

ASSIGNMENT-4.2

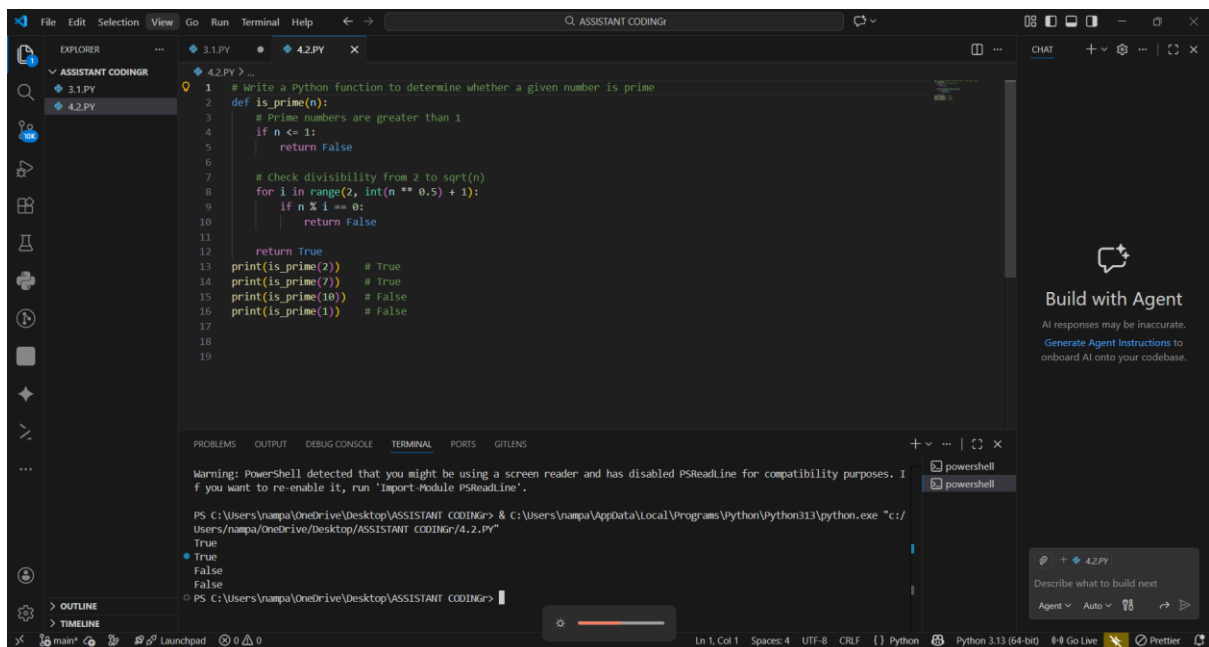
NAME: AKSHITHA

ROLL NO:2303A51360

TASK-1: ZERO-SHOT PROMPTING

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

CODE:



```
1 # write a python function to determine whether a given number is prime
2 def is_prime(n):
3     # Prime numbers are greater than 1
4     if n <= 1:
5         return False
6
7     # Check divisibility from 2 to sqrt(n)
8     for i in range(2, int(n ** 0.5) + 1):
9         if n % i == 0:
10            return False
11
12    return True
13
14    print(is_prime(2)) # True
15    print(is_prime(7)) # True
16    print(is_prime(10)) # False
17    print(is_prime(1)) # False
18
19
```

Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING> & c:\Users\nampa\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/nampa/OneDrive/Desktop/ASSISTANT CODING/4.2.py"

True
True
False
False

OBSERVATION:

- AI model understands the concept of a prime number without being given any examples or additional guidance
- It applies correct mathematical reasoning purely from the instruction
- The model generates syntactically correct and logically sound python code

TASK-2

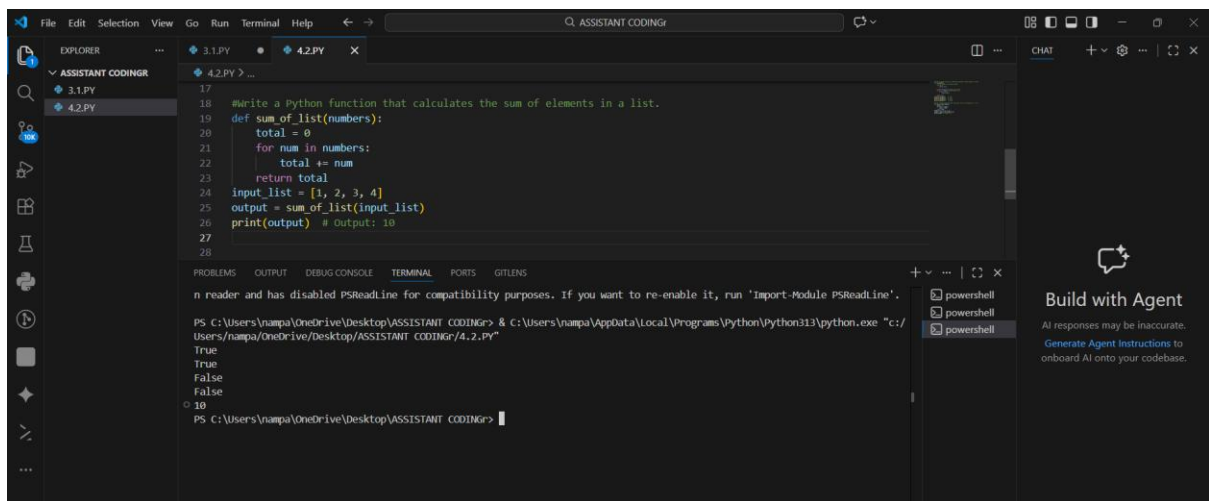
PROMPT: Write a Python function that calculates the sum of elements in a list.

Example:

Input: [1, 2, 3, 4]

Output: 10

CODE:

A screenshot of a code editor interface. The main editor window shows a Python script with the following code:

```
17 #Write a Python function that calculates the sum of elements in a list.
18 def sum_of_list(numbers):
19     total = 0
20     for num in numbers:
21         total += num
22     return total
23
24 input_list = [1, 2, 3, 4]
25 output = sum_of_list(input_list)
26 print(output) # Output: 10
27
28
```

The left sidebar shows the Explorer view with a folder named 'ASSISTANT CODING' containing two files: '3.1.PY' and '4.2.PY'. The bottom panel shows the Terminal view with the following output:

```
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING> & c:\Users\nampa\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/nampa/OneDrive/Desktop/ASSISTANT CODING/4.2.Py"
True
False
False
10
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING>
```

The right sidebar shows a 'CHAT' panel with a 'Build with Agent' section and a 'Generate Agent Instructions' button.

OBSERVATION:

The single example guides the AI model to understand the expected input and output relationship

The model correctly generalizes the pattern from the example to any list of numbers

TASK-3

PROMPT: Write a Python function that extracts only digits from an alphanumeric string.

Examples:

Input: "a1b2c3"

Output: "123"

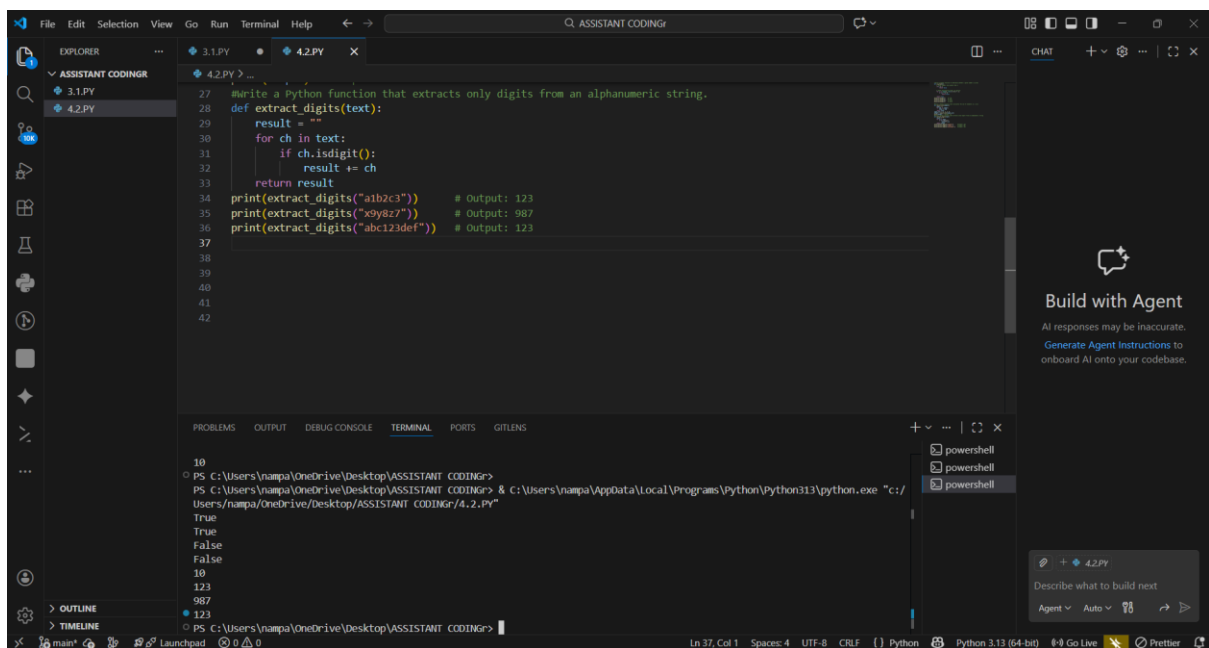
Input: "x9y8z7"

Output: "987"

Input: "abc123def"

Output: "123"

CODE:



```
27 # Write a Python function that extracts only digits from an alphanumeric string.
28 def extract_digits(text):
29     result = ""
30     for ch in text:
31         if ch.isdigit():
32             result += ch
33     return result
34 print(extract_digits("a1b2c3")) # Output: 123
35 print(extract_digits("x9y8z7")) # Output: 987
36 print(extract_digits("abc123def")) # Output: 123
37
38
39
40
41
42
```

```
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING>
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING> & C:\Users\nampa\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/nampa/OneDrive/Desktop/ASSISTANT CODING/4.2.Py"
True
True
False
False
10
123
987
123
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING>
```

OBSERVATION:

- Multiple examples help the AI model clearly identify the pattern to be learned
- The model focuses only on digit characters and ignores alphabetic content
- The AI demonstrates improved confidence and reduced ambiguity compared to zero shot and one shot prompting

TASK-4

PROMPT: ZERO-SHOT: Write a Python function that counts the number of vowels in a string.

FEW-SHOT: Write a Python function that counts the number of vowels in a string.

Examples:

Input: "hello"

Output: 2

Input: "AEIOU"

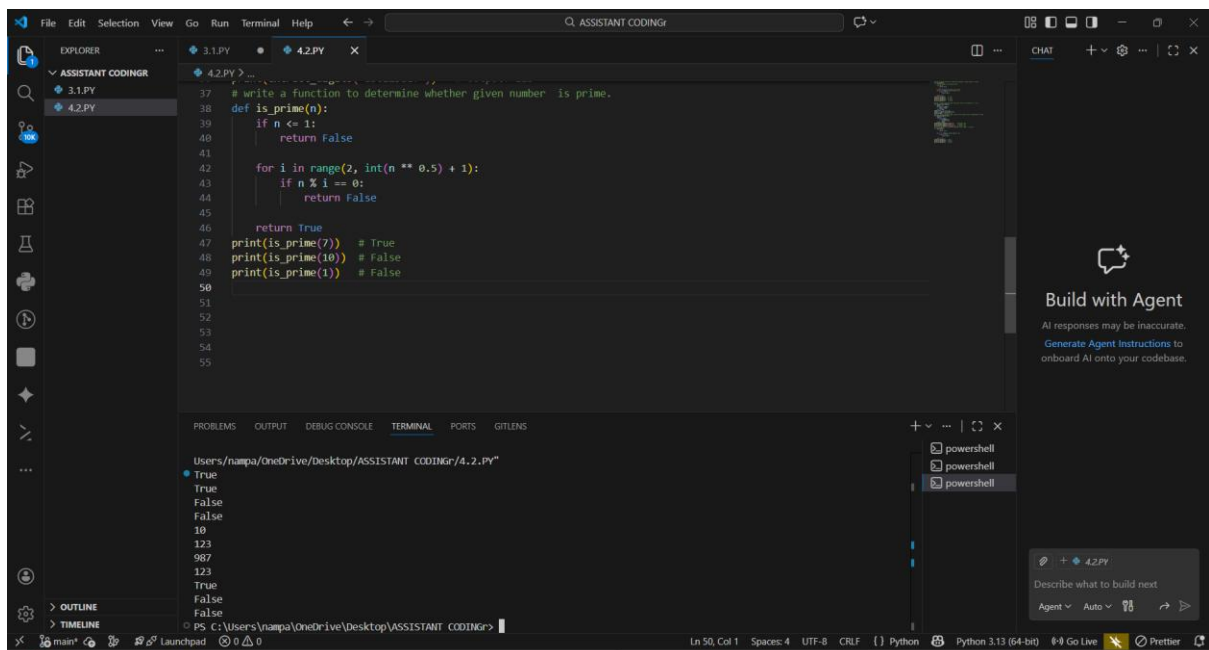
Output: 5

Input: "chatgpt"

Output: 2

CODE:

ZERO-SHOT:



The screenshot shows a Visual Studio Code editor with a Python file named 4.2.PY. The code defines a function `is_prime(n)` that checks if a number is prime. Below the function, there are test cases: `print(is_prime(7))`, `print(is_prime(10))`, and `print(is_prime(1))`. The terminal output shows the results of these tests: `True`, `False`, and `False` respectively. The status bar at the bottom indicates the file is using Python 3.13 (64-bit) and the Prettier formatter.

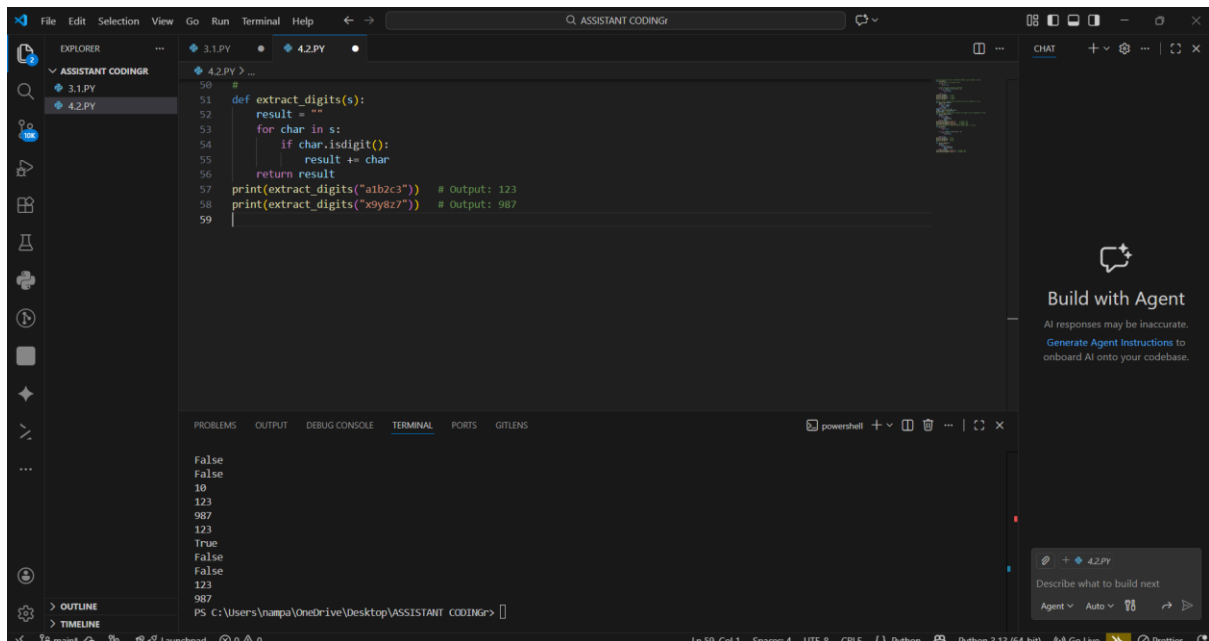
```
37 # write a function to determine whether given number is prime.
38 def is_prime(n):
39     if n <= 1:
40         return False
41
42     for i in range(2, int(n ** 0.5) + 1):
43         if n % i == 0:
44             return False
45
46     return True
47 print(is_prime(7)) # True
48 print(is_prime(10)) # False
49 print(is_prime(1)) # False
50
51
52
53
54
55
```

Users\nampa\OneDrive\Desktop\ASSISTANT CODING\4.2.PY

```
True
True
False
False
10
123
987
123
True
False
```

Ln 50, Col 1 Spaces: 4 UTF-8 CRLF Python Python 3.13 (64-bit) Prettier

FEW -SHOT:



```
def count_vowels(text):
```

```
    vowels = "aeiouAEIOU"
```

```
    count = 0
```

```
    for ch in text:
```

```
        if ch in vowels:
```

```
            count += 1
```

```
    return count
```

OBSERVATION:

FEW-SHOT OBSERVATION:

The provided examples clearly define what characters should be counted as vowels

The model confidently includes both uppercase and lowercase vowels due to examples

ZERO SHOT:

zero shot prompting the AI guesses the intent based on general knowledge which may vary for ambiguous tasks

TASK-5

PROMPT:

Write a Python function that determines the minimum of three numbers without using the built-in min() function.

Examples:

Input: 3, 7, 5

Output: 3

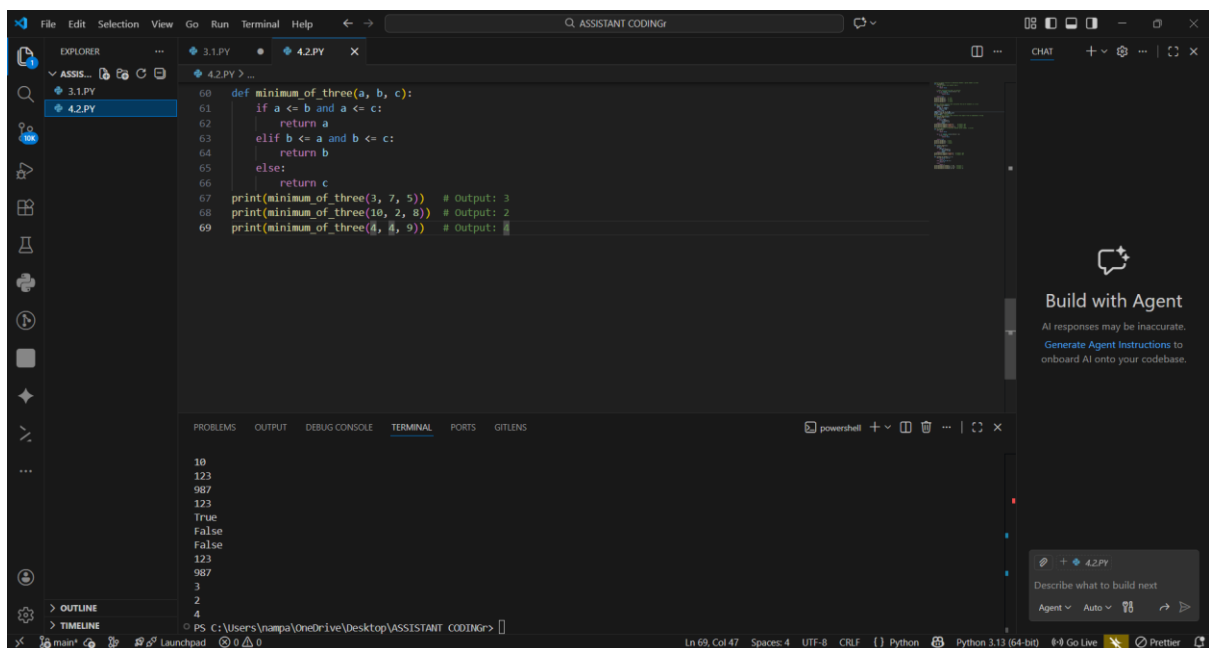
Input: 10, 2, 8

Output: 2

Input: 4, 4, 9

Output: 4

CODE:



```
File Edit Selection View Go Run Terminal Help
ASSISTANT CODING

EXPLORER
3.1.PY
4.2.PY
ASSISTANT CODING

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(3, 7, 5)) # Output: 3
print(minimum_of_three(10, 2, 8)) # Output: 2
print(minimum_of_three(4, 4, 9)) # Output: 4

TERMINAL
10
123
987
123
True
False
False
123
987
3
2
4
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING>
```

OBSERVATION:

The examples clearly establish the comparison pattern needed to identify the smallest value

The AI model infers the requirement to handle equality cases correctly

Conditional logic is generated without relying on built-in functions