

ASSIGNMENT – 3.4

NAME : s.harshavardhan

HT.NO : 2303A51563

BATCH : 10

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.

PROMPT : check whether the given number is palindrome or not in python.

The screenshot shows a code editor with a Python file named `Assignment_3.1.py`. The code implements a simple palindrome check without using functions. It prompts the user for a number, converts it to a string, and then compares it with its reverse using slice notation `[::-1]`.

```
# Simple palindrome check without functions
# Get number from user
num = int(input("Enter a number: "))

# Convert to string to check
num_str = str(abs(num))

# Check if palindrome by comparing with reverse
if num_str == num_str[::-1]:
    print(f"{num} is a palindrome")
else:
    print(f"{num} is not a palindrome")
```

In the top right corner, there is a Chat interface titled "PALINDROME CHECK FOR A NUMBER IN PYTHON". A message from the AI asks, "check whether the given number is palindrome or not in python". Below it, the AI responds with "I'll create a Python solution to check if a number is a palindrome". Another message shows the AI has created a file named `Assignment_3.1.py`. The AI also claims to have created a complete palindrome checker.

How it works:

- Converts the number to a string and removes the negative sign (if any)
- Compares the string with its reverse using slice notation `[::-1]`
- Returns `True` if it's a palindrome, `False` otherwise

Key features:

- Handles negative numbers (treats `-121` as `121`)
- Includes test cases to verify functionality
- Has an interactive input section to check numbers manually

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

- File Explorer:** Shows a file tree with `Assignment_3.1.py` as the active file.
- Code Editor:** Displays the following Python code:

```
1  # Simple palindrome check without functions
2
3  # Get number from user
4  num = int(input("Enter a number: "))
5
6  # Convert to string to check
7  num_str = str(abs(num))
8
9  # Check if palindrome by comparing with reverse
10 if num_str == num_str[::-1]:
11     print(f"{num} is a palindrome")
12 else:
13     print(f"{num} is not a palindrome")
```
- Terminal:** Shows command-line output for running the script in Python 3.12. The terminal pane has multiple tabs open, with the current tab showing the execution of `Assignment_3.1.py`.
- Output:** Shows standard output from the script execution.
- Search:** A search bar at the top with the query "AI Assistant Coding".
- Chat:** An AI Assistant Coding interface titled "PALINDROME CHECK FOR A NUMBER IN PYTHON". It includes a message from the AI asking "check whether the given number is palindrome or not in python".
- Status Bar:** Shows the current file is "Assignment_3.1.py", the line and column are 14, col 1, and the status bar also includes "Spaces: 4", "UTF-8", "Python", "Python 3.12 (64-bit)", and a date/time stamp "12-01-2024".

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.

PROMPT : Generate a Python function to compute the factorial of a given number.

Example:

Input: 5 → Output: 120

A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows a file tree with 'Assignment_3.1.py' selected. The main editor area contains Python code for calculating factorials. The status bar at the bottom shows 'Ln 33, Col 41' and 'Python 3.12 (64-bit)'. The bottom right corner displays the date and time: '08:07 PM 12-01-2026'. On the right side, there is a 'CHAT' panel titled 'PALINDROME CHECK FOR A NUMBER IN PYTHON'. It includes a message from 'AI Assistant Coding' asking to generate a Python function to compute the factorial of a given number, followed by an example: 'Input: 5 -> Output: 120'. Below this, a list of tasks is shown:

- ✓ Read Python file
- I'll add the factorial function to your file.
- ✓ Edited Assignment_3.1.py +25 -2

The terminal at the bottom shows the execution of the script:

```
n = int(input("Enter a number for factorial: "))
if n < 0:
    print("Factorial not defined for negative numbers")
elif n == 0 or n == 1:
    print(f"Factorial of {n} = 1")
else:
    result = 1
    for i in range(2, n + 1):
        result *= i
    print(f"Factorial of {n} = {result}")
# Example: Input 5 -> Output 120
print("\nExample: Factorial of 5 = 120")
```

Output:

```
Enter a number for factorial: 12321
--- Factorial Calculation ---
Enter a number for factorial: 10
Factorial of 10 = 3628800
Example: Factorial of 5 = 120
PS C:\Users\rishi\OneDrive\Desktop\UNI\SEM - VI\AI Assistant Coding>
```

A screenshot of the Visual Studio Code (VS Code) interface, similar to the first one but with a different configuration. The terminal output is different, showing the script being run in a PowerShell window:

```
PS C:\Users\rishi\OneDrive\Desktop\UNI\SEM - VI\AI Assistant Coding> & C:\Users\rishi\A
PS C:\Users\rishi\OneDrive\Desktop\UNI\SEM - VI\AI Assistant Coding> & C:\Users\rishi\A
ppData\Local\Programs\Python\Python312\python.exe "c:/Users/rishi/OneDrive/Desktop/UNI
SEM - VI/AI Assistant Coding/Assignment_3.1.py"
Enter a number: 12321
Enter a number: 12321
--- Factorial Calculation ---
Enter a number for factorial: 10
Factorial of 10 = 3628800
Example: Factorial of 5 = 120
PS C:\Users\rishi\OneDrive\Desktop\UNI\SEM - VI\AI Assistant Coding>
```

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.

PROMPT : 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

```

35 # TASK - III
36 # Armstrong Number check without functions
37 # Armstrong number: sum of digits raised to power of number of digits
38
39 print("\n--- Armstrong Number Check ---")
40
41 num = int(input("Enter a number: "))
42
43 # Convert to string to get digits
44 num_str = str(num)
45 num_digits = len(num_str)
46
47 # Calculate sum of each digit raised to power of total digits
48 sum_of_powers = 0
49 for digit in num_str:
50     sum_of_powers += int(digit) ** num_digits
51
52 # Check if Armstrong number
53 if sum_of_powers == num:
54     print(f"{num} is an Armstrong Number")
55 else:
56     print(f"{num} is Not an Armstrong Number")
57
58 # Examples
59 print("\nExamples:")
60 print("153 is an Armstrong Number (1^3 + 5^3 + 3^3 = 153)")
61 print("370 is an Armstrong Number (3^3 + 7^3 + 0^3 = 370")
62 print("123 is Not an Armstrong Number (1^3 + 2^3 + 3^3 = 36)")
63

```

✓ Read Python file
 ✓ Edited Assignment_3.1.py +28 -0
 Done! Added TASK - III to check Armstrong numbers without functions.

How it works:

- Gets the number of digits
- Raises each digit to that power and sums them
- If the sum equals the original number, it's an Armstrong number
- Examples: $153 = 1^3 + 5^3 + 3^3$, $370 = 3^3 + 7^3 + 0^3$

C ↻ ⌂ ⌂ 1 file changed +29 -1 Keep Undo ↻ 1 of 1 ↑ ↓ Claude Haiku 4.5 + tx

Describe what to build next
 Agent Auto 🚧

Ln 63, Col 1 Spaces:4 UTF-8 CRLF {} Python Python 3.12 (64-bit) ↻

08:09 PM 12-01-2026

```

31
32 # Example: Input 5 → Output 120
33 print("\nExample: Factorial of 5 = 120")
34
35 # TASK - III
36 # Armstrong Number check without functions
37 # Armstrong number: sum of digits raised to power of number of digits
38
39 print("\n--- Armstrong Number Check ---")
40
41 num = int(input("Enter a number: "))
42
43 # Convert to string to get digits
44 num_str = str(num)
45 num_digits = len(num_str)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + ↻ ⌂ ⌂ ... | ☰ X

PS C:\Users\rishi\OneDrive\Desktop\UNI\SEM - VI\AI Assistant Coding> & C:\Users\rishi\AI\ppData\Local\Programs\Python\Python312\python.exe "c:/Users/rishi/OneDrive/Desktop/UNI/SEM - VI/AI Assistant Coding/Assignment_3.1.py"

--- Armstrong Number Check ---
 Enter a number: 234
 234 is Not an Armstrong Number

Examples:
 153 is an Armstrong Number ($1^3 + 5^3 + 3^3 = 153$)
 370 is an Armstrong Number ($3^3 + 7^3 + 0^3 = 370$)
 123 is Not an Armstrong Number ($1^3 + 2^3 + 3^3 = 36$)

PS C:\Users\rishi\OneDrive\Desktop\UNI\SEM - VI\AI Assistant Coding>

77 ↻ ⌂ ⌂ 0 + Assignment_3.1.py
 Describe what to build next
 Agent Auto 🚧

Ln 1, Col 4 Spaces:4 UTF-8 CRLF {} Python Python 3.12 (64-bit) ↻

08:10 PM 12-01-2026

(Optional Extension)

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.

PROMPT : Write a Python program to classify a number as prime, composite, or neither. Ensure proper input validation and handle edge cases like 0, 1, and negatives. Use an optimized logic by checking factors only up to \sqrt{n} .

The screenshot shows the AI Assistant Coding interface. On the left, a code editor displays a Python file named 'Assignment_3.1.py' with the following content:

```
65 print("\n--- Prime, Composite, or Neither ---")
66
67 num = int(input("Enter a number: "))
68
69 # Handle edge cases
70 if num < 0:
71     print(f"{num} is Neither (negative numbers are neither prime nor composite)")
72 elif num == 0 or num == 1:
73     print(f"{num} is Neither (0 and 1 are neither prime nor composite)")
74 elif num == 2:
75     print(f"{num} is Prime (only even prime number)")
76 else:
77     # Check if prime by testing factors up to √n
78     is_prime = True
79     i = 2
80
81     while i * i <= num:
82         if num % i == 0:
83             is_prime = False
84             break
85         i += 1
86
87     if is_prime:
88         print(f"{num} is Prime")
89     else:
90         print(f"{num} is Composite")
91
92 # Test environment
93
```

On the right, a sidebar titled "PALINDROME CHECK FOR A NUMBER IN PYTHON" contains the following text:

Write a Python program to classify a number as prime, composite, or neither. Ensure proper input validation and handle edge cases like 0, 1, and negatives. Use an optimized logic by checking factors only up to \sqrt{n} .

Below this, a summary of the task is provided:

- ✓ Read Python assignment file
- ✓ Edited Assignment_3.1.py +37 -2

Done! Added TASK - IV with optimized prime/composite classification.

Key features:

- Edge cases handled: negatives, 0, 1 → classified as "Neither"
- Special case: 2 is Prime (only even prime)
- Optimized logic: Only checks factors up to \sqrt{n} using `while i * i <= num`
- Input validation: Classifies any number correctly as Prime, Composite, or Neither

This approach is efficient because checking up to \sqrt{n} is enough—if a number has a factor greater than \sqrt{n} , it must also have a corresponding factor less than \sqrt{n} .

At the bottom, there are buttons for "Keep", "Undo", and "Describe what to build next". The status bar at the bottom of the screen shows "Ln 100, Col 22 Spaces: 4 UTF-8 {} Python Python 3.12 (64-bit)" and the date/time "12-01-2026 08:13 PM".

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

- File Explorer:** Shows files like `Assignment_3.1.py` and `Assignment_3.1.py > ...`.
- Code Editor:** Displays the following Python code:

```

88     if is_prime:
89         print(f'{num} is Prime')
90     else:
91         print(f'{num} is Composite')
92
93 # Test examples
94 print("\nExamples:")
95 print("0 → Neither")
96 print("1 → Neither")
97 print("2 → Prime")
98 print("17 → Prime")
99 print("0 → Composite")
100 print("-5 → Neither")

```
- Terminal:** Shows the command `python Assignment_3.1.py` being run, with output:

```

PS C:\Users\rishi\OneDrive\Desktop\UNI\SEM - VI\AI Assistant Coding> & C:\Users\rishi\OneDrive\Desktop\UNI\SEM> python Assignment_3.1.py
Enter a number: 335
335 is Composite

Examples:
0 → Neither
1 → Neither
2 → Prime
17 → Prime
20 → Composite
-5 → Neither

```
- Chat Panel:** A task titled "PALINDROME CHECK FOR A NUMBER IN PYTHON" is shown with the following prompt:

Write a Python program to classify a number as prime, composite, or neither. Ensure proper input validation and handle edge cases like 0, 1, and negatives. Use an optimized logic by checking factors only up to \sqrt{n} .
- Bottom Status Bar:** Shows system information like battery level (77%), system tray icons, and the date/time (12-01-2026, 08:13 PM).

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.

PROMPT : check whether the number is perfect number or not in python.

The screenshot shows a code editor with Python code for checking if a number is perfect. The code iterates through divisors from 1 to num-1, sums them up, and compares the sum with the original number. It also includes examples for numbers 6, 28, and 10.

```

104     # Perfect number: equals the sum of its proper divisors (divisors except itself)
105
106     print("\n--- Perfect Number Check ---")
107
108     num = int(input("Enter a number: "))
109
110     if num <= 0:
111         print(f"{num} is Not a Perfect Number (must be positive)")
112     else:
113         # Find sum of all divisors except the number itself
114         sum_of_divisors = 0
115
116         for i in range(1, num):
117             if num % i == 0:
118                 sum_of_divisors += i
119
120         # Check if perfect number
121         if sum_of_divisors == num:
122             print(f"{num} is a Perfect Number")
123             print(f"Divisors of {num}: 1 to {num-1}, Sum = {sum_of_divisors}")
124         else:
125             print(f"{num} is Not a Perfect Number")
126             print(f"Sum of divisors: {sum_of_divisors}")
127
128     # Examples
129     print("\nExamples:")
130     print("6 is Perfect (1 + 2 + 3 = 6)")
131     print("28 is Perfect (1 + 2 + 4 + 7 + 14 = 28)") Keep Undo ⌘ 1 of 1 ↑ ↓
132     print("10 is Not Perfect (1 + 2 + 5 = 8)")

    
```

The AI Assistant Coding interface shows the following details:

- CHAT: PALINDROME CHECK FOR A NUMBER IN PYTHON
- check whether the number is perfect number or not in python
- Read Python file
- Edited Assignment_3.1.py +33 -0
- Done! Added TASK - V to check for perfect numbers.
- How it works:**
 - A perfect number equals the sum of its proper divisors (all divisors except itself)
 - Finds all divisors from 1 to n-1
 - Sums them and compares with the original number
 - Handles edge cases: negative numbers and 0
- Examples:**
 - 6 is Perfect (1 + 2 + 3 = 6)
 - 28 is Perfect (1 + 2 + 4 + 7 + 14 = 28)
 - 10 is Not Perfect (1 + 2 + 5 = 8)

The screenshot shows a terminal window running the Python script. The output shows the program asking for input, receiving '87', and then printing the sum of divisors as 33. Below the terminal, the AI Assistant Coding interface is visible.

```

PS C:\Users\rishi\OneDrive\Desktop\UNI\SEM - VI\AI Assistant Coding> & C:\Users\rishi\Ap
ppData\Local\Programs\Python\Python312\python.exe "c:/Users/rishi/OneDrive/Desktop/UNI/
SEM - VI\AI Assistant Coding\Assignment_3.1.py"
--- Perfect Number Check ---
Enter a number: 87
87 is Not a Perfect Number
Sum of divisors: 33

    
```

The AI Assistant Coding interface shows the following details:

- CHAT: PALINDROME CHECK FOR A NUMBER IN PYTHON
- check whether the number is perfect number or not in python
- Read Python file
- Edited Assignment_3.1.py +33 -0
- Done! Added TASK - V to check for perfect numbers.
- How it works:**
 - A perfect number equals the sum of its proper divisors (all divisors except itself)
 - Finds all divisors from 1 to n-1
 - Sums them and compares with the original number
 - Handles edge cases: negative numbers and 0
- Examples:**
 - 6 is Perfect (1 + 2 + 3 = 6)
 - 28 is Perfect (1 + 2 + 4 + 7 + 14 = 28)
 - 10 is Not Perfect (1 + 2 + 5 = 8)

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.

PROMPT : 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

A screenshot of a code editor window titled "Assignment_3.1.py". The code implements a simple function to check if a number is even or odd. It includes input validation using a try-except block and handles all cases (positive, negative, and zero). Examples are provided at the bottom of the code.

```
132 print("10 is Not Perfect (1 + 2 + 5 = 8)")  
133  
134 # TASK - VI  
135 # Even or Odd number check with input validation  
136  
137 print("\n--- Even or Odd Check ---")  
138  
try:  
    num = int(input("Enter a number: "))  
141  
    if num % 2 == 0:  
        print(f"{num} is Even")  
    else:  
        print(f"{num} is Odd")  
146  
except ValueError:  
    print("Invalid input! Please enter a valid integer.")  
149  
# Examples  
151 print("\nExamples:")  
152 print("8 -> Even")  
153 print("15 -> Odd")  
154 print("0 -> Even")  
155 print("-3 -> Odd")  
156 print("-4 -> Even")  
157  
158 |
```

The right side of the interface shows a "CHAT" panel for "PALINDROME CHECK FOR A NUMBER IN PYTHON". It displays examples and key features of the task.

Key features:

- Simple modulo check: `num % 2 == 0` for even, else odd
- Input validation:** `try-except` catches invalid inputs
- Handles all cases: positive, negative, and zero
- Examples show it works for all scenarios

Done! Added TASK - VI to check if a number is even or odd.

Agent v Auto v 90

Keep Undo 1 of 1 ↑ ↓

Ln 158, Col 1 Spaces: 4 UTF-8 {} Python Python 3.12 (64-bit) 08:15 PM 12-01-2026

A screenshot of a code editor window titled "Assignment_3.1.py". The code is identical to the one in the first screenshot. The terminal tab at the bottom shows the execution of the script and its output, confirming that the number 34 is even.

```
PS C:\Users\rishi\OneDrive\Desktop\UNI\SEM - VI\AI Assistant Coding> & C:\Users\rishi\OneDrive\Desktop\UNI\SEM - VI\AI Assistant Coding\Assignment_3.1.py  
--- Even or Odd Check ---  
Enter a number: 34  
34 is Even  
  
Examples:  
8 -> Even  
15 -> Odd  
0 -> Even  
-3 -> Odd  
-4 -> Even
```

The right side of the interface shows a "CHAT" panel for "PALINDROME CHECK FOR A NUMBER IN PYTHON". It displays examples and key features of the task.

Key features:

- Simple modulo check: `num % 2 == 0` for even, else odd
- Input validation:** `try-except` catches invalid inputs
- Handles all cases: positive, negative, and zero
- Examples show it works for all scenarios

Done! Added TASK - VI to check if a number is even or odd.

Agent v Auto v 90

Keep Undo 1 of 1 ↑ ↓

Ln 135, Col 30 Spaces: 4 UTF-8 CRLF {} Python Python 3.12 (64-bit) 08:16 PM 12-01-2026