ASSIGNMENT-2

Gujja Pranitha

2303A52171

BATCH-41

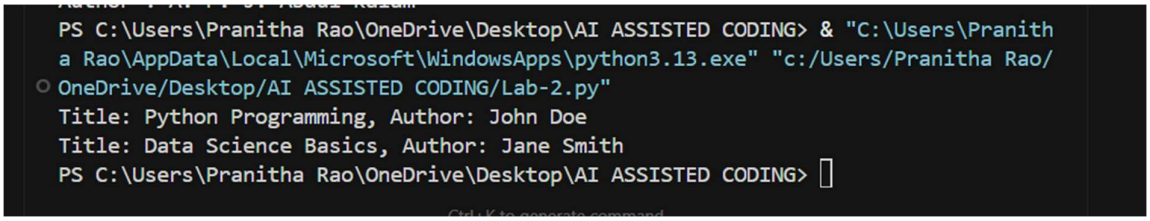
Task 1: Book Class Generation

Use Cursor AI to generate a Python class Book with attributes title, author, and a summary() method.

PROMPT: Create a simple library management module by writing a Python class named Book that includes the attributes title and author, and a method summary() to display the book details.

CODE:

OUTPUT:



JUSTIFICATION:

The Book class uses the __init__ method to save the title and author for each book object. This means every book keeps its own information. The summary() method shows the book details in a clear format. This makes the code easy to understand, simple to maintain, and easy to expand for a library management system.

Task 2: Sorting Dictionaries with AI

Use Gemini and Cursor AI to generate code that sorts a list of dictionaries by a key.

PROMPT: using gemini

Write a Python program that sorts a list of dictionaries basedon

a specified key.

CODE:

```
# Example list of dictionaries
data = [
    {'name': 'Alice', 'age': 30},
    {'name': 'Bob', 'age': 25},
    {'name': 'Charlie', 'age': 35}
]

# Sort the list by the 'age' key
sorted_data = sorted(data, key=lambda x: x['age'])

# Display the sorted data
print(sorted_data)
```

OUTPUT:

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

PROMPT: (cursor ai)

Write an efficient and readable Python function that sorts a list of dictionaries based on a given key, avoiding unnecessary variables and computations.

CODE:

```python
def display(title, data):
    print(title)
    print("-" * 45)
    for d in data:
        print(f"Name: {d['name']:<8} | Age: {d['age']:<2} | Score: {d['score']}")
    print("-" * 45, "\n")
students = [
    {"name": "Alice", "age": 27, "score": 90},
    {"name": "Bob", "age": 32, "score": 75},
    {"name": "Charlie", "age": 20, "score": 88}
]
display(
    "Sorted by age (ascending)",
    sorted(students, key=lambda x: x["age"])
)
display(
    "Sorted by score (descending)",
    sorted(students, key=lambda x: x["score"], reverse=True)
)
```

OUTPUT:

```
PS C:\Users\Pranitha Rao\OneDrive\Desktop\AI ASSISTED CODING> & "C:\Users\Pranitha Rao\AppData\Local\Microsoft\WindowsApps\python3.13.exe" "c:/Users/Pranitha Rao/OneDrive/Desktop/AI ASSISTED CODING/Lab-2.py"
Sorted by age (ascending)
---------------------------------------------
Name: Charlie  | Age: 20 | Score: 88
Name: Alice    | Age: 27 | Score: 90
Name: Bob      | Age: 32 | Score: 75
---------------------------------------------

Sorted by score (descending)
---------------------------------------------
Name: Alice    | Age: 27 | Score: 90
Name: Charlie  | Age: 20 | Score: 88
Name: Bob      | Age: 32 | Score: 75
---------------------------------------------

PS C:\Users\Pranitha Rao\OneDrive\Desktop\AI ASSISTED CODING>
```

JUSTIFICATION:

Both programs sort the data in the same way and work at the same speed. The main difference is how the code is written. The first program is short and easy, so beginners can quickly understand it. The second program is a bit longer but better organized. It uses functions, can sort by different keys, and

displays the output neatly. The first one is good for small examples, while the second one is more useful and clearer for larger programs.

Task 3: Calculator Using Functions

PROMPT:

Write a Python program to create a simple calculator that includes methods for addition, subtraction, multiplication, division, and exponentiation. For addition and multiplication, allow the user to perform the operation on a list of numbers based on their choice.

CODE:

```
Lab-2.py > ...
class Calculator:
    def addition(self, numbers):
        return sum(numbers)
    def subtraction(self, a, b):
        return a - b
    def multiplication(self, numbers):
        result = 1
        for num in numbers:
            result *= num
        return result
    def division(self, a, b):
        if b == 0:
            return "Error: Division by zero"
        return a / b
    def exponentiation(self, a, b):
        return a ** b
# Main program
calc = Calculator()
print("Simple Calculator")
print("1. Addition")
print("2. Subtraction")
print("3. Multiplication")
print("4. Division")
print("5. Exponentiation")
choice = int(input("Enter your choice (1-5): "))
if choice == 1:
```

```python
choice = int(input("Enter your choice (1-5): "))
if choice == 1:
    n = int(input("How many numbers to add? "))
    nums = [float(input(f"Enter number {i+1}: ")) for i in range(n)]
    print("Result:", calc.addition(nums))
elif choice == 2:
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    print("Result:", calc.subtraction(a, b))

elif choice == 3:
    n = int(input("How many numbers to multiply? "))
    nums = [float(input(f"Enter number {i+1}: ")) for i in range(n)]
    print("Result:", calc.multiplication(nums))

elif choice == 4:
    a = float(input("Enter numerator: "))
    b = float(input("Enter denominator: "))
    print("Result:", calc.division(a, b))

elif choice == 5:
    a = float(input("Enter base: "))
    b = float(input("Enter exponent: "))
    print("Result:", calc.exponentiation(a, b))

else:
    print("Invalid choice")
```

OUTPUT:

```
PS C:\Users\Pranitha Rao\OneDrive\Desktop\AI ASSISTED CODING> & "C:\Users\Pranitha Rao\AppData\Local\Microsoft\WindowsApps\python3.13.exe" "c:/Users/Pranitha Rao/OneDrive/Desktop/AI ASSISTED CODING/Lab-2.py"
Simple Calculator
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exponentiation
Enter your choice (1-5): 3
How many numbers to multiply? 2
Enter number 1: 2
Enter number 2: 3
Result: 6.0
PS C:\Users\Pranitha Rao\OneDrive\Desktop\AI ASSISTED CODING>
```

JUSTIFICATION:

The calculator uses methods that do not depend on any object data, so no extra memory is needed. Addition and multiplication work with a list of numbers,

which allows users to calculate multiple values at one time. Basic error checks are included to avoid problems like division by zero. Each operation is kept separate, which makes the code easy to read and update. Using built-in functions like sum() also keeps the program simple and efficient.

Task 4: Armstrong Number Optimization

PROMPT: Write a Python program to check whether a given number is an Armstrong number and print the result. (GEMINI)

CODE:

```
def is_armstrong_number(number):

    # Convert the number to a string to easily count digits and access individual digits
    num_str = str(number)
    n = len(num_str)

    sum_of_powers = 0
    for digit_char in num_str:
        digit = int(digit_char)
        sum_of_powers += digit ** n

    return sum_of_powers == number

# Test cases
n=int(input("Enter a number: "))
print(f"Is {n} an Armstrong number? {is_armstrong_number(n)}")
```

OUTPUT:

```
Enter a number: 153
Is 153 an Armstrong number? True
```

PROMPT: (using Cursor AI)

Write an efficient and readable Python function to check whether a given number is an Armstrong number, minimizing unnecessary variables and computations.

CODE:

```python
def is_armstrong_number(number: int) -> bool:
    """Check if a number is an Armstrong number."""
    num_str = str(number)
    power = len(num_str)  # number of digits

    total = sum(int(ch) ** power for ch in num_str)
    return total == number


def main() -> None:
    n = int(input("Enter a number: "))
    if is_armstrong_number(n):
        print(f"{n} is an Armstrong number.")
    else:
        print(f"{n} is not an Armstrong number.")


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

OUTPUT:

```
PS C:\Users\Pranitha Rao\OneDrive\Desktop\AI ASSISTED CODING> & "C:\Users\Pranith
a Rao\AppData\Local\Microsoft\WindowsApps\python3.13.exe" "c:/Users/Pranitha Rao/
OneDrive/Desktop/AI ASSISTED CODING/Lab-2.py"
Enter a number: 371
371 is an Armstrong number
PS C:\Users\Pranitha Rao\OneDrive\Desktop\AI ASSISTED CODING>
```

JUSTIFICATION:

The second program is better because it is short and easy to understand. It uses simple logic to achieve the same result and has clear variable and function names. It also includes a small explanation and a main part, which keeps the program neat and well organized. Overall, it does the same task in a cleaner and more efficient way.