

ASSIGNMENT 4.2

Keerthana Erukala

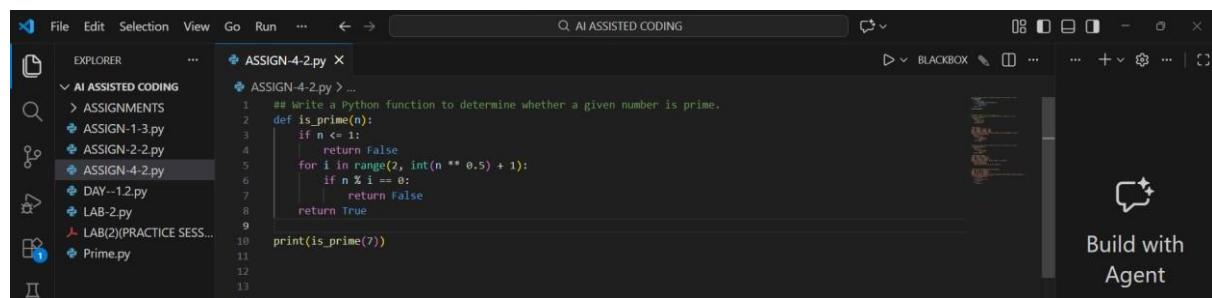
2403A51L06

B-51

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

Task Description-1: Zero-shot Prompting

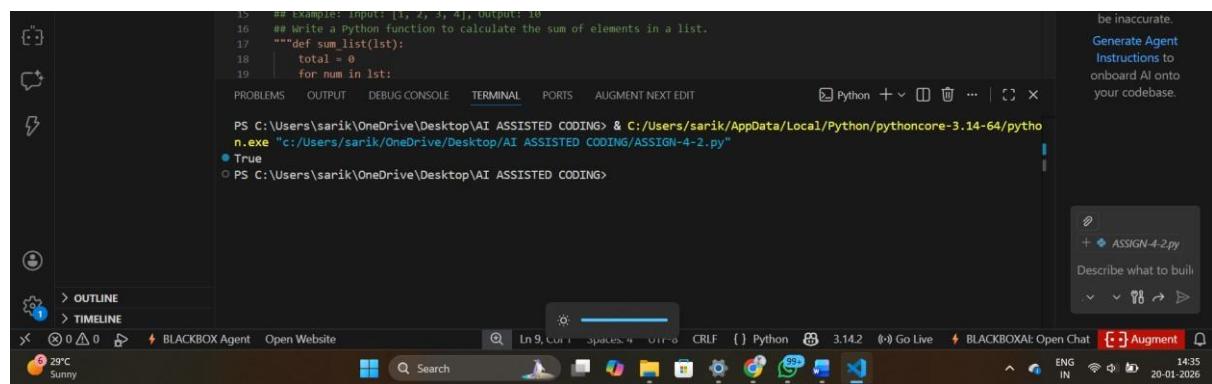
Prompt: Write a Python function to determine whether a given number is prime.



A screenshot of the Visual Studio Code interface. The left sidebar shows a file tree with several Python files like 'ASSIGN-1-3.py', 'ASSIGN-2-2.py', and 'ASSIGN-4-2.py'. The main editor area contains the following Python code:

```
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:



A screenshot of a terminal window within VS Code. The command 'python ASSIGN-4-2.py' is run, and the output is 'True'. The terminal also shows other commands related to AI-assisted coding.

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

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"
● 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

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The title bar reads "Q. AI ASSISTED CODING". The left sidebar has icons for Explorer, Search, Find, Open, and Prime. The "EXPLORER" view shows a tree structure with "AI ASSISTED CODING" expanded, listing files like "ASSIGN-1-3.py", "ASSIGN-2-2.py", and "ASSIGN-4-2.py". The main editor area displays Python code for calculating the sum of elements in a list:

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

A floating callout bubble on the right says "Build with Agent". Below it, a note states: "AI responses may be inaccurate. 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: ASSIGN-4-2.py, DAY-12.py, LAB-2.py, LAB(2)(PRACTICE SESSION), and Prime.py.
- Code Editor:** Displays the content of `ASSIGN-4-2.py`. The code defines a function `sum_list` that takes a list of numbers and returns their sum. It also contains a `Prime.py` file.
- Terminal:** Shows the command `python "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-4-2.py"` being run, resulting in the output `10`.

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 right side of the interface has a sidebar titled "Build with Agent" with a message: "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase."

OUTPUT:

The right side of the terminal has a sidebar with the same message: "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase."

Explanation:

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

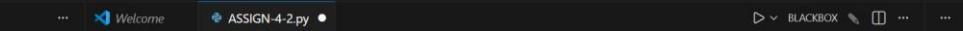
Identified digit extraction rule

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 shows the "EXPLORER" view with a tree structure: "ASSIGNMENTS" (selected), "ASSIGN-1-3.py", "ASSIGN-2-2.py", "ASSIGN-4-2.py" (selected), "DAY-12.py", "LAB-2.py", and "LAR2DIPRACTICE SESS.". The main editor area displays Python code for counting vowels in a string, with the first few lines highlighted in blue. A status bar at the bottom indicates "92 / 100". On the right, there's a "BLACKBOX" panel and a "COMMAND PREFERENCES" icon.

```
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
100
```

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 open files on the left sidebar. The main area displays Python code for vowel counting. The code includes a few-shot prompt, input-output pairs, and a function definition.

```
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
```

OUTPUT:

The screenshot shows a Python script named `count_vowels.py` in a code editor. The script defines a function `count_vowels(s)` that counts the number of vowels in a given string `s`. It uses a constant `vowels = "aeiouAEIOU"` and a list comprehension to sum the occurrences of each vowel. The script then prints two outputs: "Zero-shot Output:" followed by the count for the string "AI Tools", and "Few-shot Output:" followed by the count for the string "AI Tools".

```
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
```

The terminal below the editor shows the script being run and its output. The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and AUGMENT NEXT EDIT. The OUTPUT tab is active, showing the command `python count_vowels.py` and the resulting 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

```

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
LAB2(PRACTICE SESSION)
Prime.py
ASSIGN-4-2.py ...
ASSIGN-4-2.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))
    
```

OUTPUT:

The screenshot shows a code editor interface with the following details:

- Code Area:** Displays Python code:

```
67     else:
68         return c
69     print(minimum_of_three(6, 2, 9))
70
```
- Terminal:** Shows a terminal window with the command "python ASSIGN-4-2.py" and its 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>
```
- Bottom Status Bar:** Includes file navigation (OUTLINE, TIMELINE), document status (Ln 43, Col 17 (122 selected)), and system information (Spaces: 4, UTF-8, CRLF, Python 3.14.2, Go Live).
- Right Sidebar:** Titled "Agent", it contains a note about AI responses being inaccurate, a "Generate Agent Instructions" button, and a "Describe what to build" input field with the placeholder "ASSIGN-4-2.py...".

Explanation:

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

Equal values

All ordering cases

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