

## AI ASSISTED CODING – LAB ASSIGNMENT 4.2

### Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques

**Name:** Pavan Adhitya Bollam

**H.TNO:**2303A51047

**Course Title:** AI Assisted Coding

**Assignment No:** 4.2

**Year / Semester:** III / II

**Regulation:** R23

---


#### Task 1: Zero-Shot Prompting – Prime Number Check

##### Prompt (Zero-Shot)

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

##### AI Generated Code

python

 Copy code

```
def is_prime(n):  
    if n <= 1:  
        return False  
    for i in range(2, n):  
        if n % i == 0:  
            return False  
    return True
```

##### Sample Output

Input: 7 → Output: Prime

Input: 10 → Output: Not Prime

##### Observation

Since no examples were given, the AI generated a basic solution.

The logic is correct but not optimized.

##### Conclusion

Zero-shot prompting gives working code, but it may not be efficient or well-optimized.

---

#### Task 2: One-Shot Prompting – Sum of Elements in a List

##### Prompt (One-Shot)

Example:


Input: [1, 2, 3, 4]

Output: 10

Write a Python function to calculate the sum of elements in a list.

### AI Generated Code

python

 Copy code

```
def list_sum(arr):  
    total = 0  
    for i in arr:  
        total += i  
    return total
```

### Sample Output

Input: [5, 10, 15]

Output: 30

### Observation

The single example helped the AI clearly understand the requirement.

The code is simple and correct.

### Conclusion

One-shot prompting improves clarity and reduces confusion compared to zero-shot prompting.

---

## Task 3: Few-Shot Prompting – Extract Digits from Alphanumeric String

### Prompt (Few-Shot Examples)

Input: "a1b2c3" → Output: "123"


Input: "x9y8" → Output: "98"

Input: "abc" → Output: ""

Write a Python function to extract digits from an alphanumeric string.

### AI Generated Code

python

 Copy code

```
def extract_digits(s):  
    result = ""  
    for ch in s:  
        if ch.isdigit():  
            result += ch  
    return result
```

### Sample Output

Input: "p4q5r6"

Output: "456"

### Observation

Because multiple examples were provided, the AI generated accurate and reliable code.

### Conclusion

Few-shot prompting helps the AI understand patterns better and handle different cases correctly.

---


## Task 4: Comparison – Zero-Shot vs Few-Shot (Vowel Count)

### Zero-Shot Prompt

Write a Python function to count vowels in a string.

### Zero-Shot Code

python

 Copy code

```
def count_vowels(s):  
    count = 0  
    for ch in s:  
        if ch in "aeiou":  
            count += 1  
    return count
```

---

### Few-Shot Prompt

Examples:

"hello" → 2


"aeiou" → 5

"xyz" → 0

Write a Python function to count vowels in a string.

### Few-Shot Code

python

 Copy code

```
def count_vowels(s):  
    count = 0  
    for ch in s.lower():  
        if ch in "aeiou":  
            count += 1  
    return count
```

---

### Comparison

Feature	Zero-Shot	Few-Shot
Handles uppercase	No	Yes
Accuracy	Medium	High
Code quality	Basic	Better

### Observation

Few-shot prompting improved accuracy by handling uppercase letters and edge cases.

### Conclusion

Examples help the AI generate more complete and correct code.

---

### Task 5: Few-Shot Prompting – Minimum of Three Numbers

#### Prompt (Few-Shot Examples)

Input: (3, 7, 5) → Output: 3


Input: (10, 2, 8) → Output: 2

Input: (6, 6, 9) → Output: 6

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

#### AI Generated Code

python

 Copy code

```
def minimum(a, b, c):  
    if a <= b and a <= c:  
        return a  
    elif b <= a and b <= c:  
        return b  
    else:  
        return c
```

### Sample Output

Input: (4, 9, 1)

Output: 1

### Observation

The examples helped the AI cover all comparison cases correctly.

### Conclusion

Few-shot prompting is very effective when logic depends on conditions and comparisons.