

NAME:B.VYSHNAVI

COURSE NAME : AIAC

2303A51884

WEEK - 3

BATCH-02

Question 1: Zero-Shot Prompting (Palindrome Number Program)

Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a palindrome.

Task:

- Record the AI-generated code.
- Test the code with multiple inputs.
- Identify any logical errors or missing edge-case handling.

The screenshot shows a code editor interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. The left sidebar lists several files: question 1.py (the current file), ques 2.py, question 3.py, question 4.py, question 5.py, and question 6.py. The right sidebar features a "Build with Agent" section with a message: "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase." Below the editor is a terminal window showing the execution of the script. The terminal output is as follows:

```
py"
Enter a number: 121
121 is a palindrome.
PS C:\Users\vyshn> & C:/Users/vyshn/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/vyshn/OneDrive/Desktop/aiac/assignment 3/question 1.py"
py"
Enter a number: 31
31 is not a palindrome.
PS C:\Users\vyshn> & C:/Users/vyshn/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/vyshn/OneDrive/Desktop/aiac/assignment 3/question 1.py"
py"
Enter a number: 657756
657756 is a palindrome.
PS C:\Users\vyshn>
```

The bottom status bar indicates the file is question 1.py, line 16, column 1, with 4 spaces, using UTF-8 encoding, and the date/time is 19-01-2026. The system tray shows the temperature as 31°C and weather as sunny.

Question 2: One-Shot Prompting (Factorial Calculation)

Write a one-shot prompt by providing one input-output example and ask the AI to generate a Python function to compute the factorial of a given number.

Example:

Input: 5 → Output: 120

Task:

- Compare the generated code with a zero-shot solution.
- Examine improvements in clarity and correctness.

```
question 1.py ques 2.py question 3.py question 4.py question 5.py question 6.py
C:\Users\vyshn>OneDrive > Desktop > aiac > assignemnt 3 > ques 2.py > ...
1
2     num = 5
3     print is factorial number
4
5     def factorial(n):
6         if n == 0 or n == 1:
7             return 1
8         else:
9             return n * factorial(n - 1)
10    num = 5
11    print(f"The factorial of {num} is {factorial(num)}")
```

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

```
PS C:\Users\vyshn> & c:/users/vyshn/appdata/local/programs/python/python313/python.exe "c:/users/vyshn/onedrive/desktop/aiac/assignemnt_3/ques 2.py"
The factorial of 5 is 120
PS C:\Users\vyshn>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

LN 12, COL 1 SPACES: 4 UTF-8 CRLF Python 3.13.3 ENG IN 16:34 19-01-2026

31°C Sunny

Question 3: Few-Shot Prompting (Armstrong Number Check)

Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python function to check whether a given number is an Armstrong number.

Examples:

- Input: 153 → Output: Armstrong Number
- Input: 370 → Output: Armstrong Number
- Input: 123 → Output: Not an Armstrong Number

Task:

- Analyze how multiple examples influence code structure and accuracy.
- Test the function with boundary values and invalid inputs.

The screenshot shows a code editor interface with a dark theme. The main pane displays a Python script named `question 3.py`. The script defines a function `is_armstrong_number` that checks if a given number is an Armstrong number by summing the powers of its digits. It includes a check against a list of test numbers [153, 370, 123]. The terminal below shows the script running and printing the results for these numbers. A tooltip in the bottom right corner suggests opening a git repository found in the workspace.

```
C:\> Users > vyshn > OneDrive > Desktop > aiac > assignmnet 3 > question 3.py > ...
1 num = 153
2 print is an armstrong number or not
3 num = 370
4 print is an armstrong number or not
5 num = 123
6 print is an armstrong number or not
7 ...
8
9 def is_armstrong_number(num):
10     # Convert the number to string to easily iterate over digits
11     digits = str(num)
12     num_digits = len(digits)
13
14     # Calculate the sum of each digit raised to the power of num_digits
15     sum_of_powers = sum(int(digit) ** num_digits for digit in digits)
16
17     # An Armstrong number is one where the sum of the powers equals the original number
18     return sum_of_powers == num
19 if __name__ == "__main__":
20     test_numbers = [153, 370, 123]
21     for number in test_numbers:
22         if is_armstrong_number(number):
23             print(f"{number} is an Armstrong number.")
24         else:
25             print(f"{number} is not an Armstrong number.")

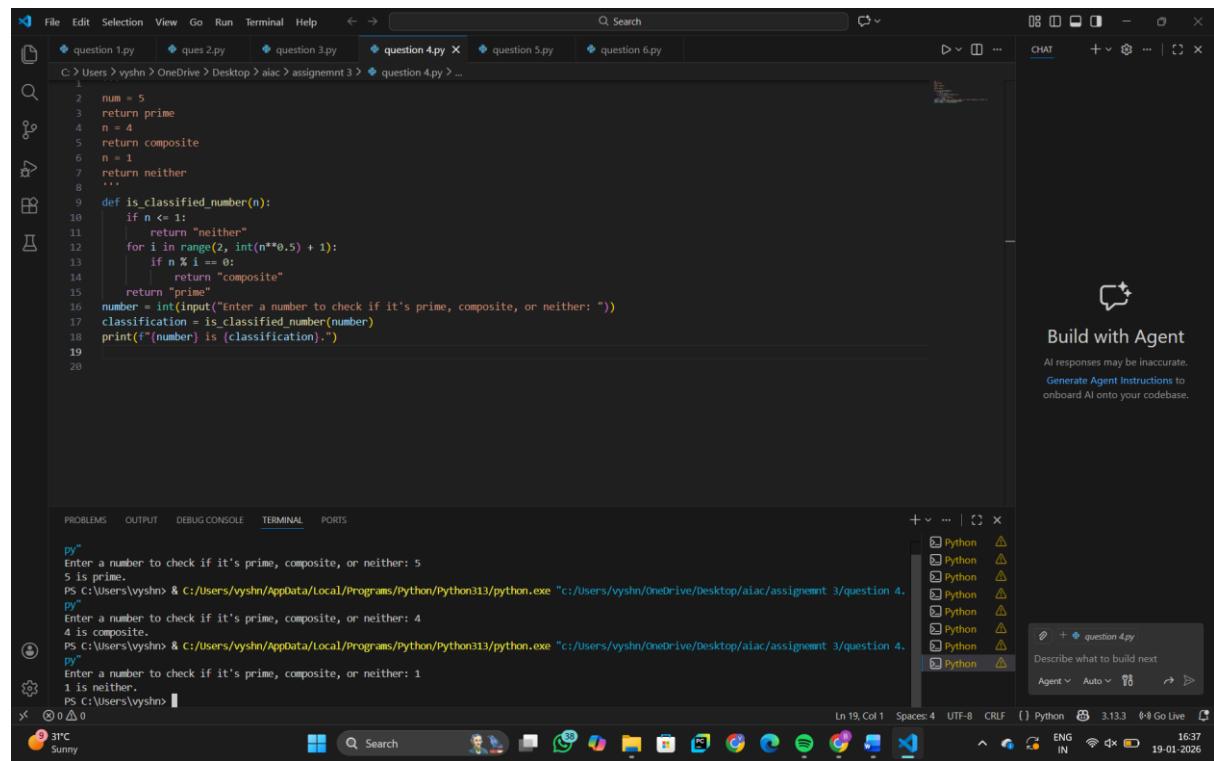
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\vyshn> & C:/Users/vyshn/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/vyshn/OneDrive/Desktop/aiac/assignmnet 3/question 3.py"
153 is an Armstrong number.
370 is an Armstrong number.
123 is not an Armstrong number.
PS C:\Users\vyshn>
```

Question 4: Context-Managed Prompting (Optimized Number Classification)

Design a context-managed prompt with clear instructions and constraints to generate an optimized Python program that classifies a number as prime, composite, or neither.

Task:

- Ensure proper input validation.
- Optimize the logic for efficiency.
- Compare the output with earlier prompting strategies.



The screenshot shows a code editor interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. Below the menu is a tab bar with several files: question 1.py, ques 2.py, question 3.py, question 4.py (which is currently selected), question 5.py, and question 6.py. The main code area contains the following Python script:

```
C:\> Users > vyshn > OneDrive > Desktop > aiac > assignemnt 3 > question 4.py > ...
1
2     num = 5
3     return prime
4     n = 4
5     return composite
6     n = 1
7     return neither
8     ...
9
10    def is_classified_number(n):
11        if n <= 1:
12            return "neither"
13        for i in range(2, int(n**0.5) + 1):
14            if n % i == 0:
15                return "composite"
16        return "prime"
17
18 number = int(input("Enter a number to check if it's prime, composite, or neither: "))
19 classification = is_classified_number(number)
20 print(f"Number {number} is {classification}.")
```

To the right of the code editor, there is a sidebar titled "Build with Agent". It includes a button to "Build with Agent", a note that AI responses may be inaccurate, and a link to "Generate Agent Instructions".

At the bottom of the screen, there is a terminal window showing the execution of the script. The terminal output is as follows:

```
py"
Enter a number to check if it's prime, composite, or neither: 5
5 is prime.
PS C:\Users\vyshn> & c:/Users/vyshn/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/vyshn/OneDrive/Desktop/aiac/assignemnt 3/question 4.py"
py"
Enter a number to check if it's prime, composite, or neither: 4
4 is composite.
PS C:\Users\vyshn> & c:/Users/vyshn/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/vyshn/OneDrive/Desktop/aiac/assignemnt 3/question 4.py"
py"
Enter a number to check if it's prime, composite, or neither: 1
1 is neither.
PS C:\Users\vyshn>
```

The terminal also displays system status information at the bottom, including battery level (31%), weather (Sunny), and system time (16:37 19-01-2026).

Question 5: Zero-Shot Prompting (Perfect Number Check)

Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a perfect number.

Task:

- Record the AI-generated code.
- Test the program with multiple inputs.
- Identify any missing conditions or inefficiencies in the logic.

The screenshot shows a code editor interface with a Python file named 'question 5.py' open. The code defines a function 'is_perfect_number' that calculates the sum of divisors for a given number and checks if it equals the number itself. It also includes a main block that prompts the user for a number and prints whether it is a perfect number or not. The code editor has a sidebar for 'Build with Agent' and a terminal tab at the bottom showing the execution of the script and its output for two different numbers.

```
C:\> Users > vyshn > OneDrive > Desktop > aiac > assignemnt 3 > question 5.py > ...
...
1  ...
2  write a python program to check whether a given number is perfect number or not.
3  ...
4  def is_perfect_number(num):
5      sum_of_divisors = 0
6      for i in range(1, num):
7          if num % i == 0:
8              sum_of_divisors += i
9      return sum_of_divisors == num
10 if __name__ == "__main__":
11     number = int(input("Enter a number: "))
12     if is_perfect_number(number):
13         print(f"{number} is a perfect number.")
14     else:
15         print(f"{number} is not a perfect number.")

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
py"
Enter a number: 25
25 is not a perfect number.
PS C:\Users\vyshn> & c:/Users/vyshn/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/vyshn/OneDrive/Desktop/aiac/assignemnt 3/question 5.py"
py"
Enter a number: 1
1 is not a perfect number.
PS C:\Users\vyshn> & c:/Users/vyshn/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/vyshn/OneDrive/Desktop/aiac/assignemnt 3/question 5.py"
py"
Enter a number: 7
7 is not a perfect number.
PS C:\Users\vyshn>
Ln 16, Col 1 Spaces: 4 UFT-8 CRLF Python ENG IN 3.13.3 1638 19.01.2026
```

Question 6: Few-Shot Prompting (Even or Odd Classification with Validation)

Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python program that determines whether a given number is even or odd, including proper input validation.

Examples:

- Input: 8 → Output: Even
- Input: 15 → Output: Odd
- Input: 0 → Output: Even

Task:

- Analyze how examples improve input handling and output clarity.
- Test the program with negative numbers and non-integer inputs.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The left pane displays a Python file named `question 6.py` with the following code:

```
1 ...
2 num = 8
3 print is a even number
4 num = 15
5 print is a odd number
6 num = 0
7 print is a even number
8 ...
9 def iseven_number(num):
10     """ IGNORE ---
11     This function checks if a number is even.
12     --- IGNORE ---
13     """
14     return num % 2 == 0
15 # Example usage
16 numbers = [8, 15, 0]
17 for number in numbers:
18     if iseven_number(number):
19         print(f"{number} is an even number.")
20     else:
21         print(f"{number} is an odd number.")
```

The right pane shows a "Build with Agent" interface with a message: "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase." Below it is a "PROBLEMS" panel showing no errors. At the bottom, a terminal window shows the command `python question 6.py` running and outputting the expected results for the numbers 8, 15, and 0. A small modal dialog at the bottom right asks if a git repository was found in the workspace, with options "Yes", "Always", and "Never". The status bar at the bottom indicates the file is saved, the language is Python, and the current date and time are 19-01-2026.