Assignment:9.1

Name:Aashutosh

Hall Tkt:2403a51316

Task 1:Use AI to create an employee class with attributes(name,id,salary) and a method to calculate yearly salary.

Instruction:

Prompt AI to generate the Employee class.

Analyze the generated code for correctness and structure.

Ask AI to add a method to give a bonus and recalculate salary.

Ans:

Prompt:-Create a class employee with attributes name id salary and a method to calculate yearly salary. also add a method so that the user should provide it's current salary and bonus and based on that a final recalculated salary should be calculated.

```python
class Employee:
    def __init__(self, name, id, salary):
        self.name = name
        self.id = id
        self.salary = salary

    def calculate_yearly_salary(self):
        return self.salary * 12

    def give_bonus(self, bonus_amount):
        self.salary += bonus_amount
        print(f"Bonus of ${bonus_amount} given to {self.name}. New monthly salary is ${self.salary}.")
        print(f"New yearly salary after bonus: ${self.calculate_yearly_salary()}")
```

Gemini

```python
name = input("Enter employee name: ")
id = input("Enter employee ID: ")
salary = float(input("Enter employee monthly salary: "))

employee = Employee(name, id, salary)

print(f"Initial yearly salary for {employee.name}: ${employee.calculate_yearly_salary()}")

bonus = float(input("Enter bonus amount: "))
employee.give_bonus(bonus)

print(f"Yearly salary after bonus for {employee.name}: ${employee.calculate_yearly_salary()}")
```

```
Enter employee name: Aashutosh
Enter employee ID: 1001
Enter employee monthly salary: 1000000
Initial yearly salary for Aashutosh: $12000000.0
Enter bonus amount: 200000
Bonus of $200000.0 given to Aashutosh. New monthly salary is $1200000.0.
New yearly salary after bonus: $14400000.0
Yearly salary after bonus for Aashutosh: $14400000.0
```

Task2:-Prompt AI to generate a function that displays all Automorphic numbers between 1 and 1000 using a for loop.

Instruction:

Get AI-generated code to list Automorphic number using a for loop.

Analyze the correctness and efficiency of the generated logic.

Ask AI to regenerate using a while loop and compare both implementations.

Ans:-

Prompt:-

Generate a function to get all automorphic numbers between 1 and 1000 using for loop and display it put the number in a list.

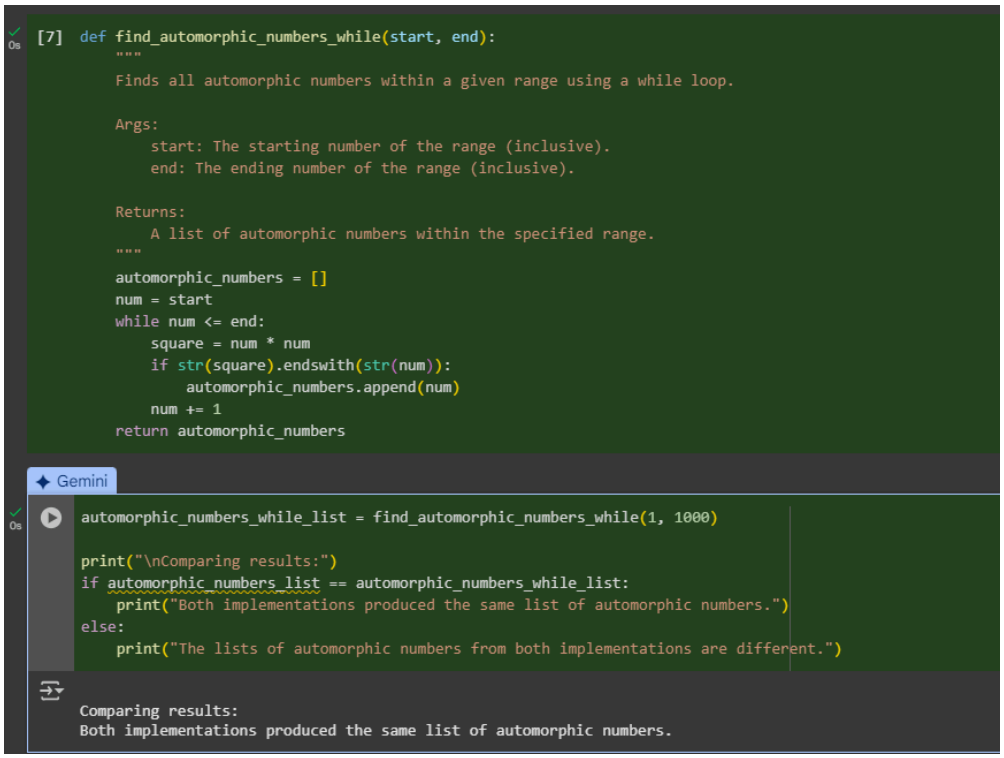
```python
def find_automorphic_numbers(start, end):
    """
    Finds all automorphic numbers within a given range.

    An automorphic number is a number whose square
     ends in the same digits as the number itself.

    Args:
        start: The starting number of the range (inclusive).
        end: The ending number of the range (inclusive).

    Returns:
        A list of automorphic numbers within the specified range.
    """
    automorphic_numbers = []
    for num in range(start, end + 1):
        square = num * num
        if str(square).endswith(str(num)):
            automorphic_numbers.append(num)
    return automorphic_numbers
```

Gemini

```python
[6] automorphic_numbers_list = find_automorphic_numbers(1, 1000)
    print("Automorphic numbers between 1 and 1000:")
    print(automorphic_numbers_list)
```

```
Automorphic numbers between 1 and 1000:
[1, 5, 6, 25, 76, 376, 625]
```

Now use while loop to generate and compare both implementation.



```
[7] def find_automorphic_numbers_while(start, end):
        """
        Finds all automorphic numbers within a given range using a while loop.

        Args:
            start: The starting number of the range (inclusive).
            end: The ending number of the range (inclusive).

        Returns:
            A list of automorphic numbers within the specified range.
        """
        automorphic_numbers = []
        num = start
        while num <= end:
            square = num * num
            if str(square).endswith(str(num)):
                automorphic_numbers.append(num)
            num += 1
        return automorphic_numbers
```

Gemini

```
automorphic_numbers_while_list = find_automorphic_numbers_while(1, 1000)

print("\nComparing results:")
if automorphic_numbers_list == automorphic_numbers_while_list:
    print("Both implementations produced the same list of automorphic numbers.")
else:
    print("The lists of automorphic numbers from both implementations are different.")
```

```
Comparing results:
Both implementations produced the same list of automorphic numbers.
```

Task3: Ask AI to write nested if-else-else condition to classify online shopping feedback as Positive, Neutral, or Negative based on numerical rating.

Instruction:

Generate initial code using nested if-elif-else.

Analyze correctness and readability.

Ask AI to rewrite using dictionary-based or match-case structure.

Ans:-

Prompt:-Write a nested if elif else condition to classify the feedback on online shopping platform as positive negative and neutral based on a numeric-rating(1-5). The user should give input between 1-5 and based on feedback it should display required output.

```
rating = int(input("Enter your rating (1-5): "))

if rating >= 1 and rating <= 5:
    if rating >= 4:
        print("Feedback: Positive")
    elif rating == 3:
        print("Feedback: Neutral")
    else:
        print("Feedback: Negative")
else:
    print("Invalid rating. Please enter a number between 1 and 5.")
```

```
Enter your rating (1-5): 3
Feedback: Neutral
```

Prompt:-Now rewrite the logic using dictionary-based or match-case structure.

```
rating = int(input("Enter your rating (1-5): "))

match rating:
    case 5 | 4:
        print("Feedback: Positive")
    case 3:
        print("Feedback: Neutral")
    case 2 | 1:
        print("Feedback: Negative")
    case _:
        print("Invalid rating. Please enter a number between 1 and 5.")
```

```
Enter your rating (1-5): 4
Feedback: Positive
```

Task 4:Generate a function using AI that display all prime numbers within a user-specified range (e.g., 1 to 500).

Instruction:

Generate AI-generated code to list all prime using a for loop.

Analyze the correctness and efficiency of the prime-checking logic.

Ask AI to regenerate an optimized version(e.g., using the square root method).

Ans:-

Prompt:-Generate a function to find prime number between two integer the function should that two integral argument treated as range between which we have to find prime number.Add option so that user can write two numbers.

```
def find_primes_in_range(start, end):
    """
    Finds all prime numbers within a given range.

    Args:
        start: The starting integer of the range (inclusive).
        end: The ending integer of the range (inclusive).

    Returns:
        A list of prime numbers within the specified range.
    """
    prime_numbers = []
    for num in range(start, end + 1):
        if num > 1:  # Prime numbers are greater than 1
            is_prime = True
            for i in range(2, int(num**0.5) + 1):
                if (num % i) == 0:
                    is_prime = False
                    break
            if is_prime:
                prime_numbers.append(num)
    return prime_numbers
```

```
start_range = int(input("Enter the starting number of the range: "))
end_range = int(input("Enter the ending number of the range: "))

prime_list = find_primes_in_range(start_range, end_range)

print(f"\nPrime numbers between {start_range} and {end_range}:")
print(prime_list)
```

Note:-Already given optimised code in complexity of order: o(root(n)).

Task5: Task: Use AI to build a Library class with methods to add_book(), issue_book(), and display_books().

Insturction:

Generate Library class code using AI.

Analyze if method handle edge cases (e.g., issuing unavailable books).

Ask AI to add comments and documentation.

Ans:-

Prompt:-Generate a class library with methods like add_book(), issue_book() and display_book(). The code should give menu options to do these and it should also handle edge cases like issuing unavailable book. It should also have a option in menu for the termination of program. Give a code that consist of all the defined classes with methods and menu down of it.

```
[16] class Library:
         def __init__(self):
             self.books = []
```

```
[19] class Library:
         def __init__(self):
             self.books = []

         def add_book(self, book):
             """Adds a book to the library's collection."""
             self.books.append(book)
             print(f"Book '{book}' added to the library.")
```

```
class Library:
    def __init__(self):
        self.books = []

    def add_book(self, book):
        """Adds a book to the library's collection."""
        self.books.append(book)
        print(f"Book '{book}' added to the library.")

    def display_books(self):
        """Displays all books in the library's collection."""
        if not self.books:
            print("There are no books in the library.")
        else:
            print("Books in the library:")
            for book in self.books:
                print(f"- {book}")
```

```
library = Library()

while True:
    print("\nLibrary Menu:")
    print("1. Add Book")
    print("2. Display Books")
    print("3. Issue Book")
    print("4. Exit")

    try:
        choice = int(input("Enter your choice: "))
    except ValueError:
        print("Invalid input. Please enter a number.")
        continue

    if choice == 1:
        book_name = input("Enter the name of the book to add: ")
        library.add_book(book_name)
    elif choice == 2:
        library.display_books()
    elif choice == 3:
        book_name = input("Enter the name of the book to issue: ")
        library.issue_book(book_name)
    elif choice == 4:
        print("Exiting Library System. Goodbye!")
        break
    else:
        print("Invalid choice. Please enter a number between 1 and 4.")
```
