

## ASSIGNMENT 4.2

Keerthana Erukala

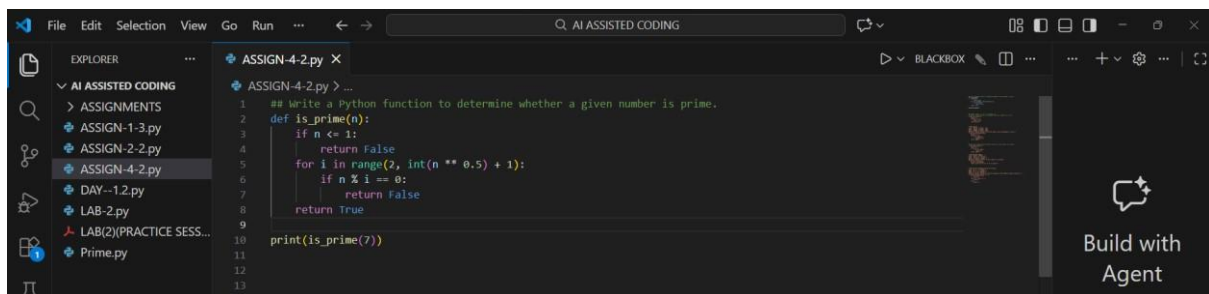
2403A51L06

B-51

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

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

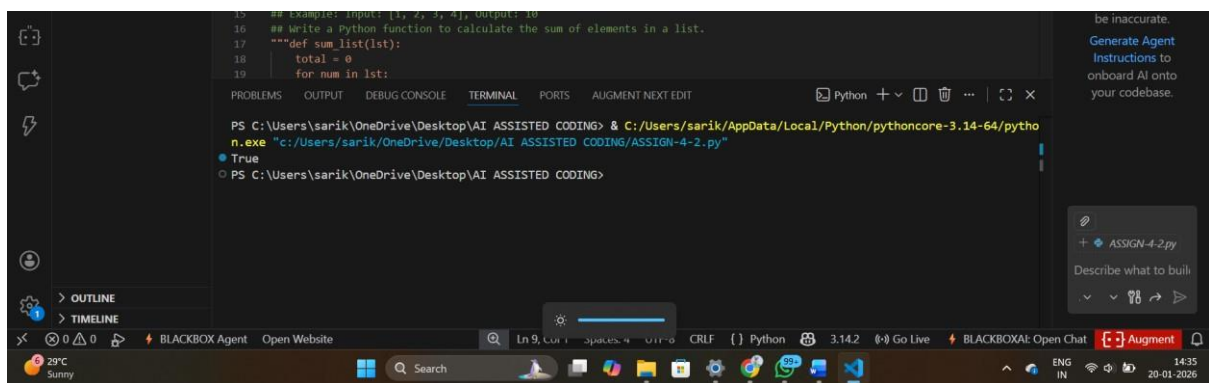
**Prompt:** Write a Python function to determine whether a given number is prime.



The screenshot shows a code editor with a file explorer on the left. The file explorer lists several files: 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. The main editor window shows 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))
11
12
13
```

**OUTPUT:**



The screenshot shows a terminal window with the following 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"
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

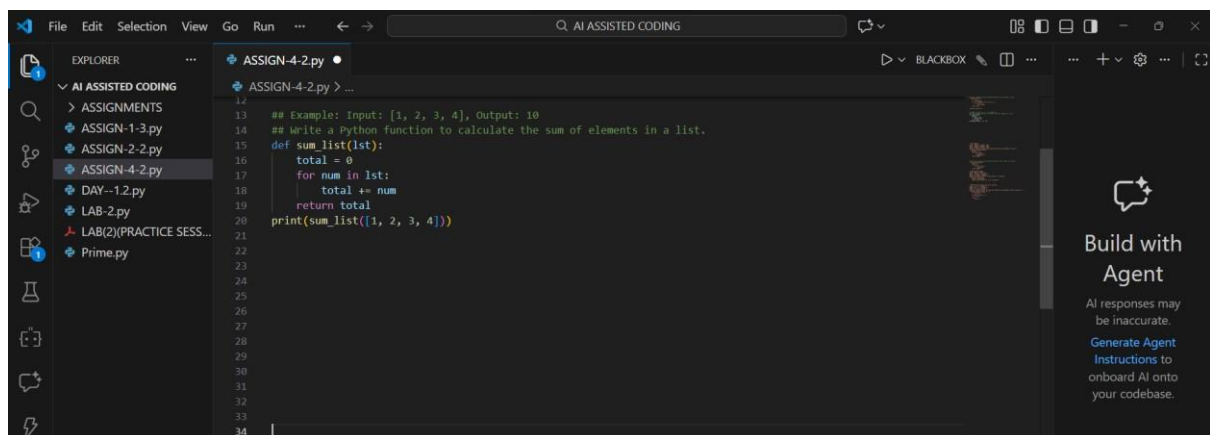
Square-root optimization

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.

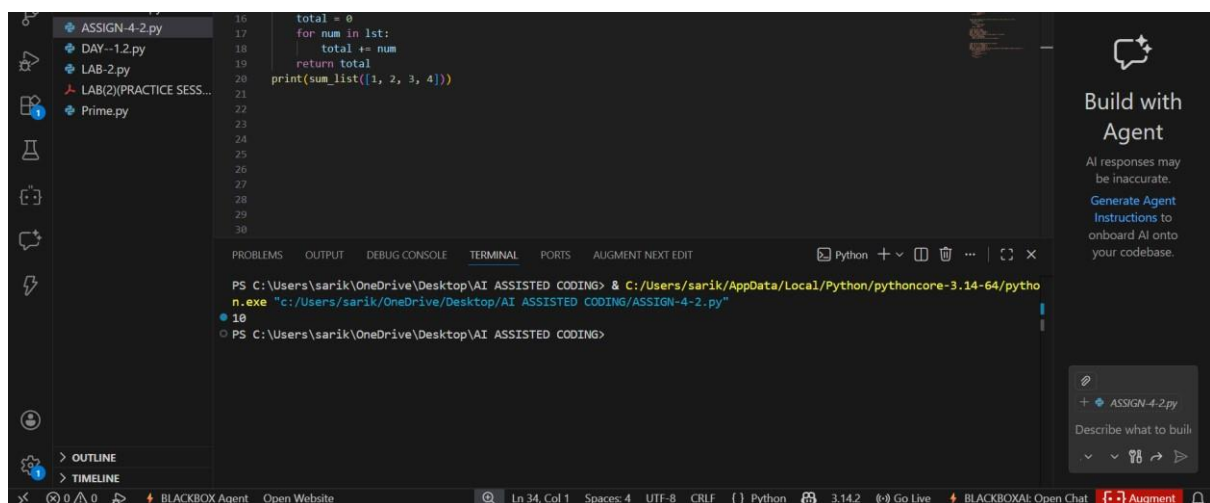
Example: Input: [1, 2, 3, 4], Output: 10



The screenshot shows the Visual Studio Code editor with a file named 'ASSIGN-4-2.py'. The code defines a function 'sum\_list' that takes a list 'lst' as input and returns the sum of its elements. The function uses a loop to iterate over each element in the list, adding it to a 'total' variable. The example input [1, 2, 3, 4] is used to test the function, and the output 10 is printed.

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

## OUTPUT:



The screenshot shows the Visual Studio Code editor with the same file 'ASSIGN-4-2.py'. The terminal window at the bottom shows the command to run the Python script, and the output is 10.

```
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>
```

## 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

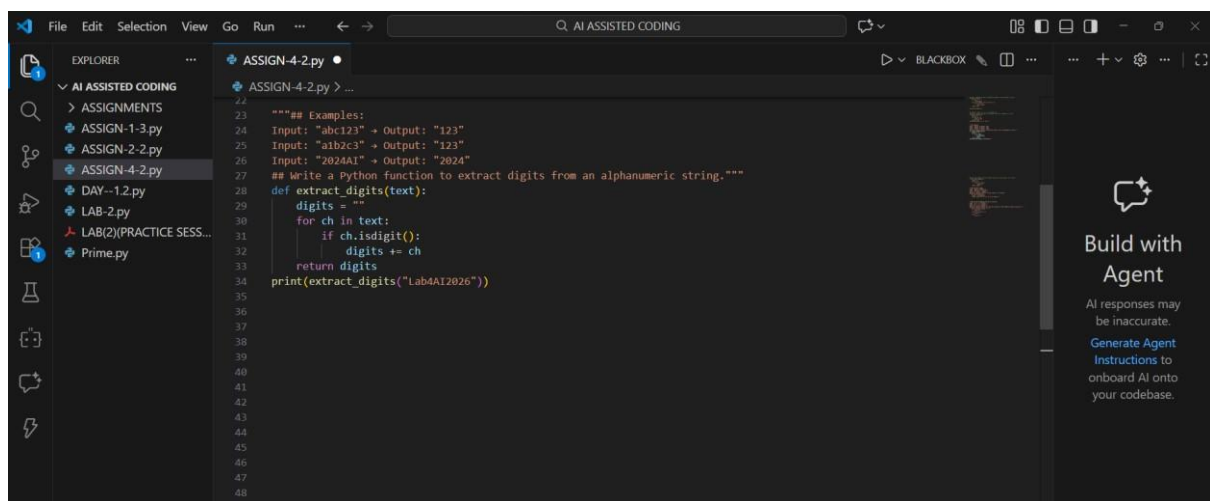
**Prompt:** Write a Python function to extract digits from an alphanumeric string.

Examples:

Input: "abc123" → Output: "123"

Input: "a1b2c3" → Output: "123"

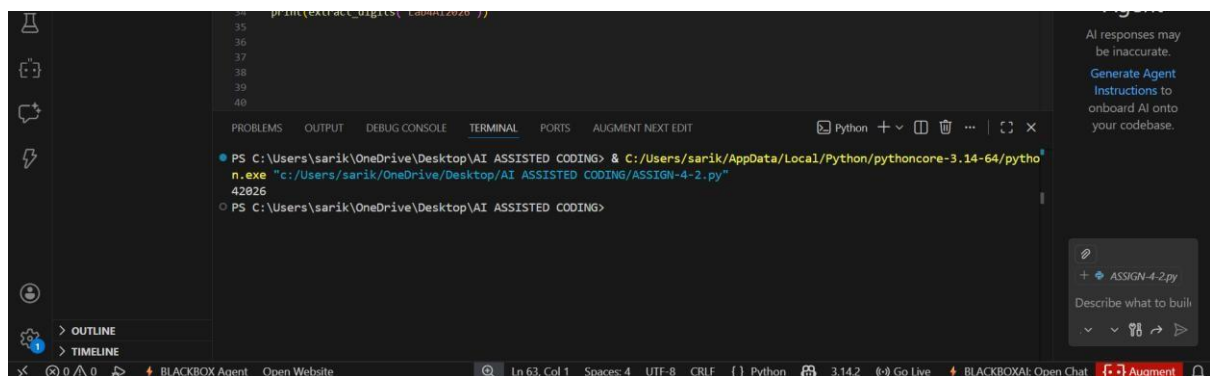
Input: "2024AI" → Output: "2024"



The screenshot shows the Visual Studio Code editor with a file named 'ASSIGN-4-2.py' open. The code defines a function 'extract\_digits(text)' that iterates through each character in the input string. If the character is a digit (checked using 'ch.isdigit()'), it is added to a string 'digits'. Finally, the function returns the 'digits' string. The code is as follows:

```
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"))
```

**OUTPUT:**



The screenshot shows the Visual Studio Code terminal window. The command executed is 'python c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-4-2.py'. The output of the program is '42026', which is the digits extracted from the input string 'Lab4AI2026'.

**Explanation:**

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

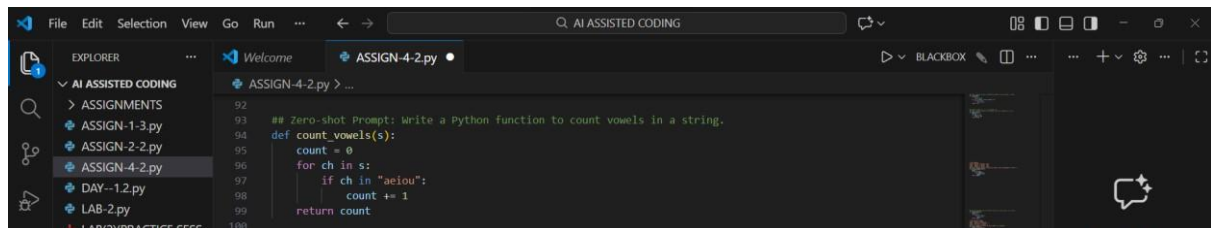
Identified digit extraction rule

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 a code editor with a file explorer on the left. The active file is 'ASSIGN-4-2.py'. The code in the editor is a Python function named 'count\_vowels(s)' that counts the number of vowels in a string 's'. The function uses a loop to iterate over each character in the string and increments a counter if the character is a vowel (a, e, i, o, u). The function returns the counter.

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

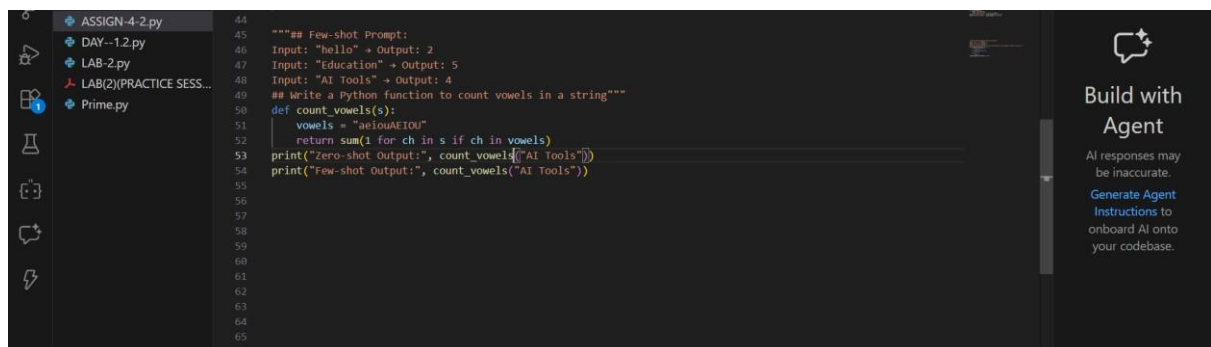
**Few-shot Prompt:** Write a Python function to count vowels in a string

Examples:

Input: "hello" → Output: 2

Input: "Education" → Output: 5

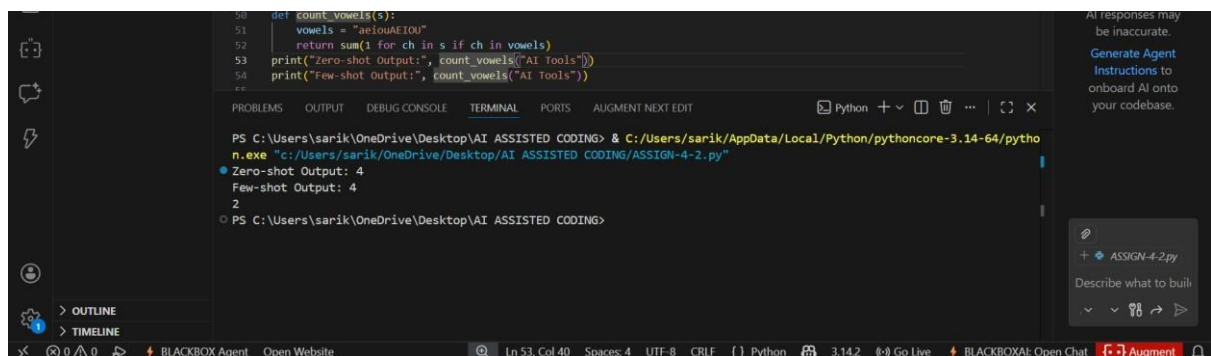
Input: "AI Tools" → Output: 4



The screenshot shows a code editor with a file explorer on the left. The active file is 'ASSIGN-4-2.py'. The code in the editor is a Python function named 'count\_vowels(s)' that counts the number of vowels in a string 's'. The function uses a loop to iterate over each character in the string and increments a counter if the character is a vowel (a, e, i, o, u). The function returns the counter. The code also includes a few-shot prompt example.

```
44  
45 """## Few-shot Prompt:  
46 Input: "hello" → Output: 2  
47 Input: "Education" → Output: 5  
48 Input: "AI Tools" → Output: 4  
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
```

**OUTPUT:**



The screenshot shows a terminal window with the output of the Python function. The output shows the zero-shot output as 4 and the few-shot output as 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"  
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

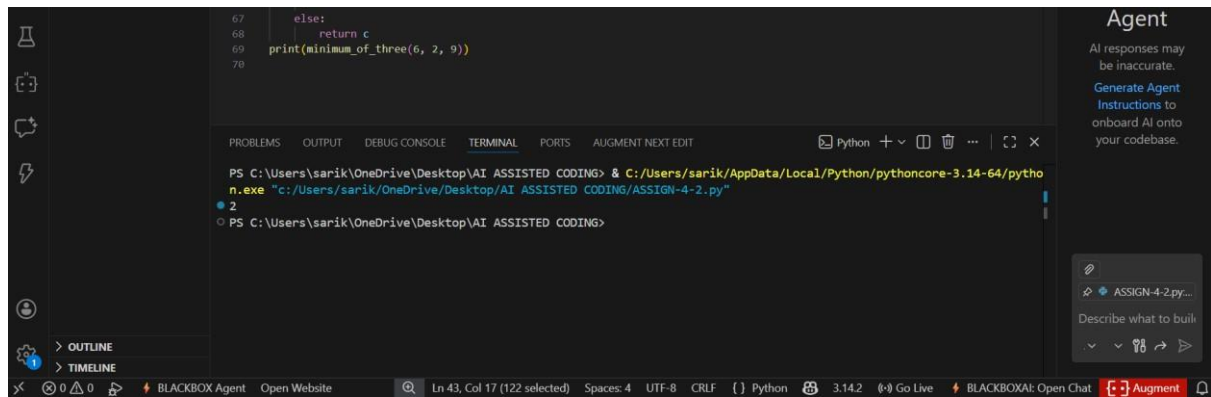
Input: (4, 4, 9) → Output: 4

```

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))
71

```

**OUTPUT:**



## Explanation:

1. Few-shot examples guided logical comparisons.
2. Handles:
  - Equal values
  - All ordering cases
3. Does not use built-in min() as instructed.