

Lab Assignment – 3.1

Hall Ticket No.: 2303A510E6

Name – Anushka Boora

Batch – 29

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.

CODE & OUTPUT

```
C:\> Users> boora> Downloads> aac3.1.py > ...
1 # Zero-Shot Prompting Palindrome Number Program
2 def is_palindrome_number(num):
3     """Check if a number is a palindrome."""
4     str_num = str(num)
5     return str_num == str_num[::-1]
6 if __name__ == "__main__":
7     number = int(input("Enter a number: "))
8     if is_palindrome_number(number):
9         print(f"{number} is a palindrome number.")
10    else:
11        print(f"{number} is not a palindrome number.")

15 # One-Shot Prompting Factorial Calculation
16 def factorial(n):
17     """Calculate the factorial of a number."""
18     if n == 0 or n == 1:
19         return 1
PS C:\Users\boora\Downloads> & 'c:/Users\boora\AppData\Local\Programs\Python\Python311\python.exe' 'c:/Users\boora\.vscode\extensions\ms-python.python.debug-202.5.18.0-win32-x64\bundle\libs\debugpy\launcher' '52521' '--' 'C:/Users\boora/Downloads/aac3.1.py'
Enter a number: 16
16 is not a palindrome number.
Enter a number to calculate its factorial: 11
The factorial of 11 is 39916800.
Enter a number: 28
28 is an Armstrong number.
Enter a number: 77
The number 77.0 is positive.
Enter a number: 16
16 is not a perfect number.
Enter an integer: 28
The number 28 is even.
PS C:\Users\boora\Downloads>
```

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.

CODE & OUTPUT

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files `aac1.py`, `aac3.1.py`, and `tr.py`.
- Run and Debug View:** Shows a configuration for "Run and Debug". It includes instructions to "Customize Run and Debug, open a folder and create a launch.json file." and "Show automatic Python configurations".
- Code Editor:** Displays two Python scripts:
 - `aac3.1.py` contains a factorial function and logic to check if a number is Armstrong.
 - `tr.py` contains a few-shot prompt for Armstrong numbers.
- Terminal:** Shows the output of running the code. It prompts for a number, calculates its factorial, and checks if it's an Armstrong number. The terminal also shows the path to the Python executable and the extension used.
- Breakpoints:** Shows breakpoints set at various lines of code.

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.

CODE & OUTPUT

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

- File Explorer:** Shows files `aac1.py`, `aac3.1.py`, and `t.py`.
- Run View:** Shows the "Run and Debug" tab selected. It includes instructions to customize run and debug settings or use a terminal or interactive chat, and a link to automatic Python configurations.
- Code Editor:** Displays Python code for checking if a number is an Armstrong number. The code uses string manipulation and loops to calculate the sum of powers of digits.
- Terminal:** Shows the command line output of the program's execution. It prompts for a number, checks if it's a palindrome, calculates its factorial, and prints the results.
- Breakpoints:** A sidebar shows breakpoints for raised exceptions, uncaught exceptions, and user uncaught exceptions.
- Status Bar:** Shows "Spaces: 4", "UTF-8", "Python", and "3.11.5".

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.

CODE & OUTPUT

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

- File Explorer:** Shows files `aac1.py`, `aac3.1.py`, and `tr.py`.
- Code Editor:** Displays Python code for classifying numbers and calculating factorials.
- Terminal:** Shows the output of running the code, including user inputs and program responses.
- Debug View:** Shows breakpoints and exception settings.

```
43
44
45 # Context-Managed Prompting Optimized Number Classification
46 def classify_number(num):
47     """Classify a number as positive, negative, or zero."""
48     if num > 0:
49         return "positive"
50     elif num < 0:
51         return "negative"
52     else:
53         return "zero"
54 if __name__ == "__main__":
55     number = float(input("Enter a number: "))
56     classification = classify_number(number)
57     print(f"The number {number} is {classification}.")
58
59
60
61 # Zero-Shot Prompting Perfect Number Check
PS C:\Users\boora\Downloads> & 'c:\Users\boora\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\boora\.vscode\extensions\ms-python.debugpy-202
5.18.0-win32-x64\bundled\libs\debugpy\launcher' '52521' '-' 'C:\Users\boora\Downloads\aac3.1.py'
Enter a number: 16
16 is not a palindrome number.
Enter a number to calculate its factorial: 11
The factorial of 11 is 39916800.
Enter a number: 28
28 is not an Armstrong number.
Enter a number: 77
The number 77.0 is positive.
Enter a number: 16
16 is not a perfect number.
Enter an integer: 28
The number 28 is even.
PS C:\Users\boora\Downloads>
```

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.

CODE & OUTPUT

The screenshot shows the Visual Studio Code interface with a Python file named 'aac3.1.py' open. The code defines a function 'is_perfect_number' that checks if a number is perfect. It also includes logic to handle command-line input and print results. The terminal below shows the execution of the script and its output for various numbers.

```
59
60
61 # Zero-Shot Prompting Perfect Number Check
62 def is_perfect_number(num):
63     """Check if a number is a perfect number."""
64     if num < 1:
65         return False
66     divisors_sum = sum(i for i in range(1, num) if num % i == 0)
67     return divisors_sum == num
68 if __name__ == "__main__":
69     number = int(input("Enter a number: "))
70     if is_perfect_number(number):
71         print(f"{number} is a perfect number.")
72     else:
73         print(f"{number} is not a perfect number.")
74
75
76
PS C:\Users\boora\Downloads> & 'c:\Users\boora\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\boora\.vscode\extensions\ms-python.debugpy-202
5.18.0-win32-x64\bundles\libs\debugpy\launcher' '52521' '-' 'C:\Users\boora\Downloads\aac3.1.py'
Enter a number: 16
16 is not a palindrome number.
Enter a number to calculate its factorial: 11
The factorial of 11 is 39916800.
Enter a number: 28
28 is not an Armstrong number.
Enter a number: 77
The number 77.0 is positive.
Enter a number: 16
16 is not a perfect number.
Enter an integer: 28
The number 28 is even.
PS C:\Users\boora\Downloads>
```

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.

CODE & OUTPUT

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

- File Explorer:** Shows files `aac1.py`, `aac3.1.py`, and `tr.py`.
- Run View:** A modal window titled "RUN" is open, showing options like "Run and Debug", "Debug using a terminal", and "Show automatic Python configurations".
- Code Editor:** Displays Python code for classifying even or odd numbers and calculating factorials.
- Terminal:** Shows the output of running the script `aac3.1.py`. The output includes:
 - Input: "Enter a number: 16"
 - Output: "16 is not a palindrome number."
 - Input: "Enter a number to calculate its factorial: 11"
 - Output: "The factorial of 11 is 39916800."
 - Input: "Enter a number: 28"
 - Output: "28 is not an Armstrong number."
 - Input: "Enter a number: 77"
 - Output: "The number 77 is positive."
 - Input: "Enter a number: 16"
 - Output: "16 is not a perfect number."
 - Input: "Enter an integer: 28"
 - Output: "The number 28 is even."
- Breakpoints:** A sidebar titled "BREAKPOINTS" shows checkboxes for "Raised Exceptions" (unchecked), "Uncaught Exceptions" (checked), and "User Uncaught Except..." (unchecked).
- Bottom Status Bar:** Shows "Spaces: 4", "UTF-8", "Python", "3.11.5", and a search icon.