

Harika Jupaka

2403a51l31

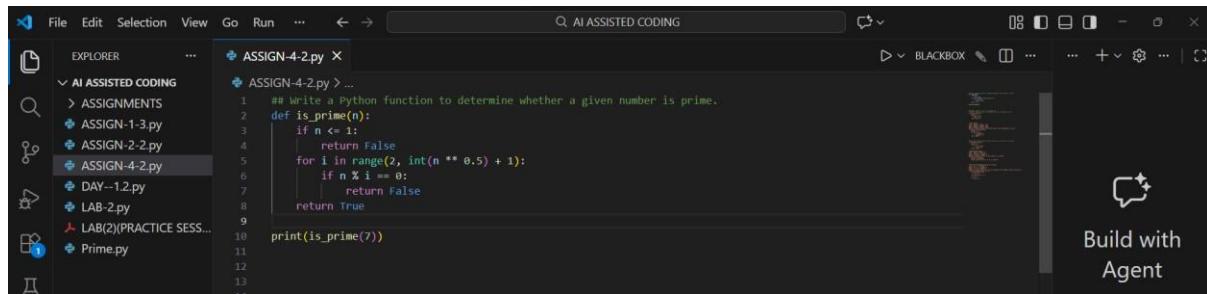
B-52

Lab 4

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

Task Description-1: Zero-shot Prompting

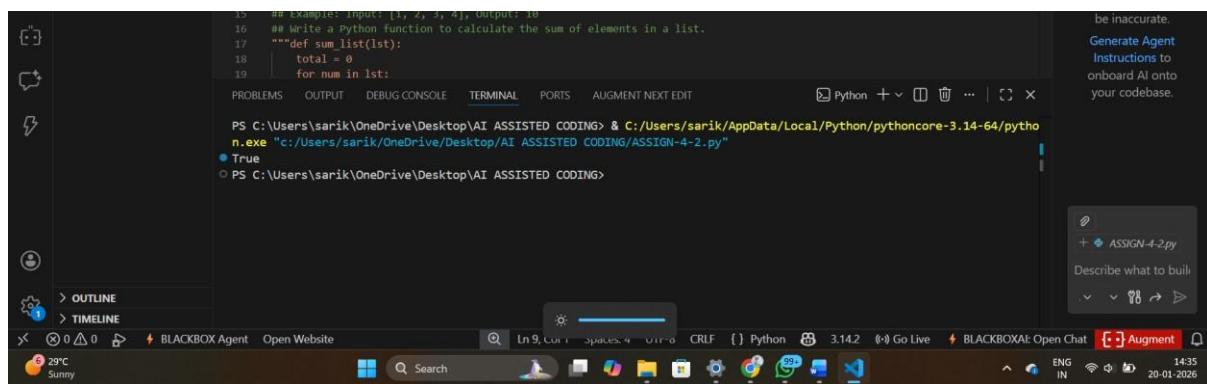
Prompt: Write a Python function to determine whether a given number is prime.



```
File Edit Selection View Go Run ... ← → Q AI ASSISTED CODING 08 □ - o 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
Build with Agent
```

```
ASSIGN-4-2.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))
```

OUTPUT:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AUGMENT NEXT EDIT Python + v w ... | x
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
```

Explanation:

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

Prime definition logic

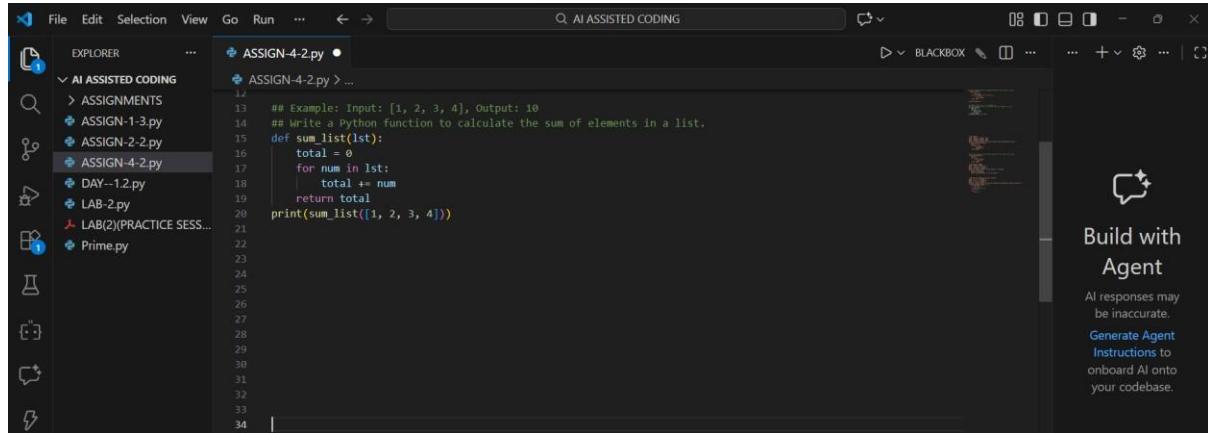
Square-root optimization

3. Demonstrates that simple logical problems work well with zero-shot prompts.

Task Description-2: One-shot Prompting

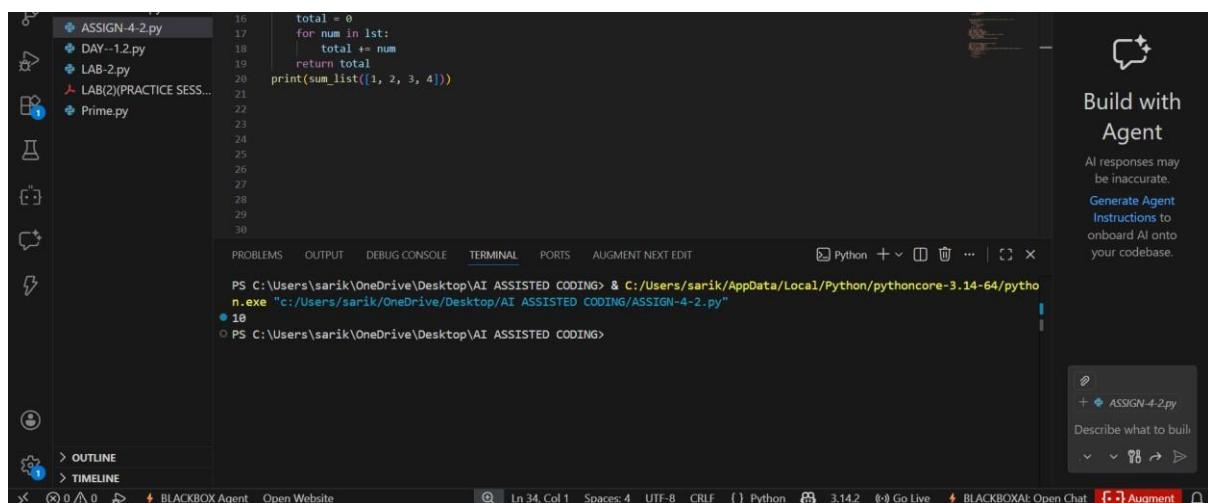
Prompt: Write a Python function to calculate the sum of elements in a list.

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



```
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]))
```

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/Desktop/AI ASSISTED CODING/ASSIGN-4-2.py"
● 10
○ PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

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"

OUTPUT:

A screenshot of the Visual Studio Code interface. The terminal tab is active, showing command-line output for a Python script named 'ASSISTED CODING'. The code in the editor is as follows:

```
34 print(extract_digits("1234567890"))
35
36
37
38
39
40
```

The terminal output shows the script being run and its execution:

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

The status bar at the bottom indicates the file is 'BLACKBOX Agent' and shows the current file path as 'C:/Users/srarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-4-2.py'. The status bar also shows the Python version as 3.14.2 and the file size as 3.14.2 KB.

Explanation:

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

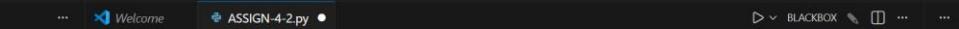
Identified digit extraction rule

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.



The screenshot shows the Visual Studio Code interface with the "AI ASSISTED CODING" extension active. The top bar has tabs for "File", "Edit", "Selection", "View", "Go", "Run", and "AI ASSISTED CODING". The left sidebar has sections for "EXPLORER", "AI ASSISTED CODING", "ASSIGNMENTS", and "LABS". The main editor area shows a Python file named "ASSIGN-4-2.py" with the following code:

```
# 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

The screenshot shows a Jupyter Notebook interface with several files listed in the sidebar: ASSIGN-4-2.py, DAY-1.2.py, LAB-2.py, LAB(2)(PRACTICE SESSION).py, and Prime.py. The main area displays the following Python code:

```
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"))  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66
```

On the right side of the interface, there is a "Build with Agent" button and a note stating "Al responses may be inaccurate." Below that, there are links to "Generate Agent Instructions" and "onboard AI onto your codebase".

OUTPUT:

The screenshot shows a Python script named `ASSIGN-4-2.py` in the code editor. The script defines a function `count_vowels` that takes a string `s` and returns the count of vowels in it. It uses a constant `vowels` containing the string "aeiouAEIOU". The script then prints two outputs: "Zero-shot Output:" followed by the result of calling `count_vowels("AI Tools")`, and "Few-shot Output:" followed by the result of calling `count_vowels("AI Tool$")`.

```
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 Tool$"))
55
```

The terminal window shows the execution of the script. It first runs `python core-3.14-64/python n.exe "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"`. The output is "Zero-shot Output: 4" and "Few-shot Output: 4". A second run of the command shows "2" as the output.

```
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"
● Zero-shot Output: 4
Few-shot Output: 4
2
○ PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

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

```
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))
```

OUTPUT:

The screenshot shows a Python code editor interface. On the left, there's a sidebar with icons for file operations like Open, Save, and Undo. Below it are buttons for 'OUTLINE' and 'TIMELINE'. The main area displays the following Python code:

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

Below the code, the terminal window shows the command being run:

```
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>
```

The status bar at the bottom provides information about the file: Ln 43, Col 17 (122 selected), Spaces: 4, UTF-8, CRLF, Python 3.14.2, Go Live, BLACKBOX Agent, Open Chat, and Augment.

Explanation:

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

Equal values

All ordering cases

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