

AIAC LAB

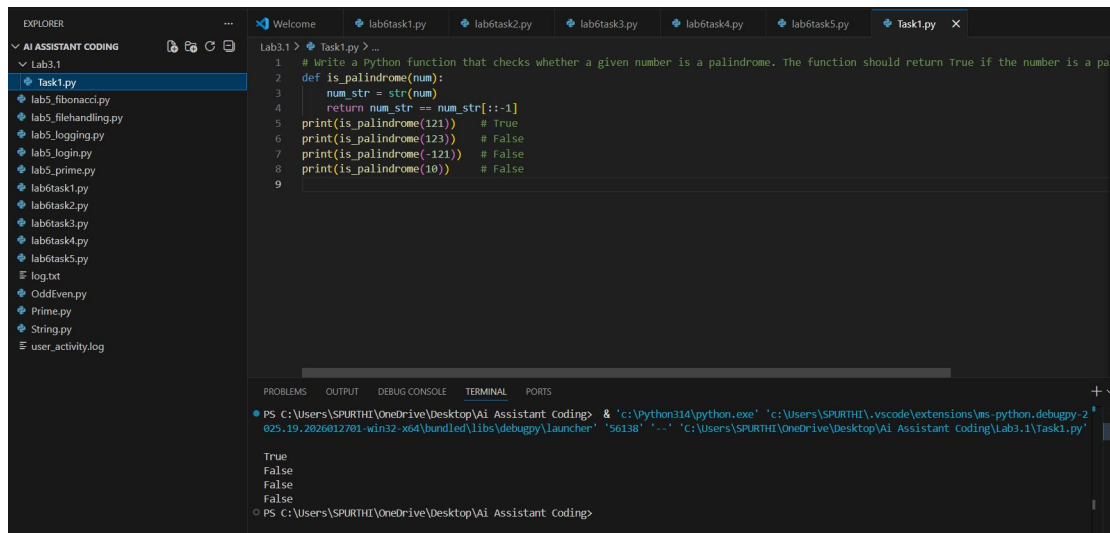
LAB 3.1

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 a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'AI Assistant Coding' with a subfolder 'Lab3.1' containing several files. The code editor shows a Python file named 'Task1.py' with 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
9
```

The bottom panel of the editor shows the 'TERMINAL' output, which displays the results of the function calls:

```
PS C:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding> & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2
025.19.2026012701-win32-x64\bundle\libs\debugpy\launcher' '56138' '-' 'c:\Users\SPURTHI\OneDrive\Desktop\AI Assistant Coding\Lab3.1\Task1.py'
True
False
False
False
```

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.

The screenshot shows a VS Code editor interface. On the left, the Explorer sidebar is open, showing a project named 'AI ASSISTANT CODING' with a subfolder 'Lab3.1'. Inside 'Lab3.1', there are files 'Task1.py' and 'Task2.py', with 'Task2.py' selected. Below 'Lab3.1', a list of other files is visible: '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'. The main editor area shows the code for 'Task2.py':

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

The bottom panel of the editor shows the 'TERMINAL' tab with the following output:

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> c::; cd 'c:\Users\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:

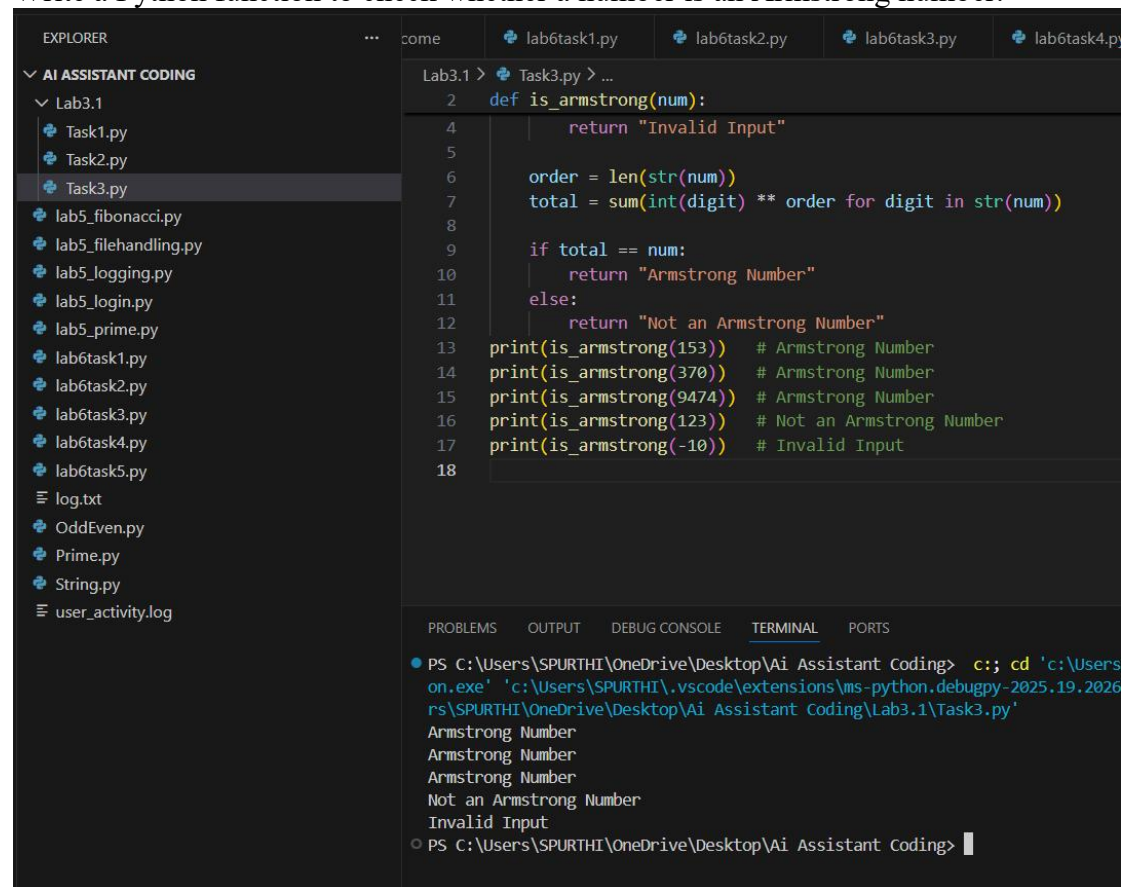
- Includes edge case handling (negative numbers).

- 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 a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'AI ASSISTANT CODING' with a subfolder 'Lab3.1' containing files 'Task1.py', 'Task2.py', and 'Task3.py'. The code editor shows the content of 'Task3.py'.

```
Lab3.1 > Task3.py > ...
2 def is_armstrong(num):
4     return "Invalid Input"
5
6     order = len(str(num))
7     total = sum(int(digit) ** order for digit in str(num))
8
9     if total == num:
10        return "Armstrong Number"
11    else:
12        return "Not an Armstrong Number"
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
18
```

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

```
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.

```
task1.py  lab6task2.py  lab6task3.py  lab6task4.py  lab6task5.py  ...
EXPLORER
  AI ASSISTANT CODING
    Lab3.1
      Task1.py
      Task2.py
      Task3.py
      Task4.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
      user_activity.log
  task1.py
  Lab3.1 > Task4.py > ...
4  def classify_number(n):
7
8      if n <= 1:
9          return "Neither Prime nor Composite"
10
11         for i in range(2, int(math.sqrt(n)) + 1):
12             if n % i == 0:
13                 return "Composite"
14
15         return "Prime"
16 print(classify_number(2)) # Prime
17 print(classify_number(4)) # Composite
18 print(classify_number(1)) # Neither Prime nor Composite
19 print(classify_number(-5)) # Neither Prime nor Composite
20 print(classify_number(9)) # Composite
21
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab3.1\Task4.py'
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32\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 a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'AI ASSISTANT CODING' with a subfolder 'Lab3.1' containing several Python files. The code editor shows the content of 'Task5.py' in 'Lab3.1'. The code defines a function 'is_perfect(n)' that checks if a number is perfect by summing its divisors. The terminal at the bottom shows the command to run the script and its output.

```
EXPLORER
... b6task2.py lab6task3.py lab6task4.py lab6task5.py Task1.py Task2
AI ASSISTANT CODING
  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
    user_activity.log
Lab3.1 > 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 PORTS
● 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
○ PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

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.

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

Analysis:

Examples improved output clarity.

Proper input validation included.

Handles negative numbers correctly.

Better structured than zero-shot.

