ASSIGNMENT 02

D. VIDHU SREE- 2303A52400(B41)

Task 1: Book Class Generation

PROMPT :

code with class book with attributes title, author, price and a summary method

CODE:

```
class Book:
    def __init__(self, title, author, price):
        self.title = title
        self.author = author
        self.price = price
    def summary(self):
        print(f"Title: {self.title}")
        print(f"Author: {self.author}")
        print(f"Price: {self.price}")
book = Book("The Great Gatsby", "F. Scott Fitzgerald", 10.99)
print("------------------------------")
print("Book Summary:")
book.summary()
```

OUTPUT:

```
------------------------------
Book Summary:
Title: The Great Gatsby
Author: F. Scott Fitzgerald
Price: 10.99
```

JUSTIFICATION:

By completing this task, we are able to print the book details as mentioned in the question using cursor ai, the code given is accurate and easy to understand.

Task 2: Sorting Dictionaries with AI

PROMPT:

Sort the dictionary by key values in dic{age:name} and print the names of persons in sorted manner with ages

CODE:

CURSOR AI:

```
dic={20:"John",25:"Jane",30:"Jim",35:"Jill",40:"Jack",45:"Jonny",50:"Johnson"}
dic=dict[int, str](sorted(dic.items(), key=lambda x: x[0]))
print()
for i in dic:
    print(f"{dic[i]} is {i} years old")
```

GEMINI:

```
my_dict = {30: 'Alice', 25: 'Bob', 35: 'Charlie', 28: 'David'}

# Sort the dictionary by keys (ages)
sorted_keys = sorted(my_dict.keys())

print("Names of persons in sorted manner with ages:")
for age in sorted_keys:
    print(f"Age: {age}, Name: {my_dict[age]}")
```

OUTPUTS:

CURSOR AI:

```
John is 20 years old
Jane is 25 years old
Jim is 30 years old
Jill is 35 years old
Jack is 40 years old
Jonny is 45 years old
Johnson is 50 years old
```

GEMINI:

```
Names of persons in sorted manner with ages:
Age: 25, Name: Bob
Age: 28, Name: David
Age: 30, Name: Alice
Age: 35, Name: Charlie
```

JUSTIFICATION:

In this task, although we used both different AI, we are getting the same required output, but we can clearly see the difference in approach of the code, clarity for both the codes is very fine and when comes to the representation of the output, the output given by cursor ai is neat and easy to understand by anyone.

Task 3: Calculator Using Functions

PROMPT:

simple calculator with functions addition of n numbers, subtraction of 2 numbers, multiplication of n numbers, division of 2 numbers and power of 2 numbers which chooses the operation on the user input

CODE:

CURSOR AI:

```
def addition(n):
    sum=0
    for i in n:
        sum+=i
    return sum
def subtraction(x,y):
    return x-y
def multiplication(n):
    product=1
    for i in n:
        product*=i
    return product
def division(x,y):
    return x/y
def power(x,y):
    return x**y
print("1 -> Addition \n2 -> Subtraction \n3 -> Multiplication \n4 -> Division \n5 -> Power")
print("--------------------------------")
c=int(input("Enter the choice: "))
```

```
c=int(input("Enter the choice: "))
if c==1:
    n=int(input("Enter the number of numbers for addition: "))
    l=[]
    for i in range(0,n):
        x=int(input("Enter the number: "))
        l.append(x)
    print("Addition of numbers: ",addition(l))
elif c==2:
    x=int(input("Enter the number for subtraction: "))
    y=int(input("Enter the number for subtraction: "))
    print("Subtraction of numbers: ",subtraction(x,y))
elif c==3:
    n=int(input("Enter the number of numbers for multiplication: "))
    l=[]
    for i in range(0,n):
        x=int(input("Enter the number: "))
        l.append(x)
    print("Multiplication of numbers: ",multiplication(l))
elif c==4:
    x=int(input("Enter the number for division: "))
    y=int(input("Enter the number for division: "))
    print("Division of numbers: ",division(x,y))
elif c==5:
    x=int(input("Enter the number for base: "))
    y=int(input("Enter the number for power: "))
    print("Power of numbers: ",power(x,y))
else:
    print("Invalid choice")
```

GEMINI:

```python
def add_numbers(*args):
    """Adds a variable number of arguments."""
    return sum(args)

def subtract_numbers(num1, num2):
    """Subtracts two numbers."""
    return num1 - num2

def multiply_numbers(*args):
    """Multiplies a variable number of arguments."""
    result = 1
    for num in args:
        result *= num
    return result

def divide_numbers(num1, num2):
    """Divides two numbers."""
    if num2 == 0:
        raise ValueError("Error: Division by zero is not allowed.")
    return num1 / num2

def power_numbers(base, exponent):
    """Calculates the power of two numbers (base^exponent)."""
    return base ** exponent
```

```python
def get_n_numbers(prompt_msg):
    """Helper function to get multiple numbers from user input."""
    numbers = []
    while True:
        user_input = input(prompt_msg + " (type 'done' to finish): ")
        if user_input.lower() == 'done':
            break
        try:
            numbers.append(float(user_input))
        except ValueError:
            print("Invalid input. Please enter a valid number or 'done'.")
    return numbers

def main_calculator():
    """Main function for the calculator program."""
    print("Welcome to the Simple Calculator!")

    while True:
        print("\nSelect an operation:")
        print("1. Addition (n numbers)")
        print("2. Subtraction (2 numbers)")
        print("3. Multiplication (n numbers)")
        print("4. Division (2 numbers)")
        print("5. Power (2 numbers)")
        print("6. Exit")

        choice = input("Enter choice (1/2/3/4/5/6): ")
```

```
        if choice == '6':
            print("Exiting calculator. Goodbye!")
            break

        try:
            if choice == '1':
                nums = get_n_numbers("Enter numbers to add")
                if len(nums) < 1:
                    print("Please enter at least one number for addition.")
                    continue
                print(f"Result: {add_numbers(*nums)}")
            elif choice == '2':
                num1 = float(input("Enter first number: "))
                num2 = float(input("Enter second number: "))
                print(f"Result: {subtract_numbers(num1, num2)}")
            elif choice == '3':
                nums = get_n_numbers("Enter numbers to multiply")
                if len(nums) < 1:
                    print("Please enter at least one number for multiplication.")
                    continue
                print(f"Result: {multiply_numbers(*nums)}")
            elif choice == '4':
                num1 = float(input("Enter first number (dividend): "))
                num2 = float(input("Enter second number (divisor): "))
                print(f"Result: {divide_numbers(num1, num2)}")
            elif choice == '5':
                base = float(input("Enter base number: "))
                exponent = float(input("Enter exponent number: "))
                print(f"Result: {power_numbers(base, exponent)}")
            else:
                print("Invalid input. Please enter a number between 1 and 6.")
        except ValueError as e:
            print(e)
        except Exception as e:
            print(f"An unexpected error occurred: {e}")

# Run the calculator
main_calculator()
```

OUTPUTS:

CURSOR AI:

```
1 -> Addition
2 -> Subtraction
3 -> Multiplication
4 -> Division
5 -> Power
--------------------------------
Enter the choice: 3
Enter the number of numbers for multiplication: 3
Enter the number: 4
Enter the number: 5
Enter the number: 6
Multiplication of numbers:  120
```

GEMINI AI:

```
Welcome to the Simple Calculator!

Select an operation:
1. Addition (n numbers)
2. Subtraction (2 numbers)
3. Multiplication (n numbers)
4. Division (2 numbers)
5. Power (2 numbers)
6. Exit
Enter choice (1/2/3/4/5/6): 1
Enter numbers to add (type 'done' to finish): 2
Enter numbers to add (type 'done' to finish): 3
Enter numbers to add (type 'done' to finish): 6
Enter numbers to add (type 'done' to finish): 5
Enter numbers to add (type 'done' to finish): 0
Enter numbers to add (type 'done' to finish): done
Result: 16.0

Select an operation:
1. Addition (n numbers)
2. Subtraction (2 numbers)
3. Multiplication (n numbers)
4. Division (2 numbers)
5. Power (2 numbers)
6. Exit
Enter choice (1/2/3/4/5/6): 6
Exiting calculator. Goodbye!
```

JUSTIFICATION:

Here, in this task, the main idea is same, to prepare the simple calculator, when we see the code for both cursor ai and gemini, we can see that the code is almost same but a small difference is present which brings the major change, which is, in gemini once if we run the code the calculator will run again and again until we exit it. This feature is very much important in the daily life. When we look into clarity, cursor ai output is clear and easy to understand but gemini is well in advanced for real world.

Task 4: Armstrong Number Optimization

PROMPT:

python code to check whether the given number is Armstrong or not

CODE:

GEMINI:

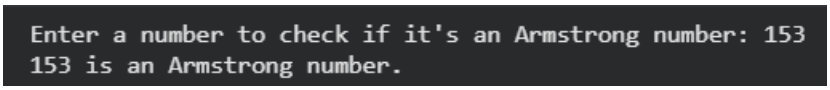
```
def is_armstrong(number):
    """Checks if a given number is an Armstrong number."""
    # Convert number to string to find the number of digits
    str_number = str(number)
    num_digits = len(str_number)

    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:



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. Eventhough the outputs are same, both the codes given by both gemini and cursor ai are efficient.