

Sarika Palle

2403a51l33

B-52

Lab 4

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

Task Description-1: Zero-shot Prompting

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

OUTPUT:

The screenshot shows the Visual Studio Code interface with an AI-assisted coding feature. The code editor displays a Python script for calculating the sum of elements in a list:

```
15 ## Example: input: [1, 2, 3, 4], output: 10
16 ## Write a python function to calculate the sum of elements in a list.
17 """def sum_list(lst):
18     total = 0
19     for num in lst:
20         total += num
21
22     return total
23
24 print(sum_list([1, 2, 3, 4]))
```

The terminal window shows the command being run and its output:

```
PS C:\Users\sarik\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/Desktop/AI ASSISTED CODING/ASSIGN-4-2.py"
● True
○ PS C:\Users\sarik\Desktop\AI ASSISTED CODING>
```

The status bar at the bottom indicates the file is 3.14.2 and has 4 changes.

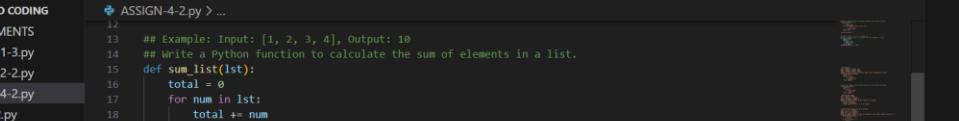
Explanation:

1. Zero-shot prompting provides only instructions, no examples.
 2. The AI correctly implemented:
 - Prime definition logic
 - Square-root optimization
 3. Demonstrates that simple logical problems work well with zero-shot prompts.

Task Description-2: One-shot Prompting

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

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



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a tree view with files like `ASSIGN-4-2.py`, `ASSIGN-4-2.py > ...`, `ASSIGN-1-3.py`, `ASSIGN-2-2.py`, `ASSIGN-4-2.py` (selected), `DAY--1-2.py`, `LAB-2.py`, `LAB(2)(PRACTICE SESS..`, and `Prime.py`.
- Code Editor:** Displays the content of `ASSIGN-4-2.py`. The code defines a function `sum_list` that calculates the sum of elements in a list. It includes a docstring and a call to the function with the list `[1, 2, 3, 4]`.
- Right Panel:** Features an "AI ASSISTED CODING" section with a progress bar at 100%, a "BLACKBOX" button, and a "Build with Agent" button. Below it is a message stating "AI responses may be inaccurate." and "Generate Agent Instructions to onboard AI onto your codebase."

OUTPUT:

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows files like `ASSIGN-4-2.py`, `DAY-12.py`, `LAB-2.py`, `LAB(2)/PRACTICE SESSION`, and `Prime.py`.
- Code Editor:** Displays Python code for calculating the sum of a list:

```
total = 0
for num in lst:
    total += num
return total

print(sum_list([1, 2, 3, 4]))
```
- Terminal:** Shows the command run in the terminal:

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-4-2.py"
```
- Status Bar:** Shows file paths, line numbers (Ln 34, Col 1), and other status information.

Explanation:

1. One example clarifies the expected behavior.
 2. The AI correctly inferred:

Iteration over list

Accumulation of sum

3. The example helped remove ambiguity.

Task Description-3: Few-shot Prompting

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

Examples:

Input: "abc123" → Output: "123"

Input: "a1b2c3" → Output: "123"

Input: "2024AI" → Output: "2024"

The screenshot shows the AI ASSISTED CODING interface. In the code editor, the file `ASSIGN-4-2.py` contains the following code:

```
22
23 ##### Examples:
24 Input: "abc123" → Output: "123"
25 Input: "a1b2c3" → Output: "123"
26 Input: "2024AI" → Output: "2024"
27 ## Write a Python function to extract digits from an alphanumeric string."""
28 def extract_digits(text):
29     digits = ""
30     for ch in text:
31         if ch.isdigit():
32             digits += ch
33     return digits
34 print(extract_digits("Lab4AI2026"))
35
36
37
38
39
40
41
42
43
44
45
46
47
48
```

The terminal window at the bottom shows the output of running the script:

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-4-2.py"
42026
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

OUTPUT:

The screenshot shows the terminal window of the AI ASSISTED CODING interface. The command `& C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-4-2.py"` was run, resulting in the output `42026`.

Explanation:

1. Few-shot prompting provides pattern recognition.
2. AI correctly:

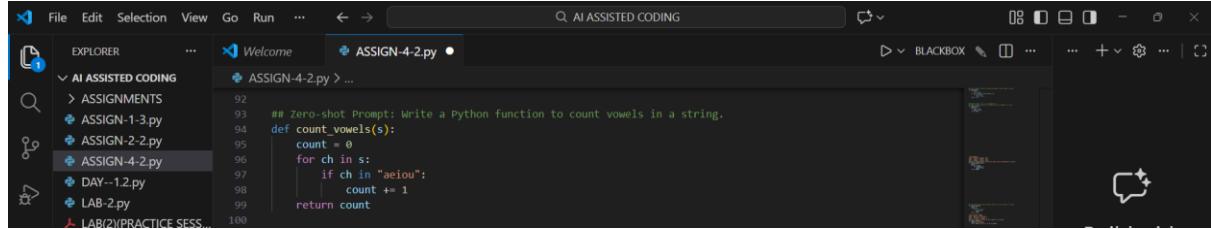
Identified digit extraction rule

Ignored alphabetic characters

- Output accuracy improved due to multiple examples.

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

Zero-shot Prompt: Write a Python function to count vowels in a string.



```

File Edit Selection View Go Run ... ← → 🔍 AI ASSISTED CODING
EXPLORER ASSIGNMENTS ASSIGN-1-3.py ASSIGN-2-2.py ASSIGN-4-2.py DAY-1.2.py LAB-2.py LAB(2)\PRACTICE SESS...
ASSIGN-4-2.py •
# Zero-shot Prompt: Write a Python function to count vowels in a string.
def count_vowels(s):
    count = 0
    for ch in s:
        if ch in "aeiou":
            count += 1
    return count
  
```

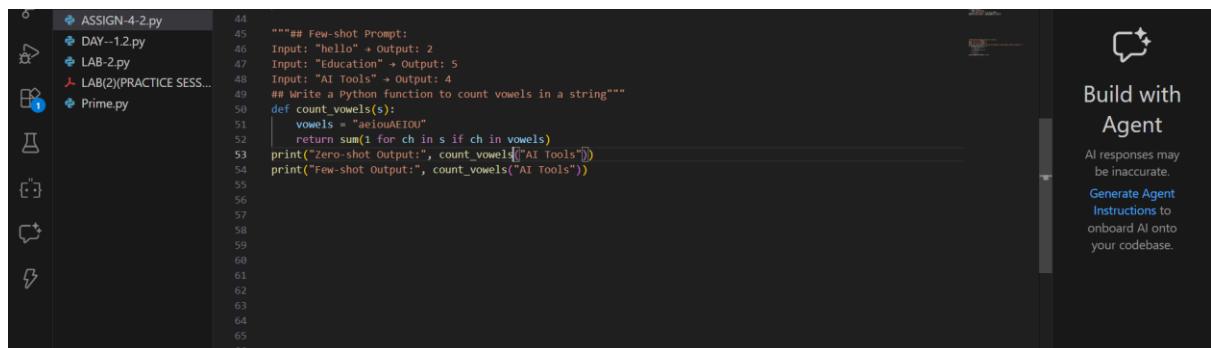
Few-shot Prompt: Write a Python function to count vowels in a string

Examples:

Input: "hello" → Output: 2

Input: "Education" → Output: 5

Input: "AI Tools" → Output: 4

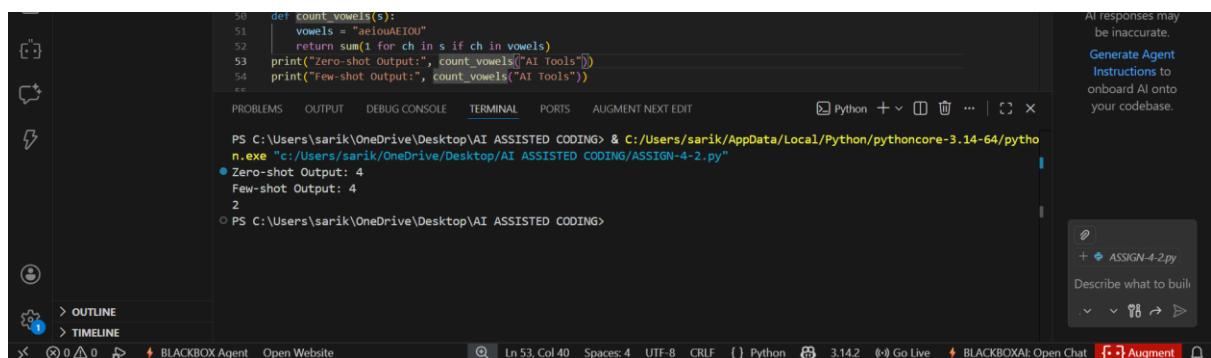


```

ASSIGN-4-2.py
44 ##### Few-shot Prompt:
45 Input: "hello" → Output: 2
46 Input: "Education" → Output: 5
47 Input: "AI Tools" → Output: 4
48 ## Write a Python function to count vowels in a string"""
49 def count_vowels(s):
50     vowels = "aeiouAEIOU"
51     return sum(1 for ch in s if ch in vowels)
52 print("zero-shot Output:", count_vowels("AI Tools"))
53 print("few-shot Output:", count_vowels("AI Tools"))
  
```

Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

OUTPUT:



```

50 def count_vowels(s):
51     vowels = "aeiouAEIOU"
52     return sum(1 for ch in s if ch in vowels)
53 print("zero-shot Output:", count_vowels("AI Tools"))
54 print("few-shot Output:", count_vowels("AI Tools"))
  
```

AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

```

PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-4-2.py"
● Zero-shot Output: 4
Few-shot Output: 4
2
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
  
```

+ ASSIGN-4-2.py
Describe what to build

Comparison Table:

Feature	Zero-shot	Few-shot
Case handling	Only lowercase	Upper & lowercase
Accuracy	Moderate	High
Robustness	Basic	Improved
Readability	Simple	Optimized

Explanation:

1. Few-shot prompting improved the output by providing examples that showed:
 - Upper and lowercase handling
 - Realistic input patterns

This helped the AI generate a more accurate and generalized solution.

Task Description-5: Few-shot Prompting (No min() function)

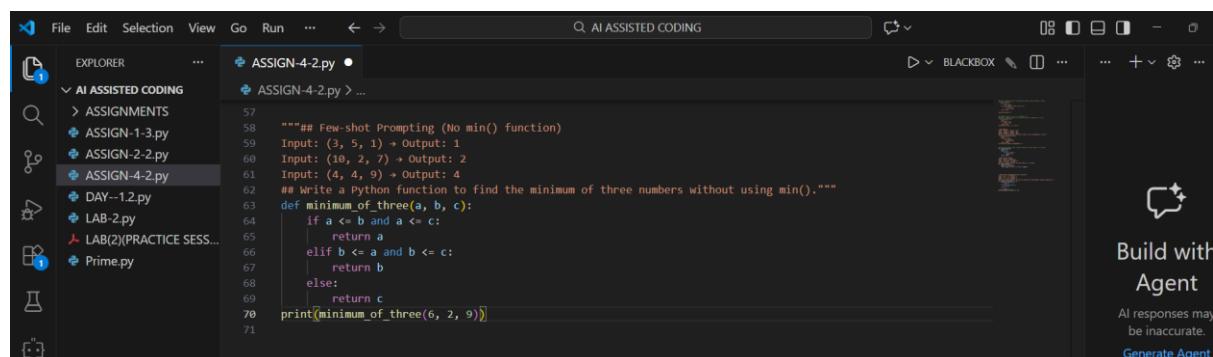
Prompt: Write a Python function to find the minimum of three numbers without using min().

Examples:

Input: (3, 5, 1) → Output: 1

Input: (10, 2, 7) → Output: 2

Input: (4, 4, 9) → Output: 4



```

File Edit Selection View Go Run ... ⏪ ⏩ Q AI ASSISTED CODING
EXPLORER ASSIGN-4-2.py ...
AI ASSISTED CODING > ASSIGNMENTS
ASSIGN-1-3.py
ASSIGN-2-2.py
ASSIGN-4-2.py
DAY-1.2.py
LAB-2.py
LAB(2)/PRACTICE SESS...
Prime.py

57 """# Few-shot Prompting (No min() function)
58 Input: (3, 5, 1) → Output: 1
59 Input: (10, 2, 7) → Output: 2
60 Input: (4, 4, 9) → Output: 4
61 ## Write a Python function to find the minimum of three numbers without using min().
62 def minimum_of_three(a, b, c):
63     if a <= b and a <= c:
64         return a
65     elif b <= a and b <= c:
66         return b
67     else:
68         return c
69
70 print(minimum_of_three(6, 2, 9))
71
    
```

Build with Agent
AI responses may be inaccurate.
Generate Agent

OUTPUT:

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with icons for file operations like Open, Save, and Undo. Below it are sections for OUTLINE and TIMELINE. The main area has a code editor window and a terminal window.

Code Editor (Left):

```
67     else:
68         return c
69     print(minimum_of_three(6, 2, 9))
70 
```

Terminal (Right):

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-4-2.py"
● 2
○ PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

Right Panel:

Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

ASSIGN-4-2.py...
Describe what to build

Explanation:

1. Few-shot examples guided logical comparisons.
2. Handles:

Equal values

All ordering cases

3. Does not use built-in min() as instructed.