AI Assisted Coding Lab 2

Shivani Pambidi

2303A52172

Batch – 41

Question 1:

Task 1: Book Class Generation

Prompt:

Design a Python program using classes where a class named Book stores information about a book.

Code:

```
class Book:
    def __init__(self, title, author, description):
        self.title = title
        self.author = author
        self.description = description

    def summary(self):
        return f"The book {self.title} is written by {self.author} and is about {self.description}"

book1 = Book("The Great Gatsby", "F. Scott Fitzgerald", "A story of love and tragedy")
print(book1.summary())
```

Output:

```
PS C:\Users\pambi\OneDrive\Documents\Desktop\AI-Assited-Coding
> & C:\Users\pambi\AppData\Local\Programs\Python\Python313\pyt
hon.exe "c:/Users/pambi/OneDrive/Documents/Desktop/AI-Assited-
Coding/Lab 2.py"
The book The Great Gatsby is written by F. Scott Fitzgerald an
d is about a story of love and tragedy
```

Justification:

This program helps manage book details in a simple way. It uses a Book class to store the book name and author. The constructor sets the values properly when the object is created. The summary method shows the book information in a clear sentence. The code is written neatly and is easy for beginners to understand. This class can be used again in other projects. The code generated by Cursor AI is organized and beginner-friendly.

Question 2:

Task 2: Sorting Dictionaries with AI

Prompt:

Write a Python program that sorts a list of dictionaries using a given key.

Code:

Cursor AI:

```
def sort_list_of_dictionaries(list_of_dictionaries, key):
    return sorted(list_of_dictionaries, key=lambda x: x[key])
records = [
    {"name": "John", "age": 30},
    {"name": "Jane", "age": 25},
    {"name": "Jim", "age": 35},
]
print(sort_list_of_dictionaries(records, "age"))
```

Gemini:

Gemini

```
list_of_dicts = [
    {'name': 'Alice', 'age': 30},
    {'name': 'Bob', 'age': 25},
    {'name': 'Charlie', 'age': 35}
]

# Sort the list of dictionaries by the 'age' key
sorted_list = sorted(list_of_dicts, key=lambda x: x['age'])

print("Original list:", list_of_dicts)
print("Sorted list by 'age':", sorted_list)

# You can also sort in descending order
sorted_list_desc = sorted(list_of_dicts, key=lambda x: x['age'], reverse=True)
print("Sorted list by 'age' (descending):", sorted_list_desc)
```

Output:

Cursor AI:

```
[{'name': 'Jane', 'age': 25}, {'name': 'John', 'age': 30}, {'name': 'Jim', 'age': 35}]
```

Gemini:

```
Original list: [{'name': 'Alice', 'age': 30}, {'name': 'Bob', 'age': 25}, {'name': 'Charlie', 'age': 35}]
Sorted list by 'age': [{'name': 'Bob', 'age': 25}, {'name': 'Alice', 'age': 30}, {'name': 'Charlie', 'age': 35}]
Sorted list by 'age' (descending): [{'name': 'Charlie', 'age': 35}, {'name': 'Alice', 'age': 30}, {'name': 'Bob', 'age': 25}]
```

Justification:

The Cursor AI code uses a function, so it can be used again in other programs.You can sort the data using any key, which makes it flexible.The code is short, clean, and works fast. It is suitable for big or real projects.

The Gemini AI code explains each step clearly, which helps beginners understand better. It shows both ascending and descending sorting in a simple way.The code is easy to read but

is written directly in the main program.It is good for learning but not as reusable as the Cursor AI code.

Question 3:

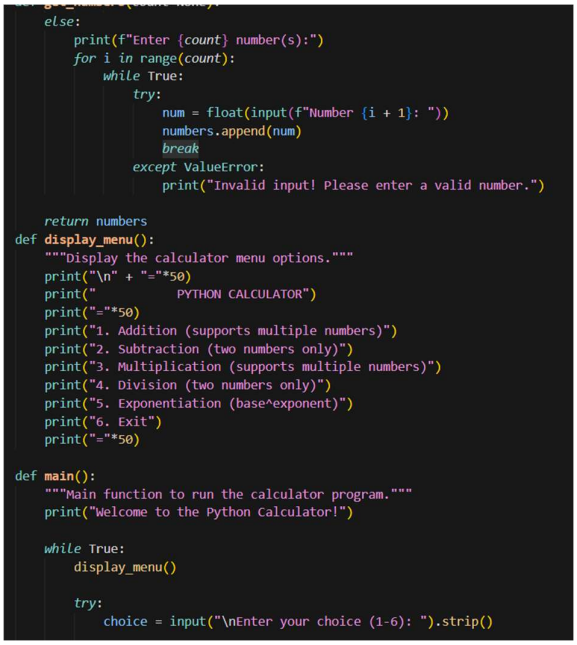
Task 3: Calculator Using Functions

Prompt:

Generate a simple calculator program using which include 5 methods for addition, subtraction, multiplication, division and exponention.For addition and multiplication take list of numbers based on user choice.

Code:

```
def add(*numbers):
    """Add multiple numbers together."""
    return sum(numbers)
def subtract(num1, num2):
    """Subtract second number from first number."""
    return num1 - num2
def multiply(*numbers):
    """Multiply multiple numbers together."""
    result = 1
    for num in numbers:
        result *= num
    return result
def divide(num1, num2):
    """Divide first number by second number."""
    if num2 == 0:
        raise ValueError("Error: Division by zero is not allowed!")
    return num1 / num2
def power(base, exponent):
    """Raise base to the power of exponent."""
    return base ** exponent
def get_numbers(count=None):
    """Get numbers from user input."""
    numbers = []
    if count is None:
        print("Enter numbers (press Enter without input when done):")
        while True:
            try:
                user_input = input("Number: ").strip()
                if user_input == "":
                    if len(numbers) < 2:
                        print("Please enter at least 2 numbers.")
                        continue
                    break
                numbers.append(float(user_input))
            except ValueError:
                print("Invalid input! Please enter a valid number.")
```



```
    else:
        print(f"Enter {count} number(s):")
        for i in range(count):
            while True:
                try:
                    num = float(input(f"Number {i + 1}: "))
                    numbers.append(num)
                    break
                except ValueError:
                    print("Invalid input! Please enter a valid number.")

    return numbers
def display_menu():
    """Display the calculator menu options."""
    print("\n" + "="*50)
    print("          PYTHON CALCULATOR")
    print("="*50)
    print("1. Addition (supports multiple numbers)")
    print("2. Subtraction (two numbers only)")
    print("3. Multiplication (supports multiple numbers)")
    print("4. Division (two numbers only)")
    print("5. Exponentiation (base^exponent)")
    print("6. Exit")
    print("="*50)

def main():
    """Main function to run the calculator program."""
    print("Welcome to the Python Calculator!")

    while True:
        display_menu()

        try:
            choice = input("\nEnter your choice (1-6): ").strip()
```

```python
def main():
            if choice == "1":  # Addition
                numbers = get_numbers()
                result = add(*numbers)
                print(f"\nResult: {' + '.join(map(str, numbers))} = {result}")

            elif choice == "2":  # Subtraction
                numbers = get_numbers(2)
                result = subtract(numbers[0], numbers[1])
                print(f"\nResult: {numbers[0]} - {numbers[1]} = {result}")

            elif choice == "3":  # Multiplication
                numbers = get_numbers()
                result = multiply(*numbers)
                print(f"\nResult: {' * '.join(map(str, numbers))} = {result}")

            elif choice == "4":  # Division
                numbers = get_numbers(2)
                try:
                    result = divide(numbers[0], numbers[1])
                    print(f"\nResult: {numbers[0]} ÷ {numbers[1]} = {result}")
                except ValueError as e:
                    print(f"\n{e}")

            elif choice == "5":  # Exponentiation
                numbers = get_numbers(2)
                result = power(numbers[0], numbers[1])
                print(f"\nResult: {numbers[0]} ^ {numbers[1]} = {result}")

            elif choice == "6":  # Exit
                print("\nThank you for using the Python Calculator. Goodbye!")
                break

            else:
                print("\nInvalid choice! Please enter a number between 1 and 6.")
```

```python
    def main():

            elif choice == "6":  # Exit
                print("\nThank you for using the Python Calculator. Goodbye!")
                break

            else:
                print("\nInvalid choice! Please enter a number between 1 and 6.")

        except KeyboardInterrupt:
            print("\n\nProgram interrupted by user. Goodbye!")
            break
        except Exception as e:
            print(f"\nAn unexpected error occurred: {e}")

        # Ask if user wants to continue
        if choice in ["1", "2", "3", "4", "5"]:
            continue_choice = input("\nDo you want to perform another operation? (y/n)
            if continue_choice != 'y':
                print("\nThank you for using the Python Calculator. Goodbye!")
                break
if __name__ == "__main__":
    main()
```

Output:

```
Problems  Output  Debug Console  Terminal  Ports
PS C:\Users\pambi\OneDrive\Documents\Desktop\AI-Assited-Coding> & C:\Users\pamb
i\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/pambi/OneDrive/D
ocuments/Desktop/AI-Assited-Coding/Lab 2.py"
Welcome to the Python Calculator!

========================================
        PYTHON CALCULATOR
========================================
1. Addition (multiple numbers)
2. Subtraction (two numbers)
3. Multiplication (multiple numbers)
4. Division (two numbers)
5. Exponentiation
6. Exit
========================================
Enter your choice (1-6): 1
Enter numbers (press Enter without input to finish):
Number: 2
Number: 3
Number: 4
Number: 5
Number: 1
Number: 2
Number:
Result: 17.0
Do you want to continue? (y/n): y

========================================
        PYTHON CALCULATOR
========================================
```

```
========================================
        PYTHON CALCULATOR
========================================
1. Addition (multiple numbers)
2. Subtraction (two numbers)
3. Multiplication (multiple numbers)
4. Division (two numbers)
5. Exponentiation
6. Exit
========================================
Enter your choice (1-6): 2
Number 1: 2
Number 2: 1
Result: 1.0
Do you want to continue? (y/n): y

========================================
        PYTHON CALCULATOR
========================================
1. Addition (multiple numbers)
2. Subtraction (two numbers)
3. Multiplication (multiple numbers)
4. Division (two numbers)
5. Exponentiation
6. Exit
========================================
Enter your choice (1-6): 3
Enter numbers (press Enter without input to finish):
Number: 2
Number: 2
```

```
Number: 2
Number: 2
Number: 2
Number:
Result: 8.0
Do you want to continue? (y/n): y

========================================
        PYTHON CALCULATOR
========================================
1. Addition (multiple numbers)
2. Subtraction (two numbers)
3. Multiplication (multiple numbers)
4. Division (two numbers)
5. Exponentiation
6. Exit
========================================
Enter your choice (1-6): 4
Number 1: 6
Number 2: 3
Result: 2.0
Do you want to continue? (y/n): y

========================================
        PYTHON CALCULATOR
========================================
1. Addition (multiple numbers)
2. Subtraction (two numbers)
3. Multiplication (multiple numbers)
4. Division (two numbers)
```

```
========================================
        PYTHON CALCULATOR
========================================
1. Addition (multiple numbers)
2. Subtraction (two numbers)
3. Multiplication (multiple numbers)
4. Division (two numbers)
5. Exponentiation
6. Exit
========================================
Enter your choice (1-6): 5
Number 1: 3
Number 2: 5
Result: 243.0
Do you want to continue? (y/n):
Thank you for using the Python Calculator. Goodbye!
PS C:\Users\pambi\OneDrive\Documents\Desktop\AI-Assited-Coding>
```

Justification:

The program is divided into small functions, so each task is handled separately.This makes the calculator simple to understand and easy to modify.It allows adding and multiplying more than two numbers at a time.The instructions shown to the user are clear, which makes the program easy to use.The program keeps running until the user chooses to exit.Overall, the code is neat, works well, and is easy to manage later.

Question 4

Task 4: Armstrong Number Optimization

Prompt:

Create a Python program that takes a number from the user and checks if it is an Armstrong number.The program should print a message saying whether the entered number is an Armstrong number or not.

Code:

Gemini:

```
num_str = input("Enter a number: ")

if not num_str.isdigit():
    print("Invalid input. Please enter a positive integer.")
else:
    num = int(num_str)
    order = len(num_str)
    sum_of_powers = 0
    temp_num = num

    while temp_num > 0:
        digit = temp_num % 10
        sum_of_powers += digit ** order
        temp_num //= 10

    if num == sum_of_powers:
        print(f"{num} is an Armstrong number.")
    else:
        print(f"{num} is not an Armstrong number.")
```

Cursor AI:

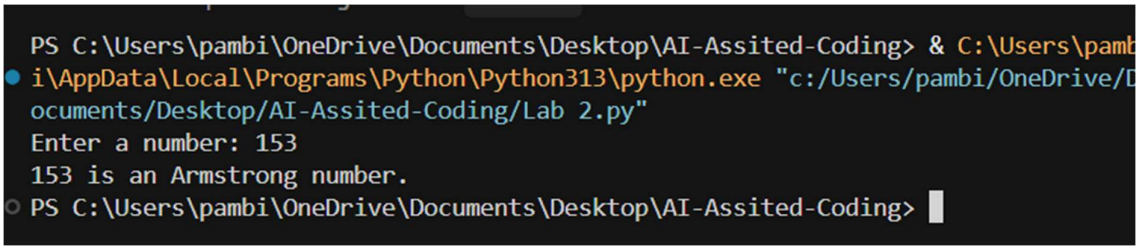
```
LAB-2.py > ...
def is_armstrong_number(num: int) -> bool:

    An n-digit number is an Armstrong number if:
        sum(digit^n for each digit in the number) == number itself
    """
    if num < 0:
        # By common definition, negative numbers are not Armstrong numbers
        return False

    digits_str = str(num)
    num_digits = len(digits_str)

    # Compute the sum of each digit raised to the power of num_digits
    armstrong_sum = 0
    for ch in digits_str:
        digit = ord(ch) - ord('0')  # Faster than int(ch) for single digits
        armstrong_sum += digit ** num_digits

    return armstrong_sum == num


def main():
    try:
        user_input = int(input("Enter a number: ").strip())
    except ValueError:
        print("Invalid input. Please enter an integer.")
        return

    if is_armstrong_number(user_input):
        print(f"{user_input} is an Armstrong number.")
    else:
        print(f"{user_input} is NOT an Armstrong number.")


if __name__ == "__main__":
    main()
```

Outputs:

Gemini:

```
Enter a number: 153
153 is an Armstrong number.
```

Cursor AI:



Justification:

It checks user input carefully, which helps prevent errors while running.The main work is placed inside a function, so the code can be reused.Unnecessary steps are removed, making the program shorter and cleaner.Good coding rules are followed throughout the program.The names of variables and functions are easy to understand.This makes finding mistakes and updating the code much simpler.Overall, the program runs smoothly and is more dependable.