
Lab Assignment # 4

Name of Student : T. Sai Teja
Enrollment No. : 2303A51535
Batch No. : 22

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

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

Code-

```
# Task 1: Zero-shot Prompting
# Write a function to check whether a 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

# Sample Output
print("Task 1 Output:")
print(is_prime(7))    # True
print(is_prime(10))   # False
```

Output-

```
Task 1 Output:
True
False
```

Justification-

- A prime number is greater than 1 and divisible only by 1 and itself.
- The loop checks divisibility up to \sqrt{n} to optimize performance.
- No examples were provided—this is **pure instruction-based prompting (zero-shot)**.

Task 2: One-shot Prompting

Prompt- Example: Input: [1, 2, 3, 4] → Output: 10

Using this example, write a Python function that calculates the sum of elements in a list.

Code-

```
# -----  
# Task 2: One-shot Prompting  
# Example: Input [1,2,3,4] → Output 10  
# Write a function to calculate the sum of list elements  
# -----  
  
def list_sum(numbers):  
    total = 0  
    for num in numbers:  
        total += num  
    return total  
  
# Sample Output  
print("\nTask 2 Output:")  
print(list_sum([1, 2, 3, 4])) # 10
```

Output-

```
Task 2 Output:  
10
```

Justification-

- The single example guides the AI to understand the expected behavior.
- The function iterates through the list and accumulates the sum.
- This demonstrates **one-shot prompting**, where one example improves clarity.

Task 3: Few-shot Prompting

Prompt- Examples:

Input: "a1b2c3" → Output: "123"

Input: "abc456" → Output: "456"

Input: "9x8y" → Output: "98"

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

Code-

```
# -----  
# Task 3: Few-shot Prompting  
# Extract digits from an alphanumeric string  
# Examples:  
# "a1b2c3" → "123"  
# "abc456" → "456"  
# -----  
  
def extract_digits(text):  
    result = ""  
    for char in text:  
        if char.isdigit():  
            result += char  
    return result  
  
# Sample Output  
print("\nTask 3 Output:")  
print(extract_digits("a1b2c3")) # 123  
print(extract_digits("9x8y")) # 98
```

Output-

```
Task 3 Output:  
123  
98
```

Justification-

- Multiple examples help the model identify a pattern.
- The function checks each character and appends digits only.
- This is few-shot prompting, which improves accuracy for pattern-based tasks.

Task 4: Comparison: Zero-shot vs Few-shot Prompting

Prompt-

Zero-shot Prompt:

Write a Python function to count vowels in a string.

Few-shot Prompt

Input: "hello" → Output: 2

Input: "AI Coding" → Output: 4

Write a Python function to count vowels.

Code-

```
# -----
# Task 4: Comparison - Zero-shot vs Few-shot Prompting
# Count number of vowels in a string
# -----

def count_vowels(text):
    vowels = "aeiouAEIOU"
    count = 0
    for char in text:
        if char in vowels:
            count += 1
    return count

# Sample Output
print("\nTask 4 Output:")
print(count_vowels("hello"))      # 2
print(count_vowels("AI Coding"))  # 4
```

Output-

```
Task 4 Output:
2
4
```

Justification-

Aspect	Zero-shot	Few-shot
Guidance	Only instruction	Clear examples
Code style	Basic loop	Optimized logic
Accuracy	Correct	Correct
Readability	Moderate	High

Task 5: Few-shot Prompting (Minimum of Three Numbers)**Prompt-**

input: (3, 5, 1) → Output: 1

Input: (10, 7, 9) → Output: 7

Input: (4, 4, 2) → Output: 2

Write a Python function to find the minimum of three numbers without using min().

Code-

```
# -----  
# Task 5: Few-shot Prompting  
# Find the minimum of three numbers without using min()  
# Examples:  
# (3,5,1) → 1  
# (10,7,9) → 7  
# -----  
  
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  
  
# Sample Output  
print("\nTask 5 Output:")  
print(minimum_of_three(3, 5, 1))    # 1  
print(minimum_of_three(10, 7, 9))   # 7
```

Output-

```
Task 5 Output:  
1  
7
```

Justification-

- The function uses conditional comparisons.
- Few-shot examples ensure all logical cases are handled.
- Built-in min() is avoided as instructed.