

## AIAC LAB

### LAB 3.1

2303A51802

Billa Spurthi

Batch 29

**Task 1 Prompt:** Write a Python function that checks whether a given number is a palindrome. The function should return True if the number is a palindrome and False otherwise.

The screenshot shows the VS Code interface. The Explorer sidebar on the left lists files in the 'Lab3.1' folder, including 'Task1.py'. The 'Task1.py' file is open in the editor, containing the following code:

```
1 # Write a python function that checks whether a given number is a palindrome. The function should return True if the number is a pa
2 def is_palindrome(num):
3     num_str = str(num)
4     return num_str == num_str[::-1]
5 print(is_palindrome(121))      # True
6 print(is_palindrome(123))      # False
7 print(is_palindrome(-121))     # False
8 print(is_palindrome(10))       # False
```

The terminal tab at the bottom shows the execution of the script and its output:

```
PS C:\Users\SPURTHI\Desktop\Ai Assistant Coding> & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\Desktop\Ai Assistant Coding\Lab3.1\Task1.py'
True
False
False
False
PS C:\Users\SPURTHI\Desktop\Ai Assistant Coding>
```

### Observations:

Works correctly for positive numbers.

Negative numbers are not handled explicitly.

No input validation for non-integer values.

Converts number to string (not purely mathematical approach).

**TASK 2 Prompt:** Write a Python function to compute the factorial of a given number.

```
# Write a Python function to compute the factorial of a given number.
def factorial(n):
    if n < 0:
        return "Factorial not defined for negative numbers"
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
print(factorial(5)) # Output: 120
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> c;; cd 'c:\Users\SPURTHI\on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-wi rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab3.1\Task2.py'  
120
- PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>

## Comparison with Zero-Shot:

Zero-Shot version usually:

Might not include negative number check.

Might not include input validation.

One-Shot improvements:

Includes edge case handling (negative numbers).

Clear loop structure.

More structured output.

### TASK 3 Prompt:

Write a Python function to check whether a number is an Armstrong number.

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

- EXPLORER**: Shows a tree view of files under "AI ASSISTANT CODING". The "Task3.py" file is selected.
- CODE**: The code for the "is\_armstrong" function is displayed. It takes a number as input, calculates the sum of its digits raised to the power of their count, and returns "Armstrong Number" if they are equal, otherwise "Not an Armstrong Number". It also includes several print statements to test the function with various inputs.
- TERMINAL**: Shows the output of running the script with different inputs. The terminal tab is active.
- OUTPUT**: Shows the results of the script execution.
- DEBUG CONSOLE**: Shows the results of the script execution.
- PROBLEMS**: Shows no problems.
- PORTS**: Shows no ports.

```
2  def is_armstrong(num):
3      return "Invalid Input"
4
5      order = len(str(num))
6      total = sum(int(digit) ** order for digit in str(num))
7
8      if total == num:
9          return "Armstrong Number"
10     else:
11         return "Not an Armstrong Number"
12
13 print(is_armstrong(153)) # Armstrong Number
14 print(is_armstrong(370)) # Armstrong Number
15 print(is_armstrong(9474)) # Armstrong Number
16 print(is_armstrong(123)) # Not an Armstrong Number
17 print(is_armstrong(-10)) # Invalid Input
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab3.1\Task3.py'
Armstrong Number
Armstrong Number
Armstrong Number
Not an Armstrong Number
Invalid Input
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

### Analysis:

Multiple examples improved output formatting.

Clear return messages.

Handles invalid inputs.

More structured than zero-shot.

**Task4 Prompt:** Write an optimized Python function to classify a number as prime, composite, or neither.

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

- EXPLORER:** Shows files in the workspace, including `task1.py`, `Lab3.1` (with `Task1.py`, `Task2.py`, `Task3.py`, `Task4.py` selected), `lab5_fibonacci.py`, `lab5_filehandling.py`, `lab5_logging.py`, `lab5_login.py`, `lab5_prime.py`, `lab6task1.py`, `lab6task2.py`, `lab6task3.py`, `lab6task4.py`, `lab6task5.py`, `log.txt`, `OddEven.py`, `Prime.py`, `String.py`, and `user_activity.log`.
- CODE EDITOR:** Displays the content of `Task4.py`:

```
def classify_number(n):
    if n <= 1:
        return "Neither Prime nor Composite"
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return "Composite"
    return "Prime"
print(classify_number(2))      # Prime
print(classify_number(4))      # Composite
print(classify_number(1))      # Neither Prime nor Composite
print(classify_number(-5))     # Neither Prime nor Composite
print(classify_number(9))      # Composite
```

- TERMINAL:** Shows the command-line output of running the script:

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> c;; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab3.1\Task4.py'
Prime
Composite
Neither Prime nor Composite
Neither Prime nor Composite
Composite
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

## Comparison:

More optimized ( $O(\sqrt{n})$ ).

Proper validation.

Clearly structured logic.

Best performance among prompting strategies.

## Task 5 Prompt:

Write a Python function to check whether a given number is a perfect number.

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

- EXPLORER** sidebar: Shows files under "AI ASSISTANT CODING" including Lab3.1, Task1.py, Task2.py, Task3.py, Task4.py, Task5.py, lab5\_fibonacci.py, lab5\_filehandling.py, lab5\_logging.py, lab5\_login.py, lab5\_prime.py, lab6task1.py, lab6task2.py, lab6task3.py, lab6task4.py, lab6task5.py, log.txt, OddEven.py, Prime.py, String.py, and user\_activity.log. Task5.py is currently selected.
- CODE** tab: Displays the content of Task5.py:

```
1  # Write a Python function to check whether a given number is a perfect number.
2  def is_perfect(n):
3      if n <= 1:
4          return False
5
6      sum_divisors = 0
7      for i in range(1, n):
8          if n % i == 0:
9              sum_divisors += i
10
11     return sum_divisors == n
12 print(is_perfect(6))    # True
13 print(is_perfect(28))  # True
14 print(is_perfect(12))  # False
15 print(is_perfect(1))   # False
16
```
- PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**, **TERMINAL**, and **PORTS** tabs are visible at the bottom.
- TERMINAL** tab content:
  - PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> c;; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab3.1\Task5.py'
  - True
  - True
  - False
  - False
  - False

## Observations:

Works correctly.

Not optimized (loops till n-1).

Can be improved using square root method.

**Task 6 Prompt:** Write a Python program to determine whether a given number is

even or odd.

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

- EXPLORER** sidebar: Shows files under "AI ASSISTANT CODING" including Lab3.1, Task1.py, Task2.py, Task3.py, Task4.py, Task5.py, and Task6.py (selected).
- CODE** tab: Displays a Python script named "Task6.py". The code defines a function "check\_even\_odd" that takes an integer "n" and returns "Even" if n % 2 == 0, "Odd" if n % 2 != 0, and "Invalid Input" if n is not an integer. It then prints the results for various test cases.
- TERMINAL**: Shows command-line output from a PowerShell session (PS) running on Windows. The user navigates to the directory "C:\Users\SPURTHI\Desktop\Ai Assistant Coding\Lab3.1\Task6.py" and runs it. The output shows the program's responses to different inputs: Even, Odd, Even, Even, Invalid Input, and Invalid Input.

```
Lab3.1 > Task6.py > ...
1 # Write a Python program to determine whether a given number is even or odd.
2 def check_even_odd(n):
3     if not isinstance(n, int):
4         return "Invalid Input"
5
6     if n % 2 == 0:
7         return "Even"
8     else:
9         return "Odd"
10 print(check_even_odd(8))      # Even
11 print(check_even_odd(15))    # Odd
12 print(check_even_odd(0))     # Even
13 print(check_even_odd(-4))   # Even
14 print(check_even_odd(3.5))  # Invalid Input
15 print(check_even_odd("10")) # Invalid Input
16
```

```
PS C:\Users\SPURTHI\Desktop\Ai Assistant Coding> cd 'c:\Users\SPURTHI\Desktop\Ai Assistant Coding\Lab3.1\Task6.py'
Even
Odd
Even
Even
Invalid Input
Invalid Input
PS C:\Users\SPURTHI\Desktop\Ai Assistant Coding>
```

## Analysis:

Examples improved output clarity.

Proper input validation included.

Handles negative numbers correctly.

Better structured than zero-shot.

