N.Sai Akash

2403a51l57

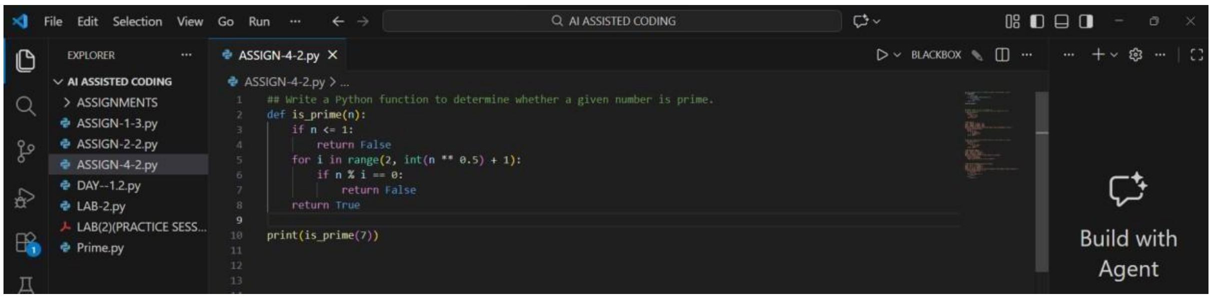
B-52

Lab 4

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

Task Description-1: Zero-shot Prompting

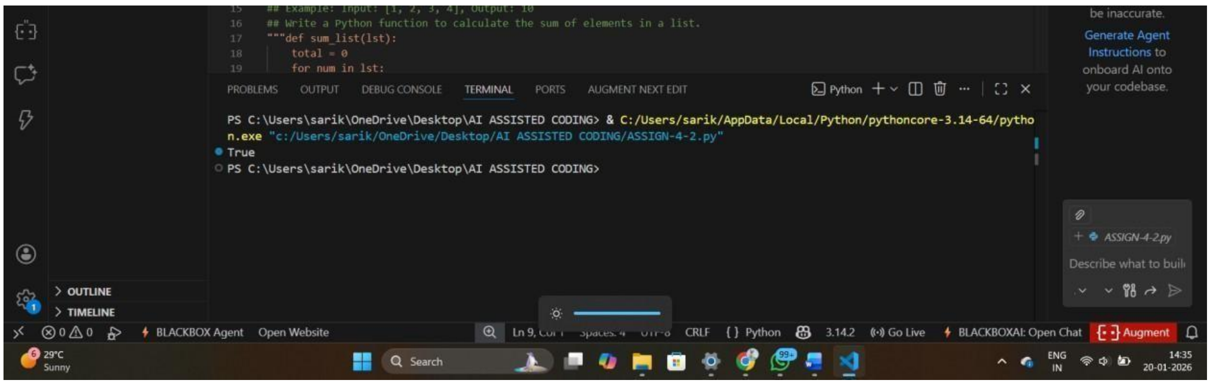
Prompt: Write a Python function to determine whether a given number is prime.



```python
## Write a Python function to determine whether a given number is prime.
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True

print(is_prime(7))
```

OUTPUT:



```python
## Example: Input: [1, 2, 3, 4], Output: 10
## Write a Python function to calculate the sum of elements in a list.
"""def sum_list(lst):
    total = 0
    for num in lst:
```

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/pytho
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

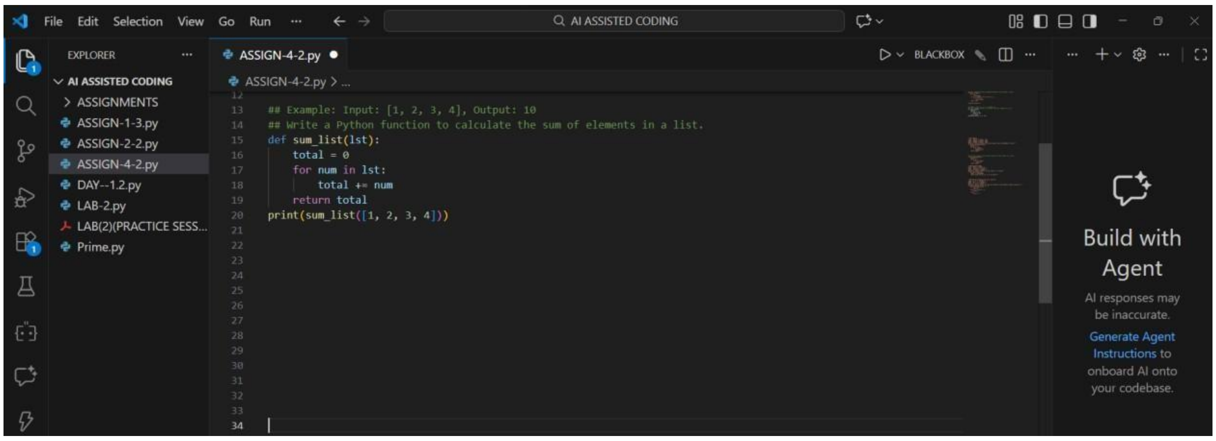
    Square-root optimization

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.

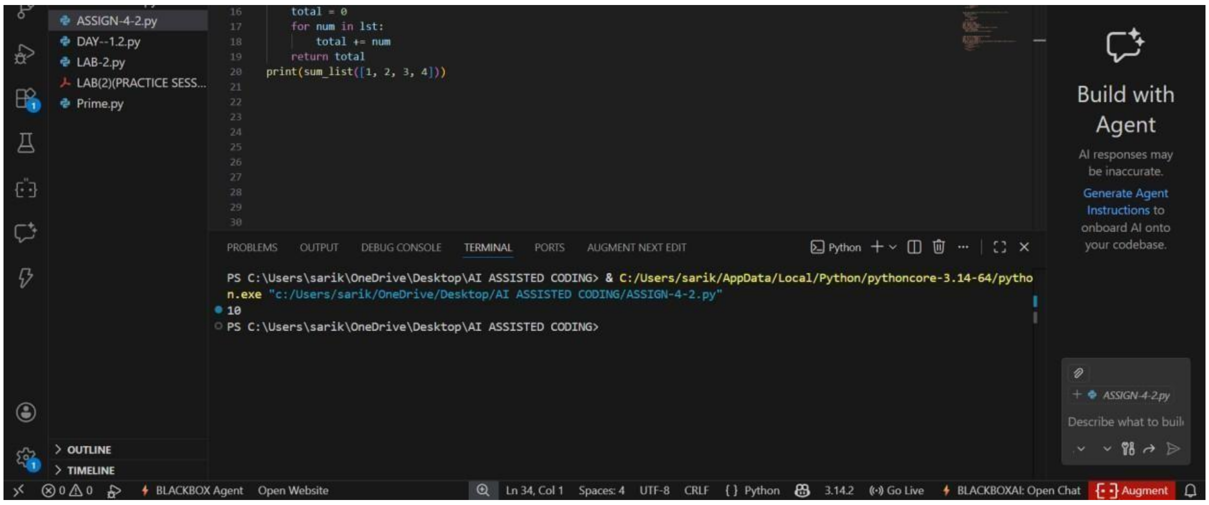
Example: Input: [1, 2, 3, 4], Output: 10



```python
## Example: Input: [1, 2, 3, 4], Output: 10
## Write a Python function to calculate the sum of elements in a list.
def sum_list(lst):
    total = 0
    for num in lst:
        total += num
    return total
print(sum_list([1, 2, 3, 4]))
```

AI ASSISTED CODING
ASSIGN-4-2.py
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

OUTPUT:



```python
    total = 0
    for num in lst:
        total += num
    return total
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"
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

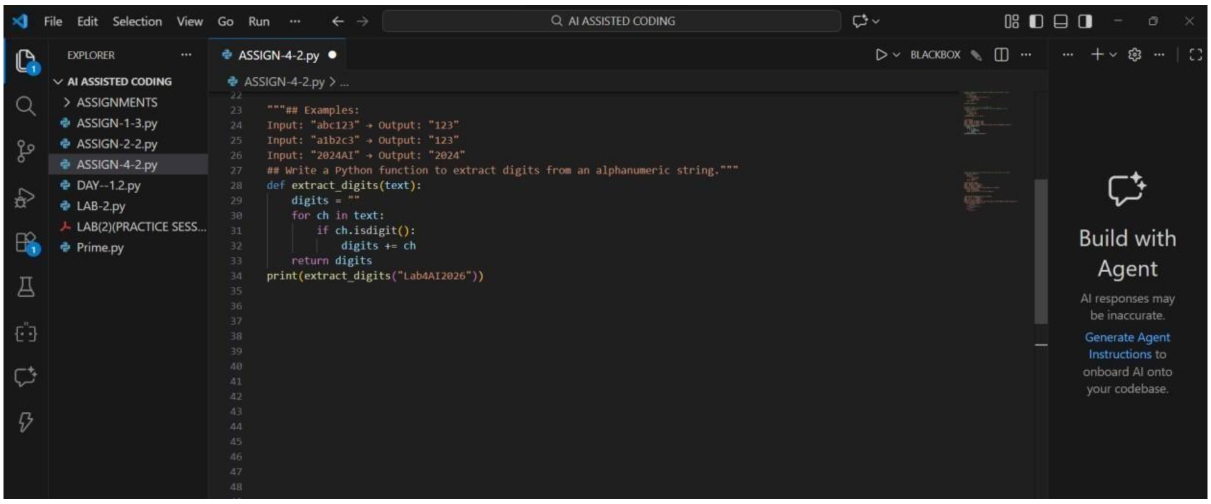
Prompt: Write a Python function to extract digits from an alphanumeric string.

Examples:

Input: "abc123" → Output: "123"

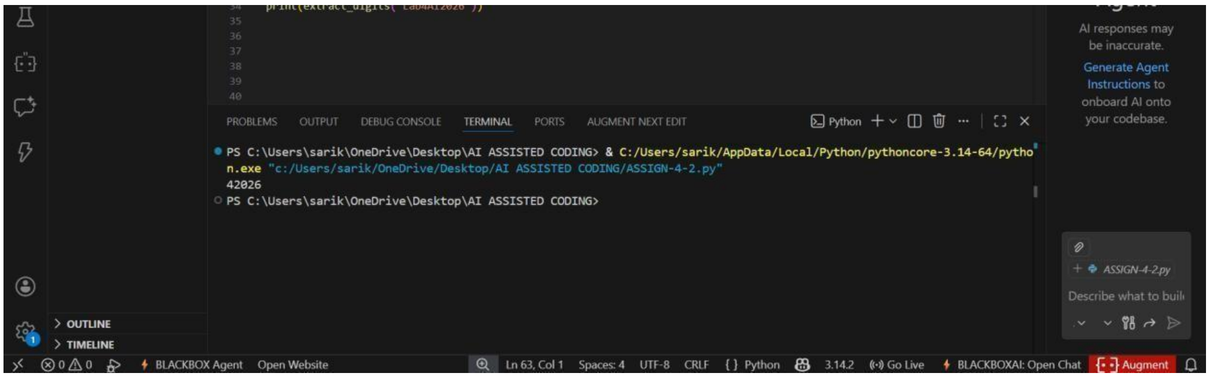
Input: "a1b2c3" → Output: "123"

Input: "2024AI" → Output: "2024"



```
"""## Examples:
Input: "abc123" → Output: "123"
Input: "a1b2c3" → Output: "123"
Input: "2024AI" → Output: "2024"
## Write a Python function to extract digits from an alphanumeric string."""
def extract_digits(text):
    digits = ""
    for ch in text:
        if ch.isdigit():
            digits += ch
    return digits
print(extract_digits("Lab4AI2026"))
```

OUTPUT:


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AUGMENT NEXT EDIT
Python
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
ASSIGN-4-2.py
Describe what to build
OUTLINE
TIMELINE
BLACKBOX Agent Open Website
Ln 63, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.14.2 Go Live BLACKBOXAI: Open Chat Augment

```
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"
42026
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

Explanation:

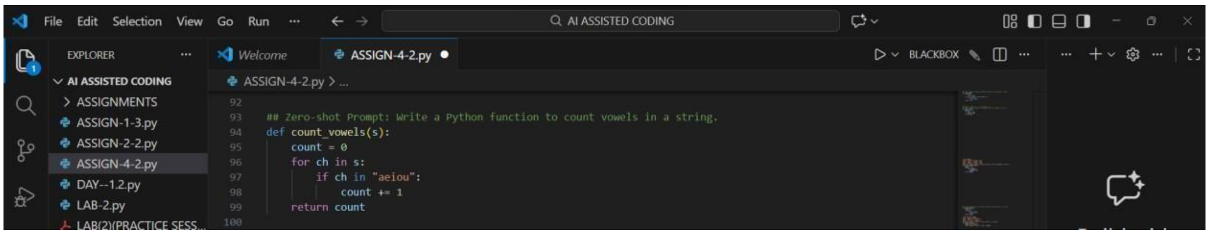
1. Few-shot prompting provides pattern recognition.
2. AI correctly:

   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.

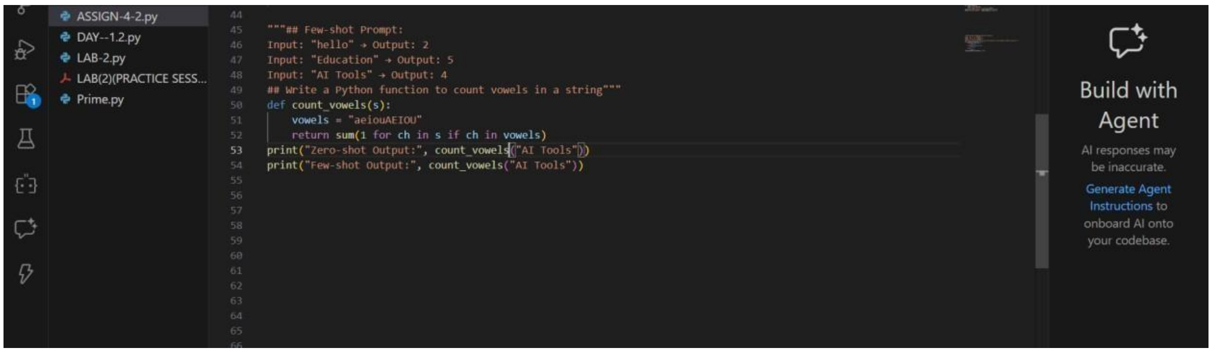


```python
## Zero-shot Prompt: Write a Python function to count vowels in a string.
def count_vowels(s):
    count = 0
    for ch in s:
        if ch in "aeiou":
            count += 1
    return count
```

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

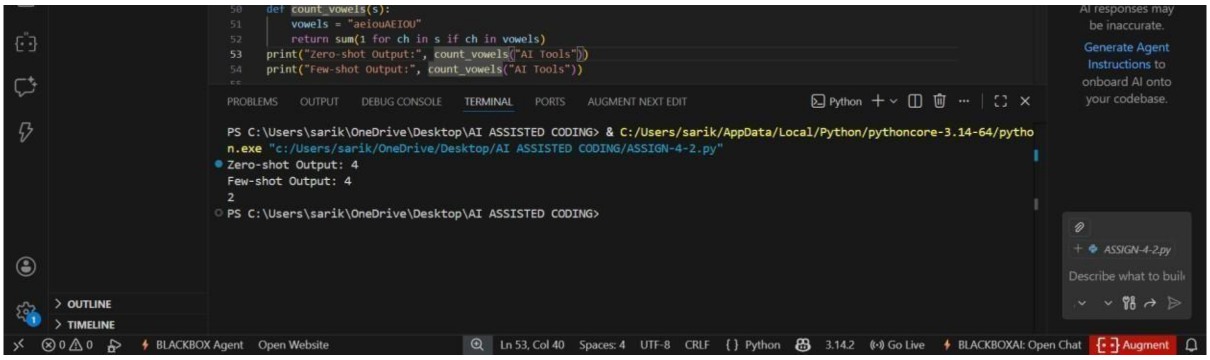
Examples:

Input: "hello" → Output: 2
Input: "Education" → Output: 5
Input: "AI Tools" → Output: 4



```python
"""## Few-shot Prompt:
Input: "hello" → Output: 2
Input: "Education" → Output: 5
Input: "AI Tools" → Output: 4
## Write a Python function to count vowels in a string"""
def count_vowels(s):
    vowels = "aeiouAEIOU"
    return sum(1 for ch in s if ch in vowels)
print("Zero-shot Output:", count_vowels("AI Tools"))
print("Few-shot Output:", count_vowels("AI Tools"))
```

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>
```

Comparison Table:

Feature	**Zero-shot**	**Few-shot**
		Upper &

Case handling	Only lowercase	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)

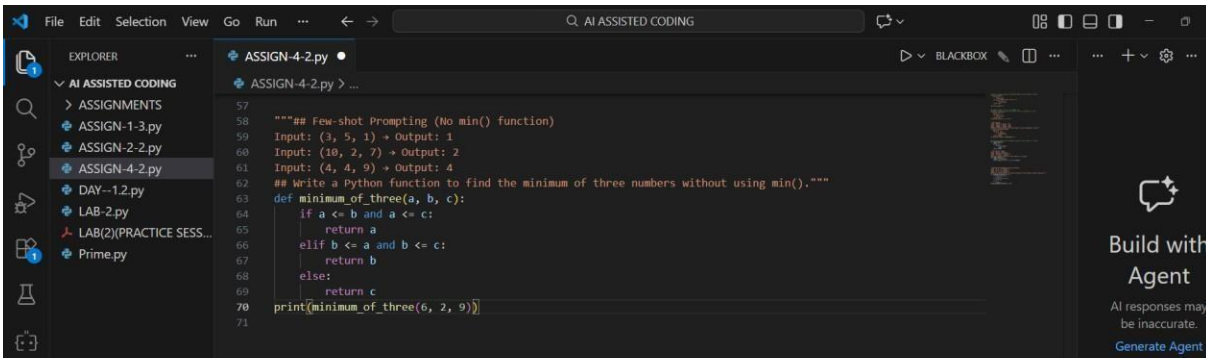
Prompt: Write a Python function to find the minimum of three numbers without using min().

Examples:

Input: (3, 5, 1) → Output: 1

Input: (10, 2, 7) → Output: 2

Input: (4, 4, 9) → Output: 4

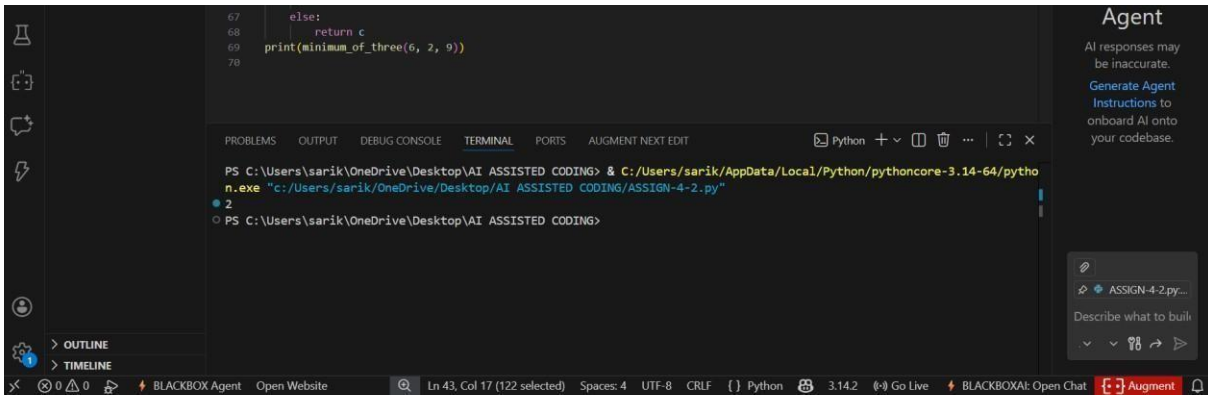


```python

"""## Few-shot Prompting (No min() function)
Input: (3, 5, 1) → Output: 1
Input: (10, 2, 7) → Output: 2
Input: (4, 4, 9) → Output: 4
## Write a Python function to find the minimum of three numbers without using min()."""
def minimum_of_three(a, b, c):
    if a <= b and a <= c:
        return a
    elif b <= a and b <= c:
        return b
    else:
        return c
print(minimum_of_three(6, 2, 9))

```

OUTPUT:



```
    else:
        return c
print(minimum_of_three(6, 2, 9))
```

```
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>
```

Explanation:

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

   All ordering cases

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