

Assignment 3.1

AI Assisted Coding

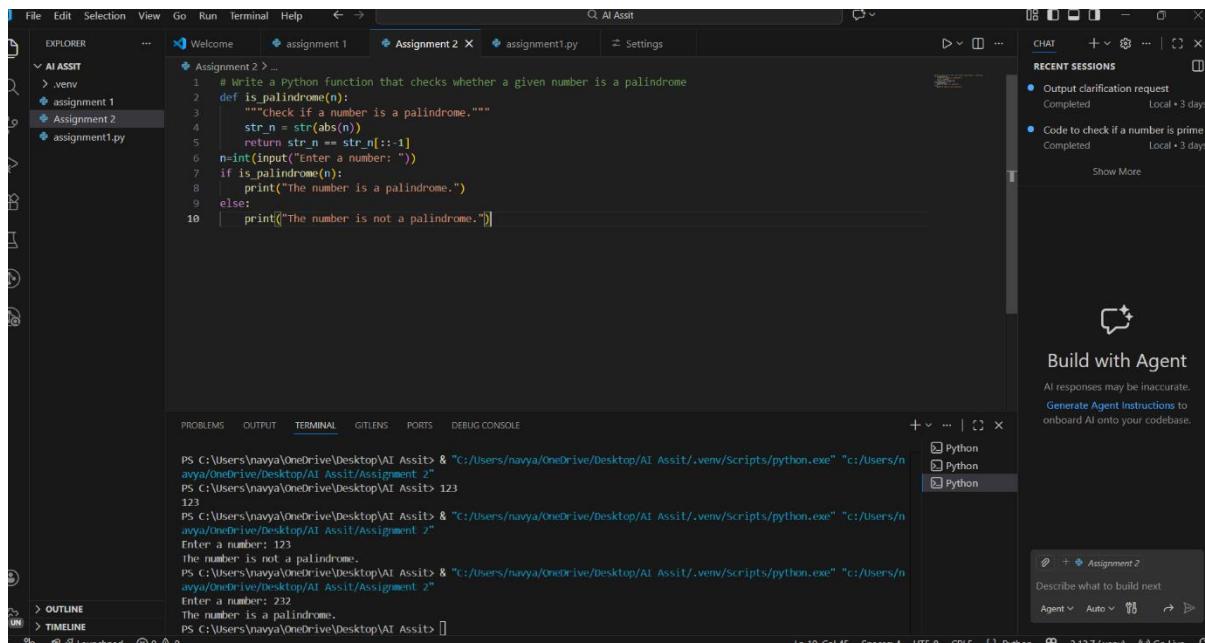
Name: Sunny Kolluri

HTNO: 2303A52444

Task 1:

Prompt:

Write a Python function that checks whether a given number is a palindrome



The screenshot shows a code editor interface with a dark theme. The top navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. On the left is an Explorer sidebar with a tree view showing 'AI ASSIT' (selected), '.env', 'assignment 1', 'Assignment 2' (selected), and 'assignment1.py'. The main workspace contains a code editor tab titled 'Assignment 2' with the following Python code:

```
1 # Write a Python function that checks whether a given number is a palindrome
2 def is_palindrome(n):
3     """Check if a number is a palindrome."""
4     str_n = str(abs(n))
5     return str_n == str_n[::-1]
6 n=int(input("Enter a number: "))
7 if is_palindrome(n):
8     print("The number is a palindrome.")
9 else:
10    print("The number is not a palindrome.")
```

Below the code editor is a terminal window showing command-line interactions:

```
PS C:\Users\navya\OneDrive\Desktop\AI Assit> & "C:/Users/navya/OneDrive/Desktop/AI Assit/.venv/Scripts/python.exe" "c:/Users/navya/OneDrive/Desktop/AI Assit/Assignment 2"
PS C:\Users\navya\OneDrive\Desktop\AI Assit> 123
PS C:\Users\navya\OneDrive\Desktop\AI Assit> & "C:/Users/navya/OneDrive/Desktop/AI Assit/.venv/Scripts/python.exe" "c:/Users/navya/OneDrive/Desktop/AI Assit/Assignment 2"
Enter a number: 123
The number is not a palindrome.
PS C:\Users\navya\OneDrive\Desktop\AI Assit> & "C:/Users/navya/OneDrive/Desktop/AI Assit/.venv/Scripts/python.exe" "c:/Users/navya/OneDrive/Desktop/AI Assit/Assignment 2"
Enter a number: 232
The number is a palindrome.
PS C:\Users\navya\OneDrive\Desktop\AI Assit>
```

The right side of the interface features a 'RECENT SESSIONS' panel with two entries: 'Output clarification request' (Completed, Local, 3 days) and 'Code to check if a number is prime' (Completed, Local, 3 days). Below it is a 'Build with Agent' section with a message: 'AI responses may be inaccurate.' and a 'Generate Agent Instructions' button.

Observation:

- The program checks whether a given number is a palindrome by converting it into a string.
- The `abs(n)` function ensures that negative numbers are handled correctly.
- The string is compared with its reverse (`str_n[::-1]`).
- If both are the same, the number is identified as a palindrome; otherwise, it is not a palindrome.

- This approach is easy to implement, efficient, and avoids complex mathematical operations.

Task 2:

Prompt: #Input: 5 → Output: 120

#Write a Python function to calculate factorial of a number.

```

10 |     print("The number is not a palindrome.")
11 | #Input: 5 → Output: 120
12 | #Write a Python function to calculate factorial of a number.
13 | def factorial(n):
14 |     if n == 0:
15 |         return 1
16 |     else:
17 |         return n * factorial(n-1)
18 | n=int(input("Enter a number: "))
19 | print(factorial(n))

```

The terminal output shows a syntax error:

```

PS C:\Users\navya\OneDrive\Desktop\AI Assit> & "c:/Users/navya/OneDrive/Desktop/AI Assit/.venv/Scripts/python.exe" "c:/Users/navya/OneDrive/Desktop/AI Assit/Assignment_2"
SyntaxError: invalid syntax
PS C:\Users\navya\OneDrive\Desktop\AI Assit> & "c:/Users/navya/OneDrive/Desktop/AI Assit/.venv/Scripts/python.exe" "c:/Users/navya/OneDrive/Desktop/AI Assit/Assignment_2"
Enter a number: 123
The number is not a palindrome.
PS C:\Users\navya\OneDrive\Desktop\AI Assit> & "c:/Users/navya/OneDrive/Desktop/AI Assit/.venv/Scripts/python.exe" "c:/Users/navya/OneDrive/Desktop/AI Assit/Assignment_2"
Enter a number: 232
The number is a palindrome.
Enter a number: 5
Enter a number: 5
120
PS C:\Users\navya\OneDrive\Desktop\AI Assit>

```

Observation:

The program calculates the **factorial of a number** using **recursion**.

The base case $n == 0$ returns 1, which stops the recursive calls. For values greater than 0, the function calls itself with $n-1$.

The result is obtained by multiplying all numbers from n down to 1.

This method clearly demonstrates the **concept of recursion** in Python.

Task3:

Prompt: #Input: 153 → Output: Armstrong Number

#Input: 370 → Output: Armstrong Number

#Input: 123 → Output: Not an Armstrong Number

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

```

18 n=int(input("Enter a number: "))
19 print(factorial(n))
20 #Input: 153 -> Output: Armstrong Number
21 #Input: 370 -> Output: Armstrong Number
22 #Input: 123 -> Output: Not an Armstrong Number
23 #Write a Python function to check whether a number is an Armstrong number.
24 def is_armstrong(n):
25     str_n = str(n)
26     num_digits = len(str_n)
27     sum_of_powers = sum(int(digit) ** num_digits for digit in str_n)
28     return sum_of_powers == n
29 n=int(input("Enter a number: "))
30 if is_armstrong(n):
31     print("The number is an Armstrong number.")
32 else:
33     print("The number is not an Armstrong number.")

```

PROBLEMS OUTPUT TERMINAL GITLENS PORTS DEBUG CONSOLE

Enter a number: 232
The number is a palindrome.
Enter a number: 12
479001600
PS C:\Users\navya\OneDrive\Desktop\AI Assit> & "C:/Users/navya/OneDrive/Desktop/AI Assit/.venv/scripts/python.exe" "c:/Users/navya/OneDrive/Desktop/AI Assit/Assignment 2"
Enter a number: 232
The number is a palindrome.
Enter a number: 5
125
Enter a number: 153
The number is an Armstrong number.
PS C:\Users\navya\OneDrive\Desktop\AI Assit>

+ ⌂ ⌂ ⌂ Launchpad ⌂ 0 ⌂ 0

Code to check if a number is prime
Completed Local 4 days
Show More

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

Assignment 2
Describe what to build next
Agent Auto Go Live

Observation:

- The program checks whether a given number is an Armstrong number.
- The number is first converted into a string to count the total number of digits.
- Each digit is raised to the power of the total number of digits and added together.
- If the sum of these powers is equal to the original number, it is identified as an Armstrong number.
- This approach is straightforward and avoids complex calculations.

Task 4:

Prompt:

#Write an optimized Python program to classify a number as Prime, Composite, or Neither.

#Constraints: Validate input Handle numbers less than or equal to 1 Use efficient logic

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there are two projects: 'assignment1' and 'assignment2'. The 'assignment2' project is currently selected. In the main editor area, a Python file named 'assignment1.py' contains the following code:

```
34 #Write an optimized Python program to classify a number as Prime,Composite, or Neither.
35 #Constraints: Validate input Handle numbers less than or equal to 1 Use efficient logic
36 def is_prime(n):
37     if n <= 1:
38         return False
39     if n <= 3:
40         return True
41     if n % 2 == 0 or n % 3 == 0:
42         return False
43     i = 5
44     while i * i <= n:
45         if n % i == 0 or n % (i + 2) == 0:
46             return False
47         i += 6
48     return True
49
50 n = int(input("Enter a number: "))
51 if is_prime(n):
52     print("The number is a Prime number.")
53 elif n > 1:
54     print("The number is a Composite number.")
55 else:
56     print("The number is Neither Prime nor Composite.")
```

In the Terminal tab, the output of running the script is shown:

```
Enter a number: 6
The number is a Composite number.
PS C:\Users\navya\OneDrive\Desktop\AI Assists:
```

The status bar at the bottom indicates the code is in Python mode, has 3.13.7 (venv) installed, and is connected to a Go Live session.

Observation:

The program classifies a given number as Prime, Composite, or Neither. Numbers less than or equal to 1 are correctly identified as Neither prime nor composite.

The function uses an optimized prime-checking logic by testing divisibility only up to \sqrt{n} . It skips unnecessary checks by eliminating multiples of 2 and 3 and then checking numbers of the form $6k \pm 1$.

This approach improves efficiency and reduces execution time for large numbers.

Task 5:

Prompt:

#Write a Python function that checks whether a given number is a perfect number.

The screenshot shows the Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. The left sidebar has sections for Explorer, AI Assist, and Outline. The main workspace contains several tabs: Welcome, assignment1, Assignment 2 (which is active), assignment1.py, and Settings. The code editor displays a Python script named 'Assignment 2' with the following content:

```

57  #Write a Python function that checks whether a given number is a perfect number.
58  def is_perfect(n):
59      if n <= 1:
60          return False
61      divisors = [i for i in range(1, n) if n % i == 0]
62      return sum(divisors) == n
63  n=int(input("Enter a number: "))
64  if is_perfect(n):
65      print("The number is a perfect number.")
66  else:
67      print("The number is not a perfect number.")

```

The terminal tab is selected, showing the output of running the script:

```

The number is a Composite number.
Enter a number: 67
The number is not a perfect number.
PS C:\Users\navya\OneDrive\Desktop\AI Assist> []

```

The right sidebar includes sections for Chat, Recent Sessions, Build with Agent, and a task pane for Assignment 2.

Observation:

- The program checks whether a given number is a **perfect number**.
- Numbers less than or equal to **1** are immediately excluded.
- All **proper divisors** of the number (excluding the number itself) are collected.
- The sum of these divisors is compared with the original number.
- If both are equal, the number is identified as a **perfect number**.

Task 6:

Prompt:

#Input: 8 → Output: Even

#Input: 15 → Output: Odd

#Input: 0 → Output: Even

#Write a Python program to determine whether a number is even or odd with proper input validation.

The screenshot shows a dark-themed instance of Visual Studio Code. In the top navigation bar, the tabs 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help' are visible. The active tab is 'Assignment 2'. Below the tabs, the code editor displays a Python script named 'Assignment 2.py'. The script contains the following code:

```
68 #Input: 8 -> Output: Even
69 #Input: 15 -> Output: Odd
70 #Input: 0 -> Output: Even
71 #Write a Python program to determine whether a number is even or odd with proper input validation.
72 def is_even():
73     return n % 2 == 0
74 n=int(input("Enter a number: "))
75 if is_even(n):
76     print("The number is Even.")
77 else:
78     print("The number is odd.")
```

Below the code editor, the 'TERMINAL' tab is selected, showing the output of running the script:

```
PS C:\Users\nayya\OneDrive\Desktop\AI Assist> Enter a number: 2
The number is Even.
PS C:\Users\nayya\OneDrive\Desktop\AI Assist>
```

In the bottom right corner, there is a 'Build with Agent' panel. It includes a text input field with placeholder text 'Describe what to build next', a dropdown for 'Agent', and a 'Python' button. The status bar at the bottom indicates 'Ln 78, Col 32', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', '3.13.7 (venv)', and 'Go Live'.

Observation:

- The program determines whether a given number is even or odd using the modulo (%) operator.
- If a number gives a remainder of 0 when divided by 2, it is classified as Even.
- Otherwise, the number is classified as Odd.
- The program correctly identifies 0 as an even number.
- This method is efficient and works for all integer inputs.