

## AI Assisted Coding – Assignment 4.2

**Course Code:** 23CS002PC304

**Course Title:** AI Assisted Coding

**Lab:** Advanced Prompt Engineering – Zero-shot, One-shot, Few-shot Techniques

**Student Name:** Ravula seetharam Reddy

**Roll no :** 2303A52440

**Batch :** 34

**Year/Sem:** III / II

**Regulation:** R23

---

### Task Description – 1: Zero-shot Prompting

#### Prompt Used (Zero-shot)

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

#### Code

```
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
```

#### Explanation

- Zero-shot prompting provides only instructions without examples.
- The function checks divisibility from 2 to  $\sqrt{n}$ .
- If any divisor exists, the number is not prime.

#### Sample Output

- `is_prime(7)` → True
- `is_prime(10)` → False

main.py	Run	Output
<pre>1- def is_prime(n): 2-     if n &lt;= 1: 3-         return False 4-     for i in range(2, int(n ** 0.5) + 1): 5-         if n % i == 0: 6-             return False 7-     return True 8 9 print(is_prime(7)) 10 print(is_prime(10)) 11</pre>		<pre>True False</pre>

## Task Description – 2: One-shot Prompting

### Prompt Used (One-shot)

“Input: [1, 2, 3, 4], Output: 10. Write a Python function that calculates the sum of elements in a list.”

### Code

```
def sum_list(lst):
    total = 0
    for num in lst:
        total += num
    return total
```

### Explanation

- One example guides the AI toward summation logic.
- Iterates through the list and accumulates the total.

### Sample Output

- `sum_list([1, 2, 3, 4]) → 10`

### Screenshot Placeholder:

main.py	Run	Output
<pre>1- def sum_list(lst): 2-     total = 0 3-     for num in lst: 4-         total += num 5-     return total 6 7 print(sum_list([1, 2, 3, 4])) 8 9</pre>		<pre>10 === Code Execution Successful ===</pre>

## Task Description – 3: Few-shot Prompting

### Prompt Used (Few-shot)

“Input: ‘a1b2c3’ → Output: ‘123’

Input: ‘abc456’ → Output: ‘456’

Input: ‘9x8y’ → Output: ‘98’

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

### Code

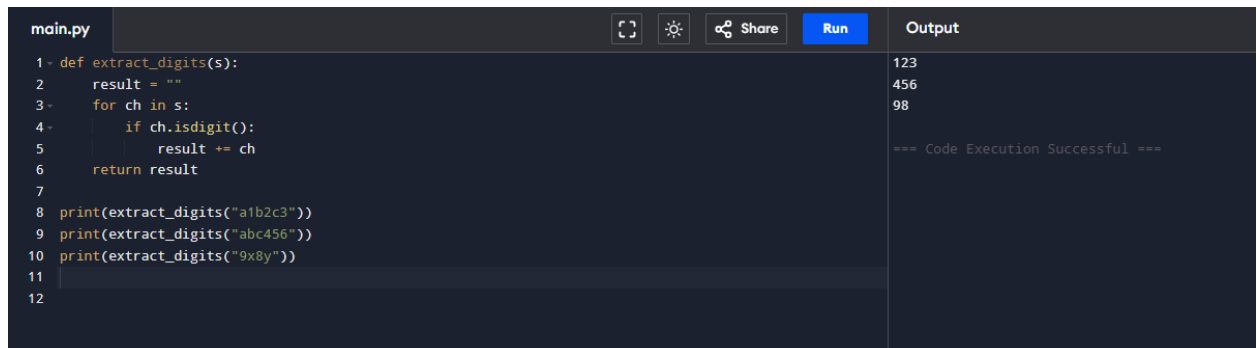
```
def extract_digits(s):  
    result = ""  
    for ch in s:  
        if ch.isdigit():  
            result += ch  
    return result
```

### Explanation

- Multiple examples clarify the pattern.
- Digits are identified using `isdigit()`.

### Sample Output

- `extract_digits("a1b2c3")` → “123”



The screenshot shows a code editor with a file named 'main.py'. The code defines a function `extract_digits(s)` that iterates through each character in the string `s` and appends it to a `result` string if it is a digit. The function is then called with three test cases: `extract_digits("a1b2c3")`, `extract_digits("abc456")`, and `extract_digits("9x8y")`. The output of the code execution is displayed on the right side of the editor, showing the results: 123, 456, and 98. A message at the bottom of the output area states "=== Code Execution Successful ===".

```
main.py  
1- def extract_digits(s):  
2-     result = ""  
3-     for ch in s:  
4-         if ch.isdigit():  
5-             result += ch  
6-     return result  
7-  
8- print(extract_digits("a1b2c3"))  
9- print(extract_digits("abc456"))  
10- print(extract_digits("9x8y"))  
11-  
12-  
Output  
123  
456  
98  
=== Code Execution Successful ===
```

## Task Description – 4: Zero-shot vs Few-shot Comparison

### Zero-shot Prompt

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

### Zero-shot Code

```
def count_vowels_zero(s):  
    vowels = "aeiouAEIOU"  
    count = 0  
    for ch in s:  
        if ch in vowels:
```

```
        count += 1
    return count
```

## Few-shot Prompt

“Input: ‘hello’ → Output: 2

Input: ‘AI’ → Output: 2

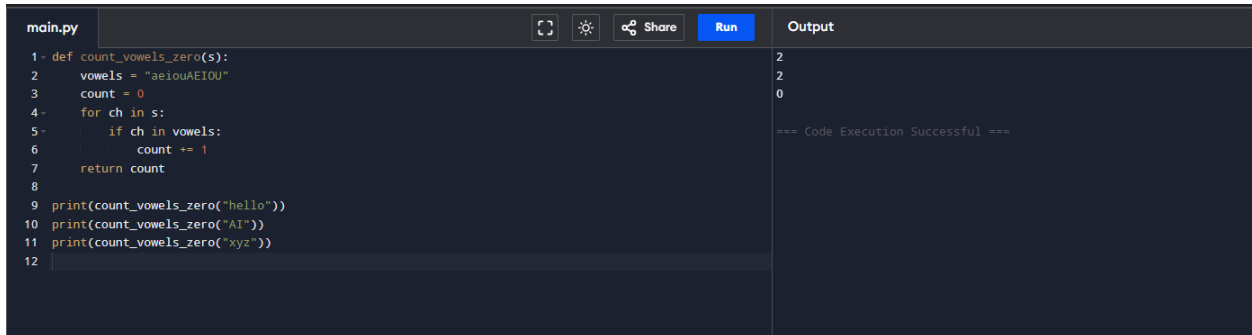
Input: ‘xyz’ → Output: 0

Write a Python function to count vowels in a string.”

## Few-shot Code

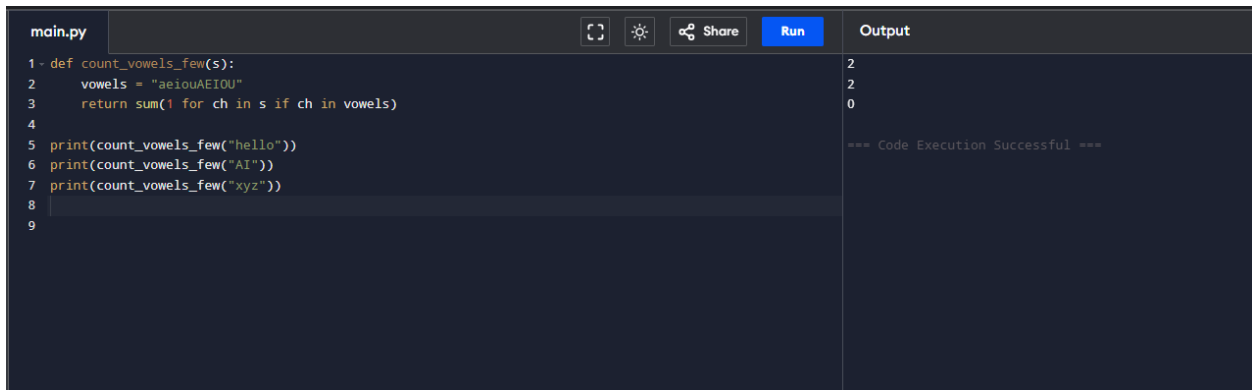
```
def count_vowels_few(s):
    vowels = "aeiouAEIOU"
    return sum(1 for ch in s if ch in vowels)
```

 **Screenshot Placeholder (Zero-shot vowel count):** > Paste screenshot of zero-shot code execution and output here.



```
main.py  [Run] [Share] [Output]
1- def count_vowels_zero(s):
2-     vowels = "aeiouAEIOU"
3-     count = 0
4-     for ch in s:
5-         if ch in vowels:
6-             count += 1
7-     return count
8-
9- print(count_vowels_zero("hello"))
10- print(count_vowels_zero("AI"))
11- print(count_vowels_zero("xyz"))
12-
Output
2
2
0
=== Code Execution Successful ===
```

 **Screenshot Placeholder (Few-shot vowel count):**



```
main.py  [Run] [Share] [Output]
1- def count_vowels_few(s):
2-     vowels = "aeiouAEIOU"
3-     return sum(1 for ch in s if ch in vowels)
4-
5- print(count_vowels_few("hello"))
6- print(count_vowels_few("AI"))
7- print(count_vowels_few("xyz"))
8-
9-
Output
2
2
0
=== Code Execution Successful ===
```

## Comparison & Explanation

- Zero-shot gives a basic loop-based solution.
  - Few-shot improves clarity and efficiency.
  - Examples help AI understand edge cases.
-

## Task Description – 5: Few-shot Prompting (Minimum of Three Numbers)

### Prompt Used (Few-shot)

“Input: (3, 5, 1) → Output: 1

Input: (10, 2, 7) → Output: 2

Input: (4, 4, 9) → Output: 4

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

### Code

```
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
```

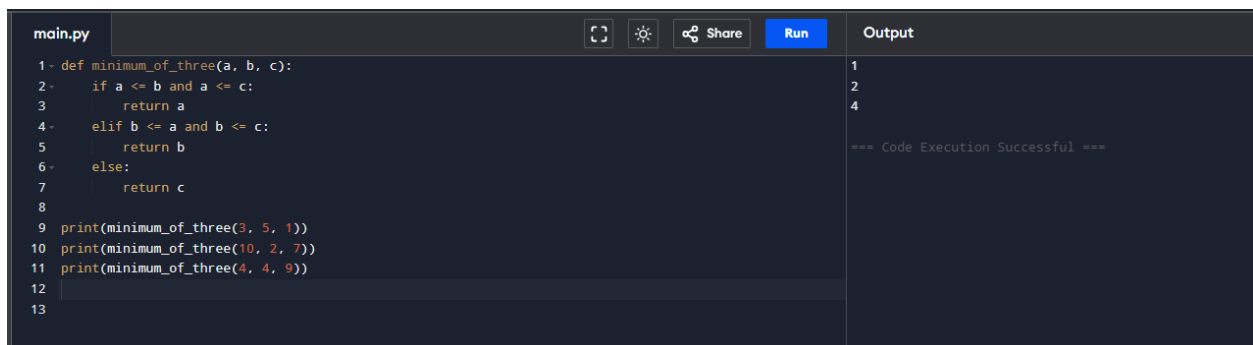
### Explanation

- Conditional comparisons handle all cases.
- No built-in min() function is used.

### Sample Output

- minimum\_of\_three(3, 5, 1) → 1

### Screenshot Placeholder:



The screenshot shows a code editor with a dark theme. The file is named 'main.py'. The code defines a function 'minimum\_of\_three(a, b, c)' that returns the minimum of three numbers using conditional logic. Below the function definition, there are three print statements: 'print(minimum\_of\_three(3, 5, 1))', 'print(minimum\_of\_three(10, 2, 7))', and 'print(minimum\_of\_three(4, 4, 9))'. The output panel on the right shows the results of these print statements: '1', '2', and '4'. Below the output, it says '=== Code Execution Successful ==='. The editor has a 'Run' button and a 'Share' button.

---

## Conclusion

- Zero-shot prompting works for simple logic.
- One-shot improves accuracy with guidance.
- Few-shot produces the most reliable and optimized results.

- Prompt richness directly impacts output quality.