

ASSIGNMENT 02

HRK Sri Harsha Vignesh

2303A52304

Batch-42

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 dictionaries 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")

```

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

```

JUSTIFICATION:

Here, in this task, The program **shows** the result clearly. The function **returns** the output. The code **provides** correct results.

Task 4: Armstrong Number Optimization

PROMPT:

Write a 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:

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

CURSOR AI:

```
Enter the number: 153
The number is amstrong
```

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.