

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:

The screenshot shows the Visual Studio Code interface with the AI Assistant extension open. The code editor displays a Python file named 4.2.PY containing the following 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 print(is_prime(2))    # True
14 print(is_prime(7))    # True
15 print(is_prime(10))   # False
16 print(is_prime(1))    # False
```

The terminal below shows the output of running the script:

```
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
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING>
```

The AI Assistant sidebar on the right is visible, showing a message: "Build with Agent".

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:

```
File Edit Selection View Go Run Terminal Help <- > ASSISTANT CODING| EXPLORER 3.1.PY 4.2.PY 4.2.PY > ... ASSISTANT CODING| 3.1.PY 4.2.PY 17 18 #write a Python function that calculates the sum of elements in a list. 19 def sum_of_list(numbers): 20     total = 0 21     for num in numbers: 22         total += num 23     return total 24 input_list = [1, 2, 3, 4] 25 output = sum_of_list(input_list) 26 print(output) # Output: 10 27 28 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS n 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 10 PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING> Build with Agent AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase.
```

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"

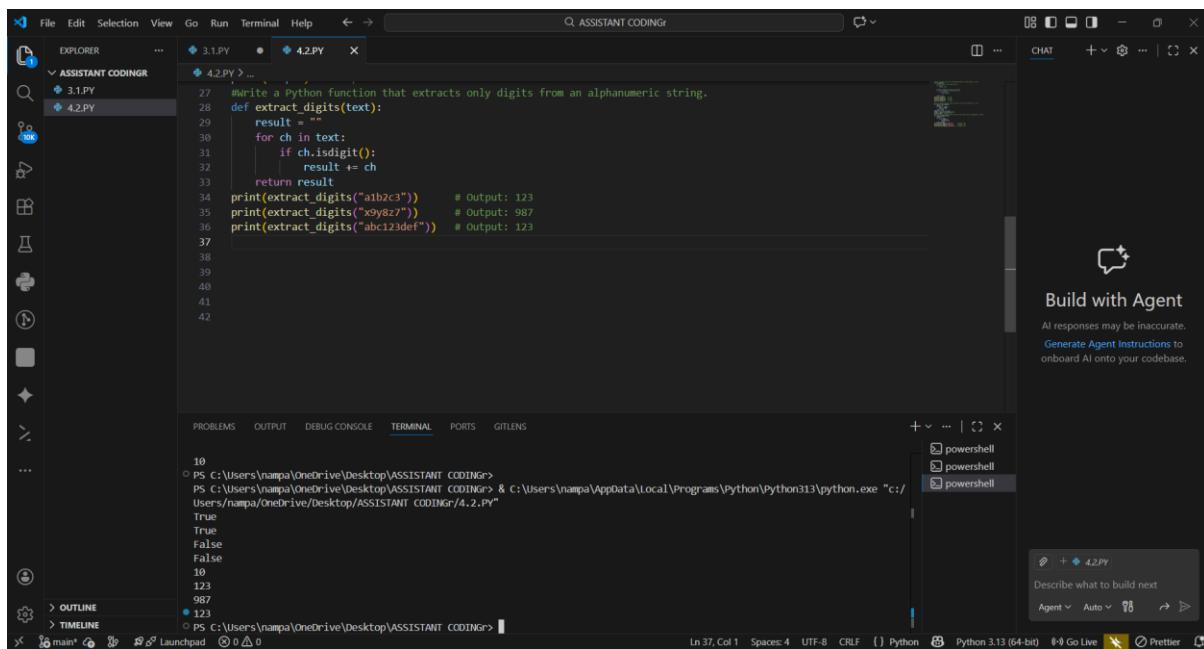
Input: "x9y8z7"

Output: "987"

Input: "abc123def"

Output: "123"

CODE:



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files 3.1.PY, 4.1.PY, and 4.2.PY. 4.2.PY is the active file.
- Code Editor:** Displays the following Python 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("xyyz7"))       # Output: 987
36 print(extract_digits("abc123def"))   # Output: 123
```
- Terminal:** Shows command-line output for 4.2.PY execution.
- Right Panel:** Contains a "Build with Agent" section with a message: "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase."

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 the Visual Studio Code interface. The Explorer sidebar on the left has 'ASSISTANT CODINGR' expanded, showing files '3.1.PY' and '4.2.PY'. The '4.2.PY' file is open in the center editor, displaying the following code:

```
# write a function to determine whether 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)) # True
print(is_prime(10)) # False
print(is_prime(1)) # False
```

The 'OUTPUT' tab at the bottom shows the execution results:

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

A 'CHAT' panel on the right contains a message from an AI agent:

Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

FEW -SHOT:

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

- File Structure:** Explorer sidebar shows files 3.1.PY and 4.2.PY.
- Code Editor:** Content of 4.2.PY is displayed:

```
def extract_digits(s):
    result = ""
    for char in s:
        if char.isdigit():
            result += char
    return result

print(extract_digits("a1b2c3")) # Output: 123
print(extract_digits("x9y8z7")) # Output: 987
```
- Terminal:** Output pane shows the execution of the code:

```
False
False
10
123
987
123
True
False
False
123
987
```
- Right Panel:** "Build with Agent" section with instructions to onboard AI onto the codebase.

```
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:

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar labeled "ASSISTANT CODING". The Explorer sidebar on the left shows files 3.1.PY and 4.2.PY. The main editor area contains the following Python code:

```
60 def minimum_of_three(a, b, c):
61     if a <= b and a <= c:
62         return a
63     elif b <= a and b <= c:
64         return b
65     else:
66         return c
67 print(minimum_of_three(3, 7, 5)) # Output: 3
68 print(minimum_of_three(10, 2, 8)) # Output: 2
69 print(minimum_of_three(4, 4, 9)) # Output: 4
```

The bottom terminal window shows the output of the code execution:

```
10
123
987
123
True
False
False
123
987
3
2
4
```

A floating "Build with Agent" interface is visible on the right side of the screen, with the message "Describe what to build next".

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