Assignment - 02

K . Laxmi Prasanna

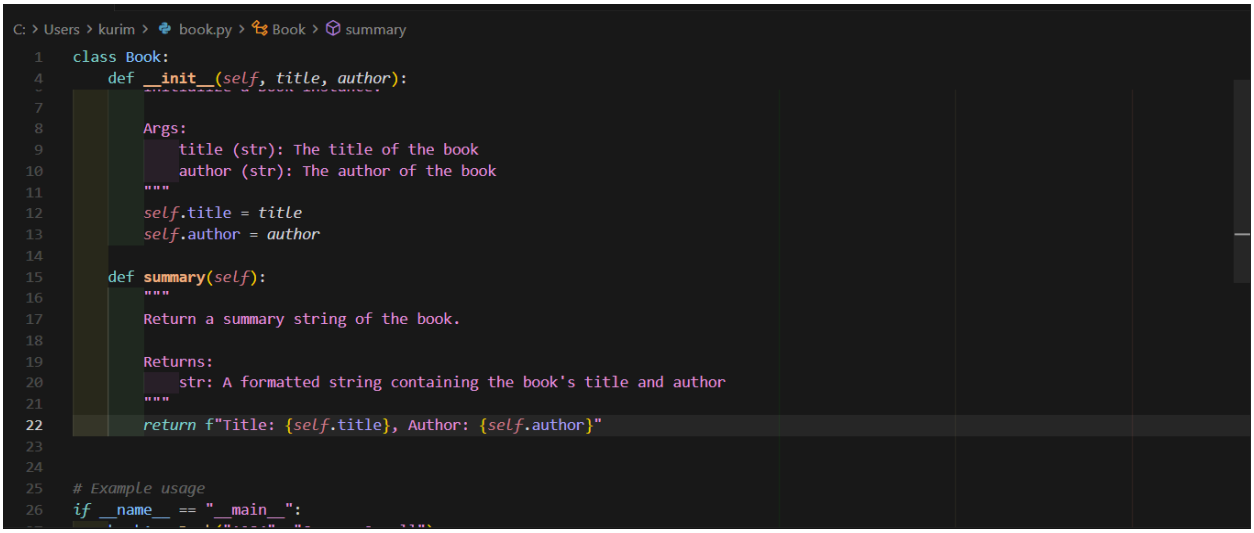
2303A52310

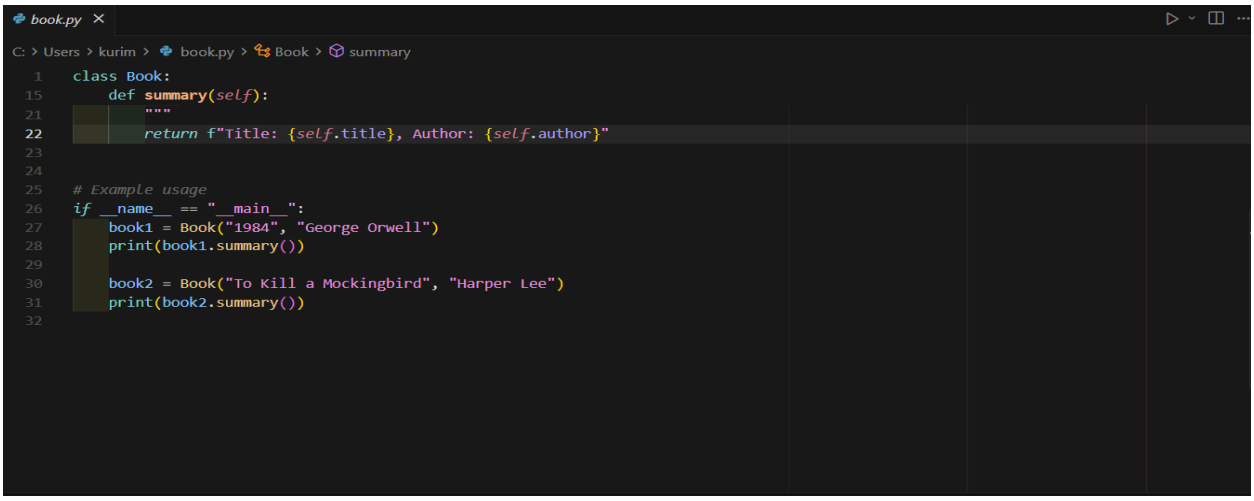
Task 1 : Book Class Generation

PROMPT:

Generate a Python class named Book for a simple library management module with attributes title and author, and a summary() method.

CODE:





OUTPUT:

```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If yo
u want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Program Files\cursor> & C:\Users\kurim\AppData\Local\Programs\Python\Python311\python.exe c:/Users/kurim/book.py
Title: 1984, Author: George Orwell
Title: To Kill a Mockingbird, Author: Harper Lee
PS C:\Program Files\cursor>
```

JUSTIFICATION:

By finishing this task, we can display the book information as required in the question using Cursor AI. The code is easy to understand.

TASK 2:

Generate Python code to sort a list of user dictionaries based on the key "age".

CODE:

```python
# List of user dictionaries
users = [
    {"name": "Alice", "age": 30, "city": "New York"},
    {"name": "Bob", "age": 25, "city": "London"},
    {"name": "Charlie", "age": 35, "city": "Paris"},
    {"name": "Diana", "age": 28, "city": "Tokyo"},
    {"name": "Eve", "age": 22, "city": "Sydney"}
]

# Method 1: Using sorted() with a lambda function (ascending order)
sorted_users_asc = sorted(users, key=lambda x: x["age"])
print("Sorted by age (ascending):")
for user in sorted_users_asc:
    print(f"  {user['name']}: {user['age']} years old")

print("\n" + "-" * 50 + "\n")

# Method 2: Using sorted() with a lambda function (descending order)
sorted_users_desc = sorted(users, key=lambda x: x["age"], reverse=True)
print("Sorted by age (descending):")
for user in sorted_users_desc:
    print(f"  {user['name']}: {user['age']} years old")
```

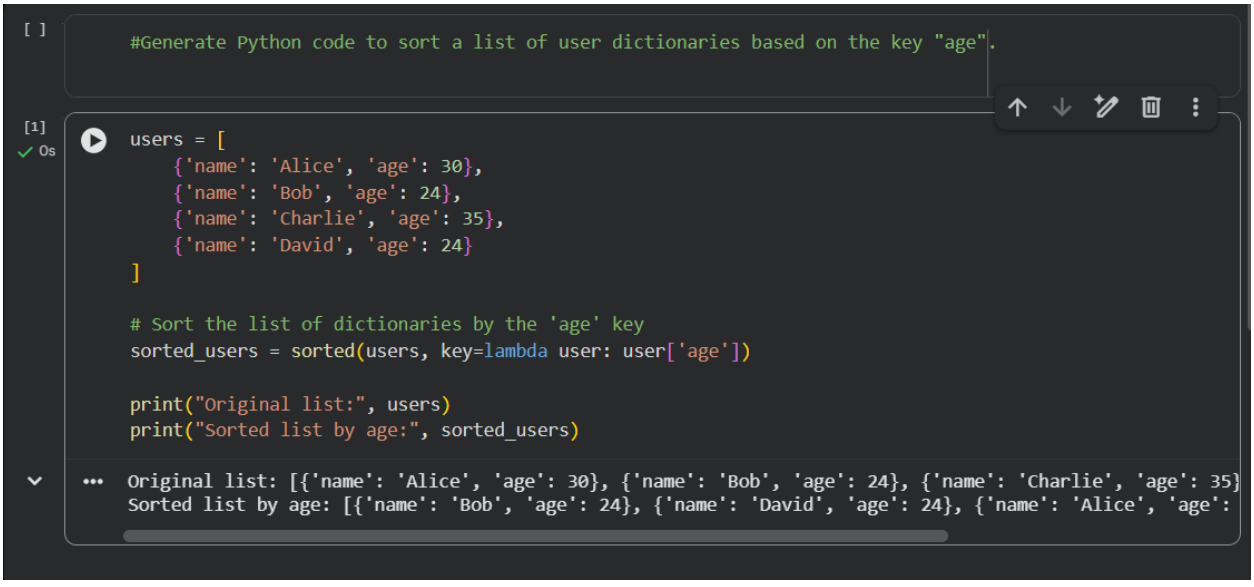
```
print("\n" + "-" * 50 + "\n")

# Method 3: Using list.sort() to sort in-place (modifies original list)
users_copy = users.copy()  # Make a copy to preserve original
users_copy.sort(key=lambda x: x["age"])
print("In-place sort (ascending):")
for user in users_copy:
    print(f"  {user['name']}: {user['age']} years old")

print("\n" + "-" * 50 + "\n")

# Method 4: Using operator.itemgetter (more efficient for large lists)
from operator import itemgetter
sorted_users_operator = sorted(users, key=itemgetter[str]("age"))
print("Sorted using operator.itemgetter:")
for user in sorted_users_operator:
    print(f"  {user['name']}: {user['age']} years old")
```

GEMINI:



OUTPUT:

```
PS C:\Program Files\cursor> & C:\Users\kurim\AppData\Local\Programs\Python\Python311\python.exe c:/Users/kurim/book.py
Title: 1984, Author: George Orwell
Title: To Kill a Mockingbird, Author: Harper Lee
PS C:\Program Files\cursor> & C:\Users\kurim\AppData\Local\Programs\Python\Python311\python.exe c:/Users/kurim/sort_users.py
Sorted by age (ascending):
  Eve: 22 years old
  Bob: 25 years old
  Diana: 28 years old
  Alice: 30 years old
  Charlie: 35 years old
```

```
----------------------------------------

Sorted by age (descending):
  Charlie: 35 years old
  Alice: 30 years old
  Diana: 28 years old
  Bob: 25 years old
  Eve: 22 years old
```

```
----------------------------------------

In-place sort (ascending):
  Eve: 22 years old
  Bob: 25 years old
  Diana: 28 years old
  Alice: 30 years old
  Charlie: 35 years old

----------------------------------------
```

```
----------------------------------------------

Sorted using operator.itemgetter:
  Eve: 22 years old
  Bob: 25 years old
  Diana: 28 years old
  Alice: 30 years old
  Charlie: 35 years old
PS C:\Program Files\cursor>
```

JUSTIFICATION:

In this task, even though we used two different AI tools, both produced the required output. we can clearly notice a difference in the way the code is written. The clarity of both codes is very good, but when it comes to output presentation, the output from Cursor AI is more neat and easier for anyone to understand.

TASK 3: Calculator Using Functions

PROMPT:

Write a python code for simple calculator with functions addition of n numbers, subtraction of n numbers, multiplication of n numbers, division of n numbers and power of n numbers which chooses the operation on the user input.

```python
def addition(numbers):
    """Addition of n numbers"""
    return sum(numbers)


def subtraction(numbers):
    """Subtraction of n numbers: first - all others"""
    if not numbers:
        return 0
    result = numbers[0]
    for num in numbers[1:]:
        result -= num
    return result

def multiplication(numbers):
    """Multiplication of n numbers"""
    if not numbers:
        return 1
    result = 1
    for num in numbers:
        result *= num
```

```
        return result
    def division(numbers):
        """Division of n numbers: first / all others"
        if not numbers:
            return 0
        result = numbers[0]
        for num in numbers[1:]:
            if num == 0:
                return "Error: Division by zero!"
            result /= num
        return result

    def power(numbers):
        """Power of n numbers: power tower (right-ass
        if not numbers:
            return 1
        if len(numbers) == 1:
            return numbers[0]

        # Right-associative power tower: a^(b^(c^d...
        result = numbers[-1]
        for i in range(len(numbers) - 2, -1, -1):
            result = numbers[i] ** result
```

```
        return result
    def get_numbers():
        """Get n numbers from user"""
        try:
            n = int(input("How many numbers do you want to enter? "))
            if n < 1:
                print("Please enter at least 1 number.")
                return None
            numbers = []
            for i in range(n):
                num = float(input(f"Enter number {i+1}: "))
                numbers.append(num)
            return numbers
        except ValueError:
            print("Invalid input! Please enter valid numbers.")
            return None
    def main():
        print("=" * 50)
        print("Simple Calculator with N Numbers")
        print("=" * 50)
        print("Operations available:")
        print("1. Addition (+)")
        print("2. Subtraction (-)")
```

```
def main():
    print("3. Multiplication (*)")
    print("4. Division (/)")
    print("5. Power (^)")
    print("0. Exit")
    print("=" * 50)

    while True:
        try:
            choice = input("\nEnter your choice (0-5): ").strip()

            if choice == '0':
                print("Thank you for using the calculator!")
                break

            elif choice == '1':
                print("\n--- Addition Operation ---")
                numbers = get_numbers()
                if numbers is not None:
                    result = addition(numbers)
                    print(f"Result: {' + '.join(map(str, numbers))} = {result}")
```

```
            elif choice == '2':
                print("\n--- Subtraction Operation ---")
                numbers = get_numbers()
                if numbers is not None:
                    result = subtraction(numbers)
                    print(f"Result: {' - '.join(map(str, numbers))} = {result}")
            elif choice == '3':
                print("\n--- Multiplication Operation ---")
                numbers = get_numbers()
                if numbers is not None:
                    result = multiplication(numbers)
                    print(f"Result: {' * '.join(map(str, numbers))} = {result}")
            elif choice == '4':
                print("\n--- Division Operation ---")
                numbers = get_numbers()
                if numbers is not None:
                    result = division(numbers)
                    if isinstance(result, str):
                        print(result)
                    else:
                        print(f"Result: {' / '.join(map(str, numbers))} = {result}")
```

```
            elif choice == '5':
                print("\n--- Power Operation (Power Tower) ---")
                numbers = get_numbers()
                if numbers is not None:
                    result = power(numbers)
                    print(f"Result: {'^'.join(map(str, numbers))} = {result}")
            else:
                print("Invalid choice! Please enter a number between 0-5.")
        except KeyboardInterrupt:
            print("\n\nProgram interrupted. Goodbye!")
            break
        except Exception as e:
            print(f"An error occurred: {e}")

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

OUTPUT :

```
==================================================
Simple Calculator with N Numbers
==================================================
Operations available:
1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/)
5. Power (^)
0. Exit
==================================================
```

```
Enter your choice (0-5): 4

--- Division Operation ---
How many numbers do you want to enter? 2
Enter number 1: 45
Enter number 2: 2
Result: 45.0 / 2.0 = 22.5

Enter your choice (0-5): 0
Thank you for using the calculator!
```

JUSTIFICATION:

By finishing this task the simple basic calculator, The program is easy to understand because each operation is written in a separate function. It can perform calculations on more than two numbers, so it is more useful than a normal calculator. The program checks for errors like wrong input and division by zero, so it runs safely.

Task 4: Armstrong Number Optimization

PROMPT:

check whether the given number is Armstrong or not

CODE:

GEMINI:

```
def is_armstrong(number):
    """Checks if a given number is an Armstrong number."""
    # Convert number to string to find the number of digits
    str_number = str(number)
    num_digits = len(str_number)

    sum_of_powers = 0
    for digit_char in str_number:
        digit = int(digit_char)
        sum_of_powers += digit ** num_digits

    return sum_of_powers == number
try:
    num_to_check = int(input("Enter a number to check if it's an Armstrong number: "))
    if is_armstrong(num_to_check):
        print(f"{num_to_check} is an Armstrong number.")
    else:
        print(f"{num_to_check} is not an Armstrong number.")
except ValueError:
    print("Invalid input. Please enter a valid integer.")
```

CURSOR AI :

```
n=int(input("Enter the number: "))
l=[]
y=n
while n>0:
    f=n%10
    l.append(f)
    n=n//10
x=len(l)
s=0
for i in l:
    s=s+i**x
if s==y:
    print("The number is amstrong")
else:
    print("The number is not amstrong")
```

OUTPUT:

GEMINI:

```
Enter a number to check if it's an Armstrong number: 153
153 is an Armstrong number.
```

CURSOR AI:

JUSTIFICATION:

In this task, we can clearly observe that, the code given by gemini is a bit complex and has more time complexity which makes code less efficient, as we have optimized it in cursor ai, the code is optimized version with only basic syntax which makes the code easy to understand for anyone. Even though the outputs are same, both the codes given by both gemini and cursor ai are efficient.