AI ASSISTED CODING

ASSIGNMENT-6.1

2403A51288

BATCH-12

Task -1:

#Use AI to create an Employee class with attributes (name, id, salary) and a method to calculate yearly salary.

code:

```
class Employee:
    def __init__(self, name, emp_id, salary):
        """
        Initialize the Employee object with name, id, and salary.
        """
        self.name = name
        (method) def calculate_yearly_salary(self: Self@Employee) -> Any
        Calculate and return the yearly salary.
    def calculate_yearly_salary(self):
        """
        Calculate and return the yearly salary.
        """
        return self.salary * 12

    def display_details(self):
        """
        Display the details of the employee.
        """
        print(f"Employee Name: {self.name}")
        print(f"Employee ID: {self.emp_id}")
        print(f"Monthly Salary: {self.salary}")
        print(f"Yearly Salary: {self.calculate_yearly_salary()}")

    def calculate_bonus(self, bonus_percentage):
        """
        Calculate and add a bonus to the salary based on the given percentage.
        """
        bonus = self.salary * (bonus_percentage / 100)
        self.salary += bonus
        print(f"Bonus of {bonus_percentage}% added. New Monthly Salary: {self.salary}")
```

output:

Employee Name: John Doe

Employee ID: 101

Monthly Salary: 5000

Yearly Salary: 60000

---- After Bonus ----

Bonus of 10% added. New Monthly Salary: 5500.0

Employee Name: John Doe

Employee ID: 101

Monthly Salary: 5500.0

Yearly Salary: 66000.0

Task-2:

#Prompt AI to generate a function that displays all Automorphic numbers between 1 and 1000 using a for loop

Code:

```python
def is_automorphic(num):
    """Check if a number is Automorphic."""
    square = num ** 2
    return str(square).endswith(str(num))

def automorphic_numbers_for_loop(start, end):
    """Find and display Automorphic numbers in a range using a for loop."""
    automorphic_nums = []
    for num in range(start, end + 1):
        if is_automorphic(num):
            automorphic_nums.append(num)
    return automorphic_nums

# Display Automorphic numbers between 1 and 1000
print("Automorphic numbers (For Loop):", automorphic_numbers_for_loop(1, 1000))
```

Output:

Automorphic numbers (for loop):

1 5 6 25 76 376 625

Automorphic numbers (while loop):

1 5 6 25 76 376 625

Task-3:

#Ask AI to write nested if-elif-else conditions to classify online shopping feedback as Positive, Neutral, or Negative based on a numerical rating (1–5).

Code:

```python
def classify_feedback(rating):
    """
    Classifies online shopping feedback based on a numerical rating (1-5).

    Parameters:
        rating (int): Numerical rating between 1 and 5.

    Returns:
        str: Feedback classification as 'Positive', 'Neutral', or 'Negative'.
    """
    if 1 <= rating <= 5:
        if rating == 5 or rating == 4:
            return "Positive"
        elif rating == 3:
            return "Neutral"
        elif rating == 2 or rating == 1:
            return "Negative"
    else:
        return "Invalid rating. Please provide a rating between 1 and 5."
```

Output:

Positive

Positive

Neutral

Negative

Negative

Invalid rating. Please provide a rating between 1 and 5.

Invalid rating. Please provide a rating between 1 and 5.

Task 4:

#Generate a function using AI that displays all prime numbers within a user-specified range (e.g., 1 to 500).

Code:

```python
def is_prime(num):
    """Check if a number is prime."""
    if num <= 1:
        return False
    for i in range(2, int(num ** 0.5) + 1):  # Optimized using square root
        if num % i == 0:
            return False
    return True

def list_primes_in_range(start, end):
    """List all prime numbers in a given range."""
    primes = []
    for num in range(start, end + 1):
        if is_prime(num):
            primes.append(num)
    return primes

# User-specified range
start = int(input("Enter the start of the range: "))
end = int(input("Enter the end of the range: "))

# Display prime numbers
print(f"Prime numbers between {start} and {end}: {list_primes_in_range(start, end)}")
```

Output:

Enter the start of the range: 1

Enter the end of the range: 20

Prime numbers between 1 and 20: [2, 3, 5, 7, 11, 13, 17, 19]

Task 5:

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

CODE:

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

    def add_book(self, book):
        self.books.append(book)
        print(f'"{book}" added.')

    def issue_book(self, book):
        if book in self.books:
            self.books.remove(book)
            print(f'"{book}" issued.')
        else:
            print(f'"{book}" not available.')

    def display_books(self):
        if self.books:
            print("Available books:")
            for b in self.books:
                print(f"- {b}")
        else:
            print("No books available.")

# Example usage
if __name__ == "__main__":
    lib = Library()
    lib.add_book("The Great Gatsby")
    lib.add_book("1984")
    lib.add_book("To Kill a Mockingbird")
    lib.display_books()
    lib.issue_book("1984")
    lib.issue_book("The Catcher in the Rye")
    lib.display_books()
```

Output:

"The Great Gatsby" added.

"1984" added.

"To Kill a Mockingbird" added.

Available books:

- The Great Gatsby

- 1984

- To Kill a Mockingbird

"1984" issued.

"The Catcher in the Rye" not available.

Available books:

- The Great Gatsby

- To Kill a Mockingbird