

Amruth sagar Vemuganti

2403a51I44

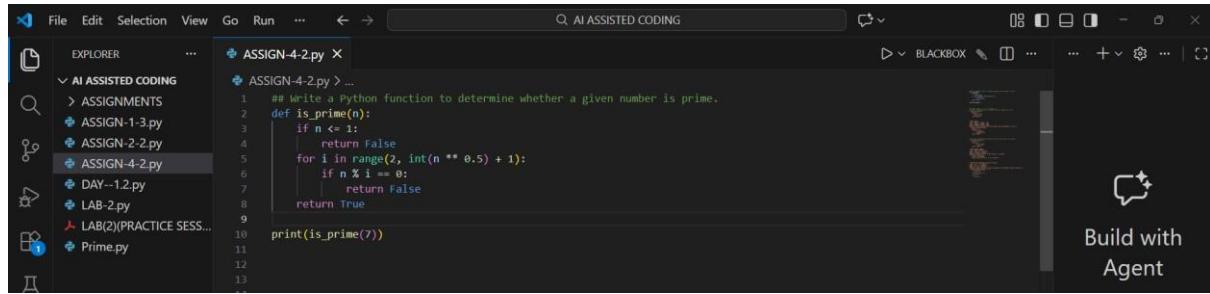
B-52

Lab 4

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

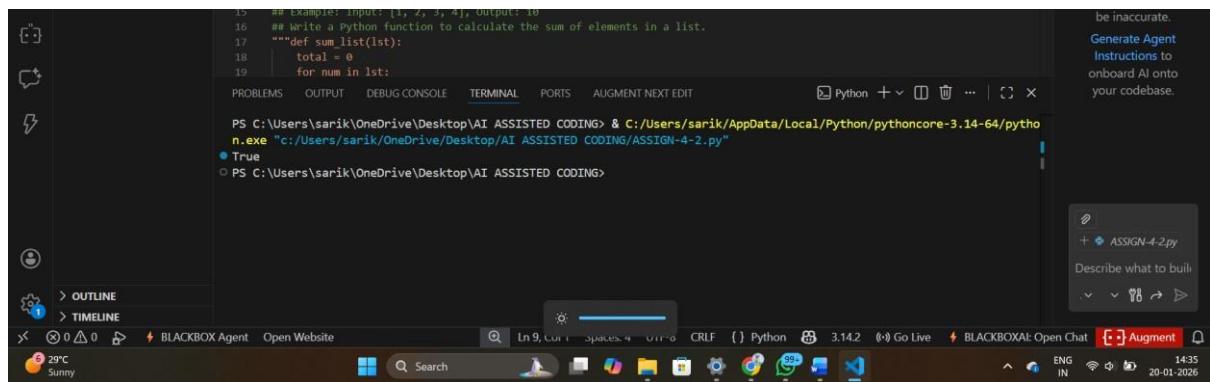
Task Description-1: Zero-shot Prompting

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



```
File Edit Selection View Go Run ... ← → Q AI ASSISTED CODING
EXPLORER ASSIGN-4-2.py ...
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
1 ## Write a Python function to determine whether a given number is prime.
2 def is_prime(n):
3     if n <= 1:
4         return False
5     for i in range(2, int(n ** 0.5) + 1):
6         if n % i == 0:
7             return False
8     return True
9
10 print(is_prime(7))
11
12
13
```

OUTPUT:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AUGMENT NEXT EDIT
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
n.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-4-2.py"
● True
○ PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING
● OUTLINE
● TIMELINE
BLACKBOX Agent Open Website
Search
Ln 9, Col 1 Spaces: 4 0170 CRLF { Python 3.14.2 ⓘ Go Live 🔥 BLACKBOXAI: Open Chat Augment
6 29°C Sunny ENG IN 14:35 20-01-2026
```

Explanation:

1. Zero-shot prompting provides only instructions, no examples.
2. The AI correctly implemented:

Prime definition logic

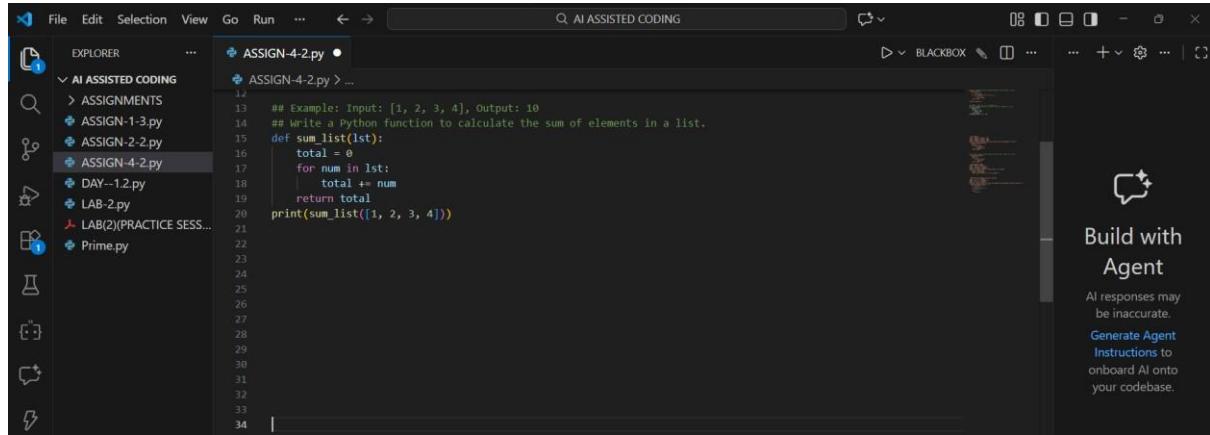
Square-root optimization

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

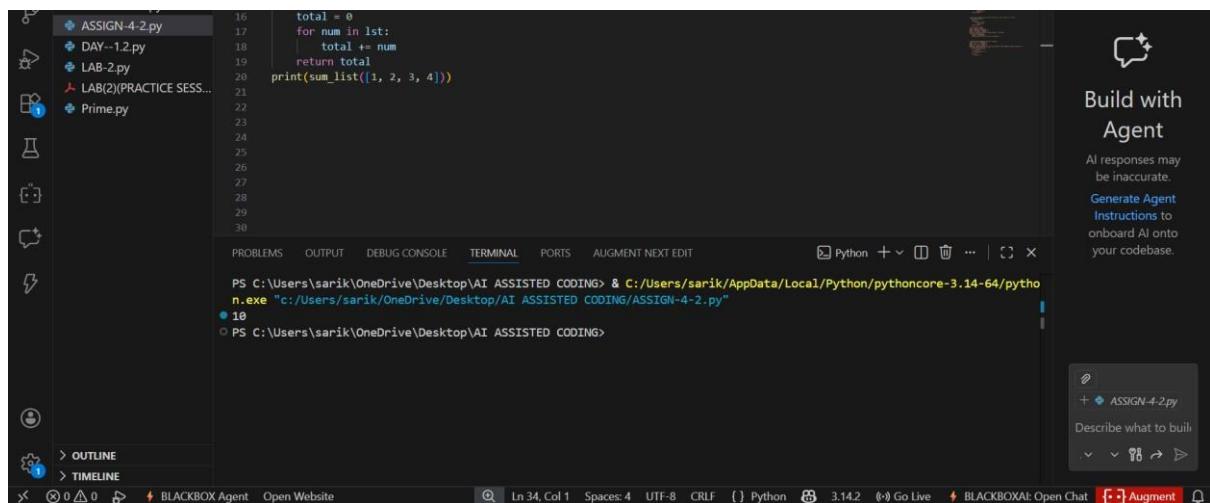


```
File Edit Selection View Go Run ... < > Q AI ASSISTED CODING 08 BLACKBOX ... + - x
EXPLORER ASSIGN-4-2.py ...
ASSIGNMENTS
  ASSIGN-1-3.py
  ASSIGN-2-2.py
  ASSIGN-4-2.py
  DAY--1.2.py
  LAB-2.py
LAB(2)(PRACTICE SESS...
  Prime.py

12
13  ## Example: Input: [1, 2, 3, 4], Output: 10
14  ## Write a Python function to calculate the sum of elements in a list.
15  def sum_list(lst):
16      total = 0
17      for num in lst:
18          total += num
19      return total
20  print(sum_list([1, 2, 3, 4]))
21
22
23
24
25
26
27
28
29
30
31
32
33
34
```

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

OUTPUT:



```
ASSIGN-4-2.py
DAY--1.2.py
LAB-2.py
LAB(2)(PRACTICE SESS...
Prime.py

16      total = 0
17      for num in lst:
18          total += num
19      return total
20  print(sum_list([1, 2, 3, 4]))
21
22
23
24
25
26
27
28
29
30

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AUGMENT NEXT EDIT Python + - x
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"
● 10
○ PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>

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

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 Visual Studio Code interface with the "AI ASSISTED CODING" extension active. The Explorer sidebar lists several Python files. The code editor displays a Python script named "ASSIGN-4-2.py". The script contains a function "extract_digits" that takes a string as input and returns a string containing only the digits from the input. The function uses a for loop to iterate through each character in the input string, checks if it is a digit using the "isdigit" method, and if so, adds it to a "digits" string. Finally, it prints the resulting "digits" string. The code editor has a dark theme, and the extension's UI elements are visible on the right side.

OUTPUT:

The screenshot shows the terminal window in VS Code. The user has run the command "python ASSIGN-4-2.py". The output of the script, which is "2026", is displayed in the terminal. The terminal also shows the full path to the Python executable and the current working directory. A tooltip on the right side of the terminal provides information about AI responses and generating agent instructions.

Explanation:

1. Few-shot prompting provides pattern recognition.

2. AI correctly:

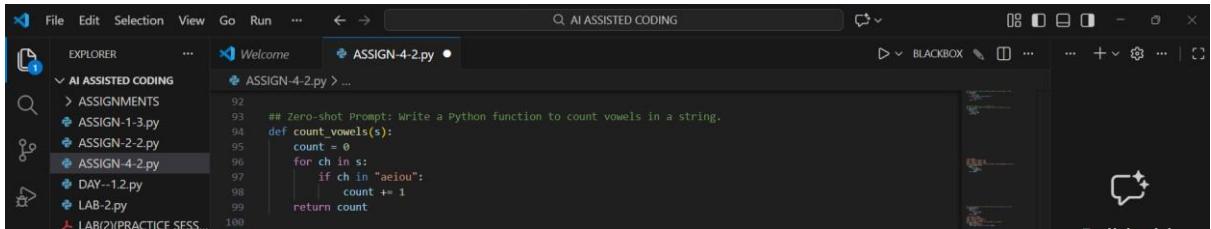
Identified digit extraction rule

Ignored alphabetic characters

3. 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.



A screenshot of the Visual Studio Code interface. The title bar says "AI ASSISTED CODING". The left sidebar shows a folder named "AI ASSISTED CODING" containing files like "ASSIGN-1-3.py", "ASSIGN-2-2.py", "ASSIGN-4-2.py", "DAY--1.2.py", and "LAB-2.py". The main editor area has the file "ASSIGN-4-2.py" open, which contains the following code:

```
92 ## zero-shot Prompt: Write a Python function to count vowels in a string,
93 def count_vowels(s):
94     count = 0
95     for ch in s:
96         if ch in "aeiou":
97             count += 1
98     return count
99
```

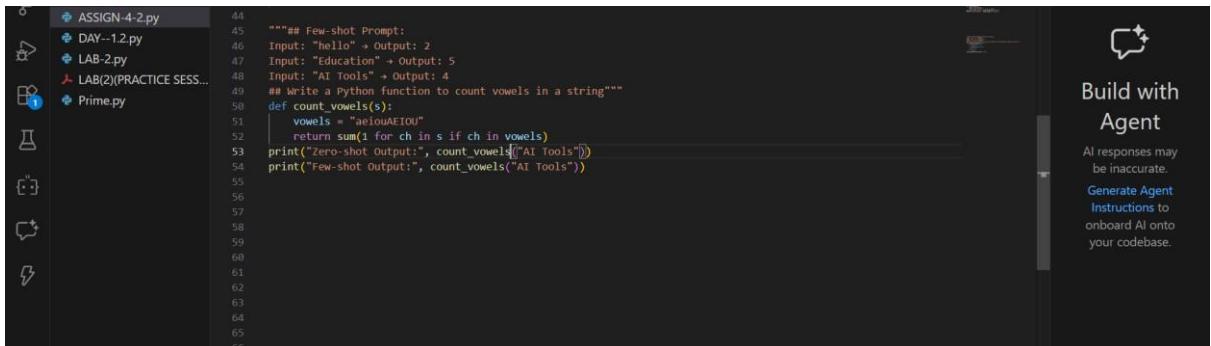
Few-shot Prompt: Write a Python function to count vowels in a string

Examples:

Input: "hello" → Output: 2

Input: "Education" → Output: 5

Input: "AI Tools" → Output: 4



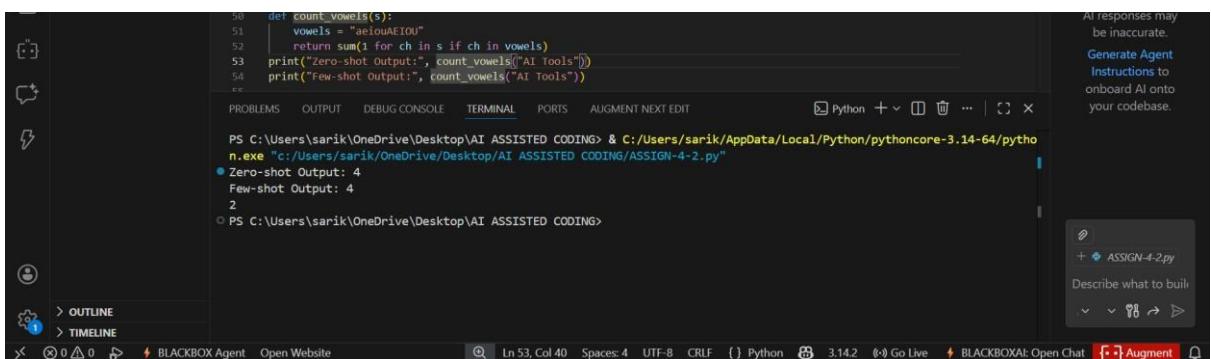
A screenshot of the Visual Studio Code interface. The title bar says "AI ASSISTED CODING". The left sidebar shows a folder named "AI ASSISTED CODING" containing files like "ASSIGN-4-2.py", "DAY--1.2.py", "LAB-2.py", and "Prime.py". The main editor area has the file "ASSIGN-4-2.py" open, which contains the following code:

```
44 """# Few-shot Prompt:
45 Input: "hello" + Output: 2
46 Input: "Education" + Output: 5
47 Input: "AI Tools" + Output: 4
48
49 # Write a Python function to count vowels in a string"""
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"))
55
56
57
58
59
60
61
62
63
64
65
```

The right side of the screen shows a "Build with Agent" panel with the following text:

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

OUTPUT:



A screenshot of the Visual Studio Code interface. The title bar says "AI ASSISTED CODING". The left sidebar shows a folder named "AI ASSISTED CODING" containing files like "ASSIGN-4-2.py", "DAY--1.2.py", "LAB-2.py", and "Prime.py". The main editor area has the file "ASSIGN-4-2.py" open, and the terminal tab is active. The terminal shows the following output:

```
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>
```

The right side of the screen shows a "Build with Agent" panel with the following text:

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

Comparison Table:

Feature	Zero-shot	Few-shot
Input	"hello"	"hello", "Education", "AI Tools"
Output	2	2, 5, 4
Prompt Type	General	Specific

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)

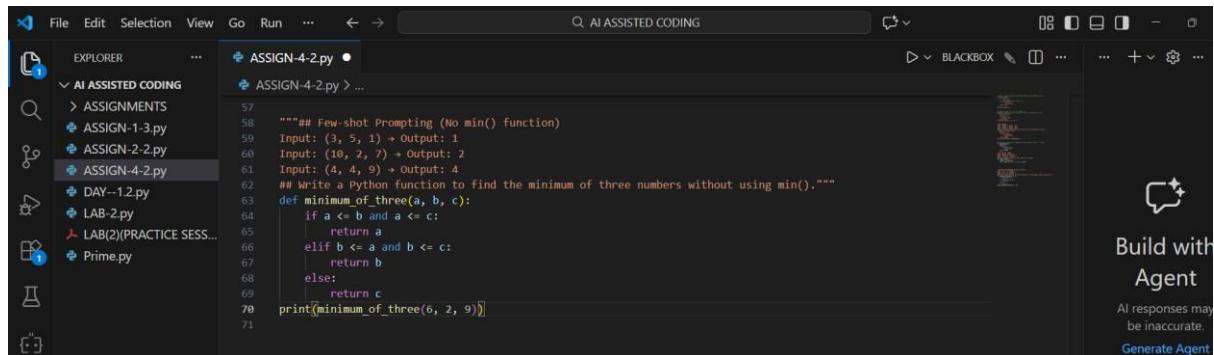
Prompt: Write a Python function to find the minimum of three numbers without using min().

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-12.py
LAB(2).PRACTICE SESS...
Prime.py
BLACKBOX 🌐 ...
... + ⚙ ...
57
58     """# Few-shot Prompting (No min() function)
59     Input: (3, 5, 1) + Output: 1
60     Input: (10, 2, 7) + Output: 2
61     Input: (4, 4, 9) + Output: 4
62     ## Write a Python function to find the minimum of three numbers without using min()."""
63     def minimum_of_three(a, b, c):
64         if a <= b and a <= c:
65             return a
66         elif b <= a and b <= c:
67             return b
68         else:
69             return c
70     print(minimum_of_three(6, 2, 9))
71
    
```

Build with Agent
AI responses may be inaccurate.
Generate Agent

OUTPUT:

A screenshot of a code editor interface. On the left, there's a sidebar with icons for file operations like new, open, save, and delete. Below that are sections for 'OUTLINE' and 'TIMELINE'. The main area shows a Python script:

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

The terminal tab at the top shows the command line output:

```
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>
```

At the bottom, there are various status indicators and toolbars. On the right, there's an 'Agent' panel with the following text:

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

Explanation:

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

Equal values

All ordering cases

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