
16 if is_prime(number):
17
18     print(f"{number} is a prime number.")
19 else:
20     print(f"{number} is not a prime number.\n")# Output: 29 is a prime number.
21
```

Output:

29 is a prime number.

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:

```
24     '''generate a function that calculates the sum of elements in a list and  Provide one example
25     input: [1, 2, 3, 4]
26     | Output: 10 and explanation.with comments.
27     '''
28
29     def sum_of_list(lst):
30         """Calculate the sum of elements in a list."""
31         total = 0
32         for num in lst:
33             total += num # Add each number to total
34         return total
35     # Example usage:
36     example_list = [1, 2, 3, 4]
37     result = sum_of_list(example_list)
38
39     print(f"The sum of {example_list} is {result}.\\n") # Output: 10
40
41
```

Output:

```
The sum of [1, 2, 3, 4] is 10.
```

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:

```
40
47 def extract_digits(s):
48     """Extract digits from an alphanumeric string."""
49     digits = ''.join(char for char in s if char.isdigit())
50     return digits
51 # Example usage:
52 examples = [
53     "abc123",          # Example 1
54     "a1b2c3d4",        # Example 2
55     "no_digits!",       # Example 3
56     "2024year"         # Example 4
57 ]
58 for example in examples:
59     result = extract_digits(example)
60     print(f"Extracted digits from '{example}': '{result}'\n") # output and explanation
61
62
```

Output:

```
Extracted digits from 'abc123': '123'

Extracted digits from 'a1b2c3d4': '1234'

Extracted digits from 'no_digits!': ''

Extracted digits from '2024year': '2024'
```

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:

```
64  '''generate a function that counts the number of vowels in a string
65  while Comparing with zero-shot vs few-shot
66  Output comparison + student explanation on how examples helped the model to perform better.
67  with comments and explaination.'''
68
69 def count_vowels(s):
70     """Count the number of vowels in a string."""
71     vowels = 'aeiouAEIOU'
72     count = 0
73     for char in s:
74         if char in vowels:
75             count += 1 # Increment count for each vowel found
76     return count
77 # Example usage:
78 example_string = "Hello World"
79 vowel_count = count_vowels(example_string)
80 print(f"The number of vowels in '{example_string}' is {vowel_count}.\\n") # Output: 3
81
```

Output:

```
The number of vowels in 'Hello World' is 3.
```

Few Shot:

Code:

```
'''  
Write a Python function to count the number of vowels in a string.  
Examples:  
Input: "hello" → Output: 2  
Input: "chatGPT" → Output: 2  
Input: "AEIOU" → Output: 5  
...  
def count_vowels(input_string):  
    vowels = 'aeiouAEIOU'  
    count = sum(1 for char in input_string if char in vowels)  
    return count  
# Example usage  
dynamic_input = input("Enter a string: ")  
result = count_vowels(dynamic_input)  
print("Number of vowels:", result)
```

Output:

```
Enter a string: Dhanush
Number of vowels: 2
```

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:

```
84  '''generate a function that determines the minimum of three numbers
85  without using the built-in min() function with comments.
86  input: should Use few-shot prompting with 3 sample
87  outputs: all cases with correct logic based on example patterns
88  explanation: how few-shot helped in better understanding of the problem.
89
90  ...
91
92  def min_of_three(a, b, c):
93      # Assume a is the minimum
94      minimum = a
95      # Compare with b
96      if b < minimum:
97          minimum = b
98      # Compare with c
99      if c < minimum:
100         minimum = c
101     return minimum
102 # Example usage:
103 examples = []
104     (3, 1, 2), # Example 1: minimum is 1
105     (10, 5, 7), # Example 2: minimum is 5
106     (-1, -5, -3) # Example 3: minimum is -5
107
108 for a, b, c in examples:
109     result = min_of_three(a, b, c)
110     print(f"The minimum of {a}, {b}, {c} is {result}.") # Output and explanation
```

Output:

```
The minimum of (3, 1, 2) is 1.
The minimum of (10, 5, 7) is 5.
The minimum of (-1, -5, -3) is -5.
```

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.