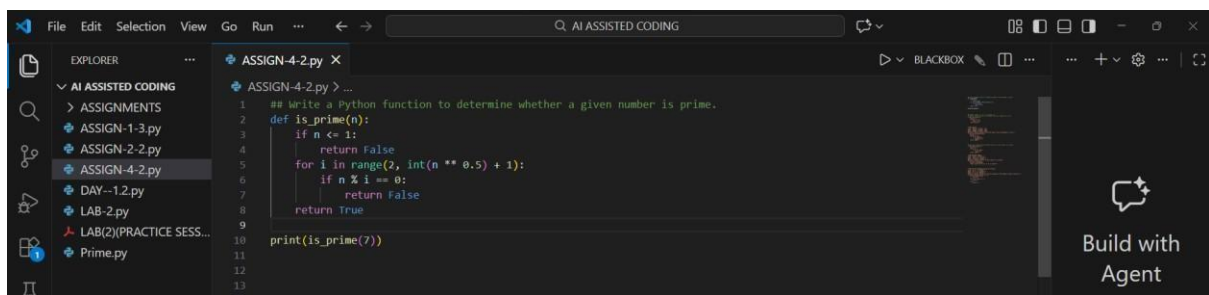


Lab 4

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

Task Description-1: Zero-shot Prompting

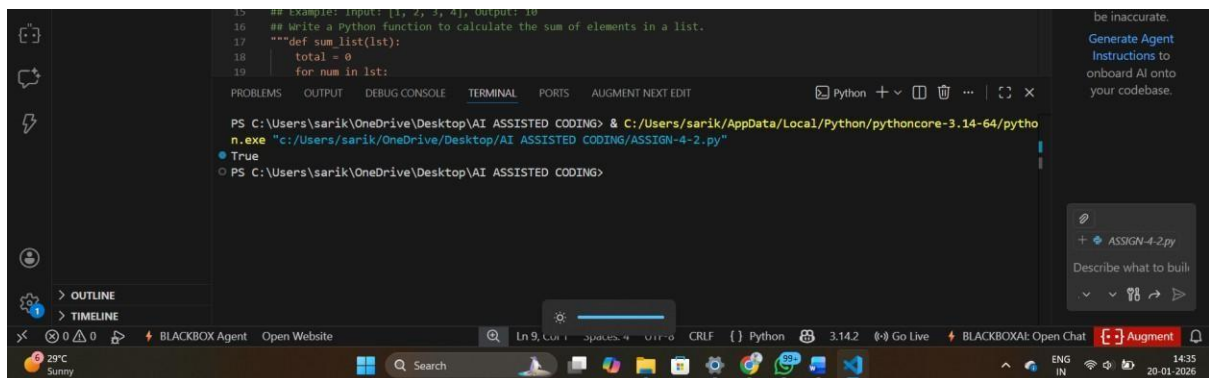
Prompt: Write a Python function to determine whether a given number is prime.



The screenshot shows a code editor with a file explorer on the left. The file explorer lists several files under 'AI ASSISTED CODING', including 'ASSIGN-1-3.py', 'ASSIGN-2-2.py', 'ASSIGN-4-2.py', 'DAY-12.py', 'LAB-2.py', 'LAB(2)(PRACTICE SESS...', and 'Prime.py'. The main editor window displays the code for 'ASSIGN-4-2.py'. The code is a Python function named 'is_prime(n)' that checks if a number 'n' is prime. It uses a square-root optimization by iterating from 2 to the square root of 'n'. The function returns 'True' if the number is prime and 'False' otherwise. A test call 'print(is_prime(7))' is shown at the bottom of the code block.

```
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 command prompt 'PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>'. The command executed is 'python c:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-4-2.py"'. The output of the command is 'True'. The terminal window also shows the command prompt 'PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>'.

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> python 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>
```

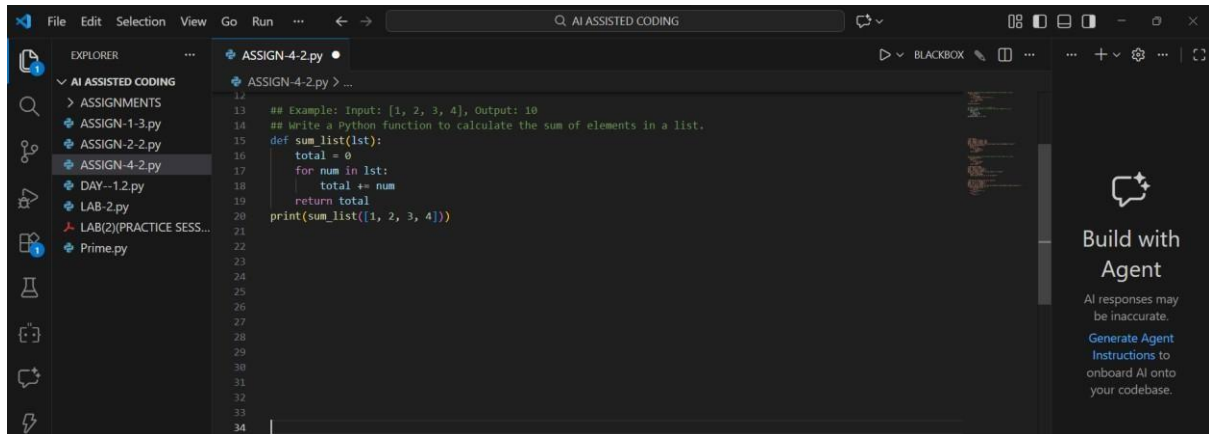
Explanation:

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

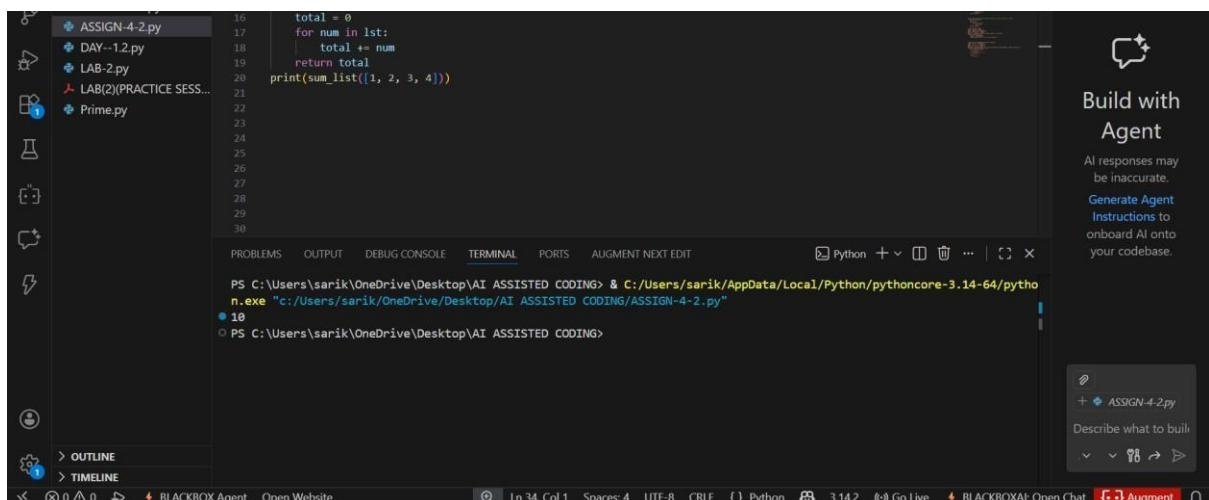
Prompt: Write a Python function to calculate the sum of elements in a list.

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



```
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
```

OUTPUT:



```
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
```

```
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"
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

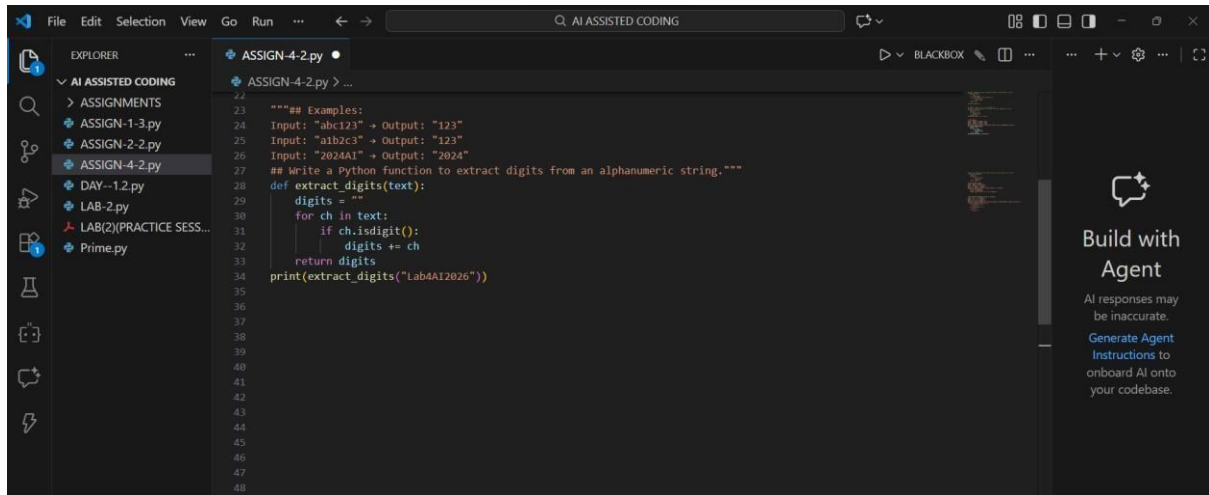
Prompt: Write a Python function to extract digits from an alphanumeric string.

Examples:

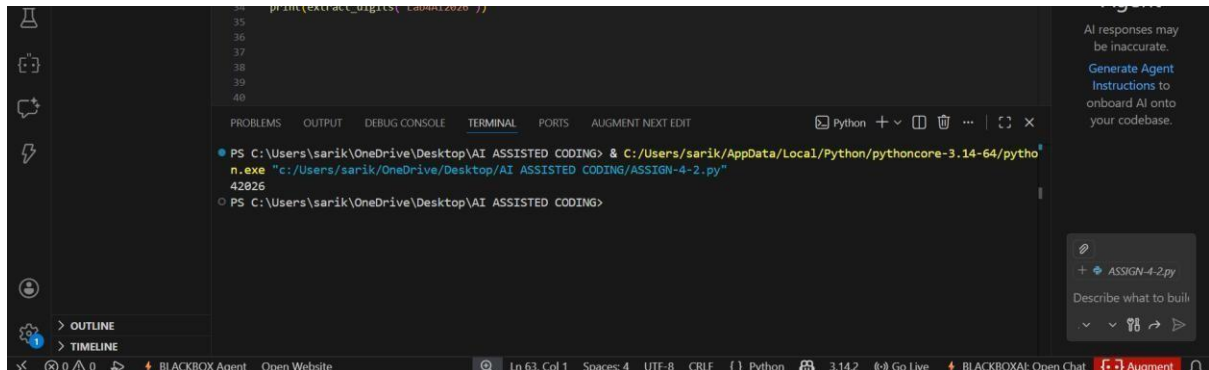
Input: "abc123" → Output: "123"

Input: "a1b2c3" → Output: "123"

Input: "2024AI" → Output: "2024"



OUTPUT:

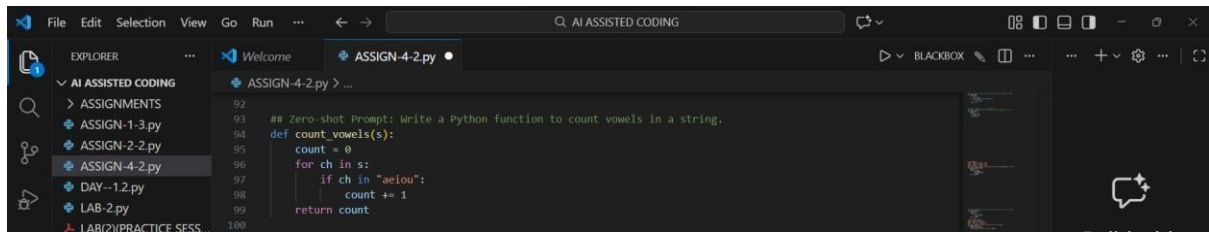


Explanation:

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.



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

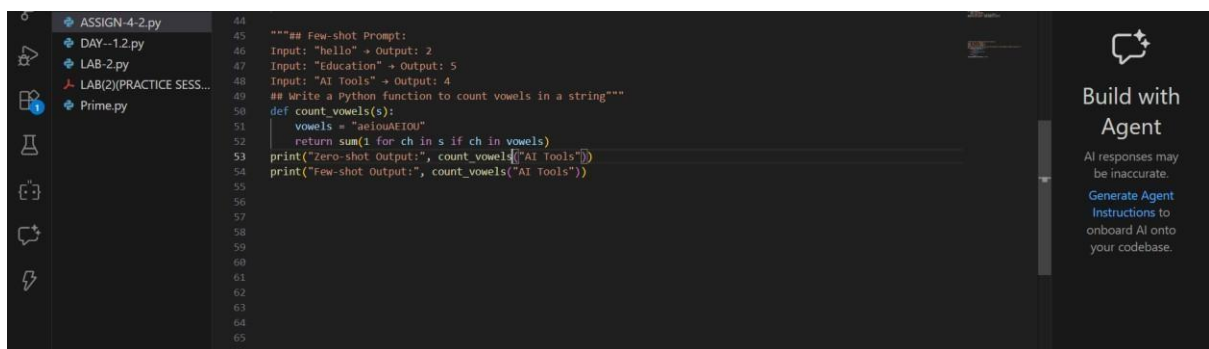
Few-shot Prompt: Write a Python function to count vowels in a string

Examples:

Input: "hello" → Output: 2

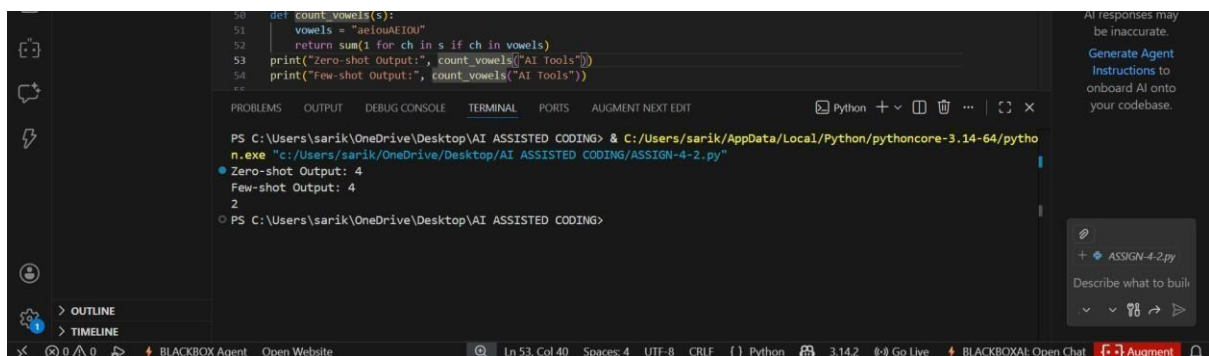
Input: "Education" → Output: 5

Input: "AI Tools" → Output: 4



```
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
66
```

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)

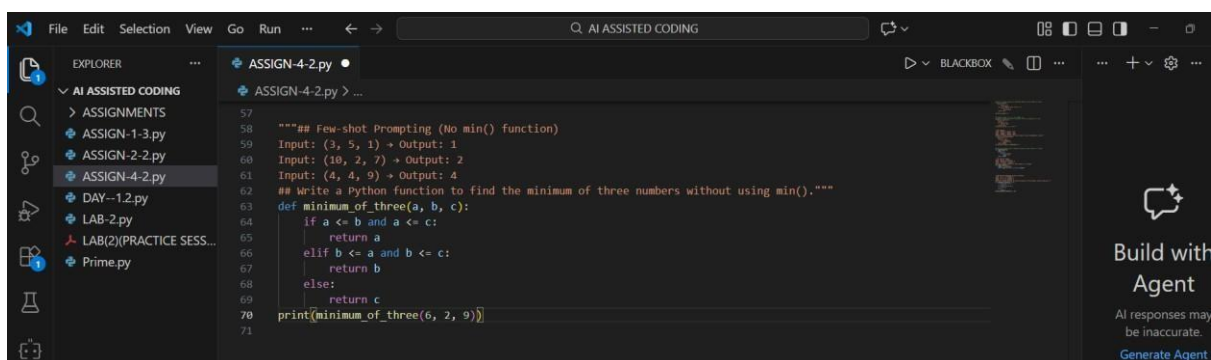
Prompt: Write a Python function to find the minimum of three numbers without using min().

Examples:

Input: (3, 5, 1) → Output: 1

Input: (10, 2, 7) → Output: 2

Input: (4, 4, 9) → Output: 4

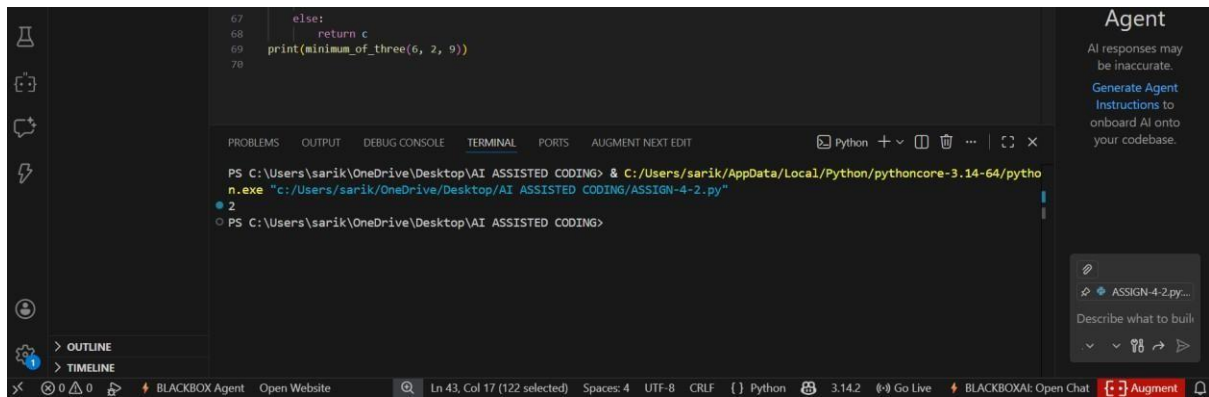


```

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))
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.