Q1. Write a program that has a class Fraction with attributes numerator and denominator. Enter the values of the attributes and print the fraction in simplified form.

```
class Fraction:
    def __init__(self, numerator, denominator):
        if denominator == 0:
            raise ValueError("Denominator cannot be zero")
        self.numerator = numerator
        self.denominator = denominator
        self.simplify()

    def gcd(self, a, b):
        """Calculate the Greatest Common Divisor of a and b."""
        while b:
            a, b = b, a % b
        return a

    def simplify(self):
        """Simplify the fraction to its lowest terms."""
        # Handle negative numbers
        sign = -1 if self.numerator * self.denominator < 0 else 1

        # Convert to positive numbers for GCD calculation
        num = abs(self.numerator)
        den = abs(self.denominator)

        # Calculate GCD and simplify
        common = self.gcd(num, den)
        self.numerator = sign * (num // common)
        self.denominator = den // common

    def __str__(self):
        """Return string representation of the fraction."""
        if self.denominator == 1:
            return str(self.numerator)
        return f"{self.numerator}/{self.denominator}"

# Example usage
def main():
    try:
        # Get input from user
        num = int(input("Enter numerator: "))
        den = int(input("Enter denominator: "))

        # Create fraction object
        fraction = Fraction(num, den)

        # Print simplified fraction
        print(f"Simplified fraction: {fraction}")
    except ValueError as e:
        print(f"Error: {e}")
    except Exception as e:
        print(f"An error occurred: {e}")

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

```
Enter numerator: 14
Enter denominator: 6
Simplified fraction: 7/3
```

Q2. Write a program that has a class store which keeps record of code and price of each product.

Display a menu of all products to the user and prompt him to enter the quantity of each item required. Generate a bill and display the total amount.

```
class Product:
    def __init__(self, code, name, price):
        self.code = code
        self.name = name
        self.price = price

class Store:
    def __init__(self):
        # Initialize store with some sample products
        self.products = {
            'P001': Product('P001', 'Laptop', 999.99),
            'P002': Product('P002', 'Mouse', 29.99),
            'P003': Product('P003', 'Keyboard', 59.99),
            'P004': Product('P004', 'Monitor', 299.99),
            'P005': Product('P005', 'Headphones', 79.99)
        }
        self.cart = {}

    def display_menu(self):
        """Display all available products"""
        print("\n=== Available Products ===")
        print("Code\tName\t\tPrice")
        print("-" * 30)
        for product in self.products.values():
            print(f"{product.code}\t{product.name:<12}\t${product.price:.2f}")

    def add_to_cart(self, code, quantity):
        """Add a product to cart with specified quantity"""
        if code in self.products:
            self.cart[code] = quantity
            return True
        return False

    def generate_bill(self):
        """Generate and display the bill"""
        if not self.cart:
            print("\nCart is empty!")
            return

        print("\n========== BILL ==========")
        print("Product\t\tQty\tPrice\tTotal")
        print("-" * 40)

        grand_total = 0
        for code, qty in self.cart.items():
            product = self.products[code]
            total = qty * product.price
            grand_total += total
            print(f"{product.name:<12}\t{qty}\t${product.price:.2f}\t${total:.2f}")

        print("-" * 40)
        print(f"Grand Total:\t\t\t${grand_total:.2f}")
        print("=" * 40)

def main():
    store = Store()

    while True:
        print("\n=== Store Management System ===")
        print("1. Display Products")
        print("2. Add to Cart")
        print("3. Generate Bill")
        print("4. Exit")

        choice = input("\nEnter your choice (1-4): ")

        if choice == '1':
            store.display_menu()

        elif choice == '2':
            store.display_menu()
            code = input("\nEnter product code: ").upper()
            try:
                quantity = int(input("Enter quantity: "))
                if quantity < 0:
                    print("Quantity cannot be negative!")
                    continue
                if store.add_to_cart(code, quantity):
                    print("Product added to cart successfully!")
                else:
                    print("Invalid product code!")
            except ValueError:
                print("Invalid quantity!")

        elif choice == '3':
            store.generate_bill()

        elif choice == '4':
            print("\nThank you for shopping with us!")
            break

        else:
            print("\nInvalid choice! Please try again.")

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

```
P002    Mouse           $29.99
P003    Keyboard        $59.99
P004    Monitor         $299.99
P005    Headphones      $79.99

=== Store Management System ===
1. Display Products
2. Add to Cart
3. Generate Bill
4. Exit

Enter your choice (1-4): 2

=== Available Products ===
Code    Name            Price
------------------------------
P001    Laptop          $999.99
P002    Mouse           $29.99
P003    Keyboard        $59.99
P004    Monitor         $299.99
P005    Headphones      $79.99

Enter product code: P005
Enter quantity: 1
Product added to cart successfully!

=== Store Management System ===
1. Display Products
2. Add to Cart
3. Generate Bill
4. Exit

Enter your choice (1-4): 3

========== BILL ==========
Product         Qty     Price   Total
----------------------------------------
Headphones      1       $79.99  $79.99
----------------------------------------
Grand Total:                    $79.99
========================================

=== Store Management System ===
1. Display Products
2. Add to Cart
3. Generate Bill
4. Exit

Enter your choice (1-4): 4

Thank you for shopping with us!
```

Q3. Write a class that stores a string and all its status details such as number of uppercase characters, vowels, consonants, spaces etc.

```python
class StringAnalyzer:
    def __init__(self, text):
        self.text = text
        self.uppercase = 0
        self.lowercase = 0
        self.vowels = 0
        self.consonants = 0
        self.spaces = 0
        self.digits = 0
        self.special_chars = 0
        self.analyze()

    def is_vowel(self, char):
        """Check if a character is a vowel"""
        return char.lower() in 'aeiou'

    def is_consonant(self, char):
        """Check if a character is a consonant"""
        return char.isalpha() and not self.is_vowel(char)
    def analyze(self):
        """Analyze the string for various characteristics"""
        for char in self.text:
            # Count uppercase letters
            if char.isupper():
                self.uppercase += 1

            # Count lowercase letters
            elif char.islower():
                self.lowercase += 1

            # Count spaces
            if char.isspace():
                self.spaces += 1

            # Count digits
            elif char.isdigit():
                self.digits += 1

            # Count vowels and consonants
            if self.is_vowel(char):
                self.vowels += 1
            elif self.is_consonant(char):
                self.consonants += 1

            # Count special characters
            elif not char.isalnum() and not char.isspace():
                self.special_chars += 1

    def display_stats(self):
        """Display all string statistics"""
        print("\nString Analysis Results:")
        print("-" * 30)
        print(f"Original Text: {self.text}")
        print(f"Length: {len(self.text)} characters")
        print("\nCharacter Counts:")
        print(f"Uppercase Letters: {self.uppercase}")
        print(f"Lowercase Letters: {self.lowercase}")
        print(f"Vowels: {self.vowels}")
        print(f"Consonants: {self.consonants}")
        print(f"Digits: {self.digits}")
        print(f"Spaces: {self.spaces}")
        print(f"Special Characters: {self.special_chars}")

        # Calculate percentages for additional insights
        total_chars = len(self.text)
        if total_chars > 0:
            print("\nPercentages:")
            print(f"Letters: {((self.uppercase + self.lowercase) / total_chars * 100):.1f}%")
            print(f"Vowels: {(self.vowels / total_chars * 100):.1f}%")
            print(f"Consonants: {(self.consonants / total_chars * 100):.1f}%")
            print(f"Digits: {(self.digits / total_chars * 100):.1f}%")
            print(f"Spaces: {(self.spaces / total_chars * 100):.1f}%")
            print(f"Special Characters: {(self.special_chars / total_chars * 100):.1f}%")

def main():
    while True:
        text = input("\nEnter a string to analyze (or press Enter to exit): ")
        if not text:
            print("Goodbye!")
```

```
                break

            analyzer = StringAnalyzer(text)
            analyzer.display_stats()

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

```
Enter a string to analyze (or press Enter to exit): Hi This is Danish from SpaceX crew dragon capsule.!!

String Analysis Results:
------------------------------
Original Text: Hi This is Danish from SpaceX crew dragon capsule.!!
Length: 52 characters

Character Counts:
Uppercase Letters: 5
Lowercase Letters: 36
Vowels: 14
Consonants: 27
Digits: 0
Spaces: 8
Special Characters: 3

Percentages:
Letters: 78.8%
Vowels: 26.9%
Consonants: 51.9%
Digits: 0.0%
Spaces: 15.4%
Special Characters: 5.8%

Enter a string to analyze (or press Enter to exit):
Goodbye!
```

Q4. Write a program that has a class person. Inherit a class Faculty from person which also has class Publications.

```
class Person:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

    def display_info(self):
        """Display basic person information"""
        print("\nPerson Information:")
        print("-" * 20)
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")
        print(f"Gender: {self.gender}")

class Publications:
    def __init__(self):
        self.publications = []

    def add_publication(self, title, year, journal):
        """Add a new publication"""
        publication = {
            'title': title,
            'year': year,
            'journal': journal
        }
        self.publications.append(publication)
    def display_publications(self):
        """Display all publications"""
        if not self.publications:
            print("\nNo publications available.")
            return

        print("\nPublications:")
        print("-" * 50)
        for i, pub in enumerate(self.publications, 1):
            print(f"\nPublication {i}:")
            print(f"Title: {pub['title']}")
            print(f"Year: {pub['year']}")
            print(f"Journal: {pub['journal']}")

class Faculty(Person, Publications):
    def __init__(self, name, age, gender, department, designation):
        Person.__init__(self, name, age, gender)
        Publications.__init__(self)
        self.department = department
        self.designation = designation
        self.courses = []

    def add_course(self, course_name):
        """Add a course taught by faculty"""
        self.courses.append(course_name)

    def display_faculty_info(self):
        """Display complete faculty information"""
        # Display basic person information
        self.display_info()

        # Display faculty-specific information
        print("\nFaculty Information:")
        print("-" * 20)
        print(f"Department: {self.department}")
        print(f"Designation: {self.designation}")

        # Display courses
        if self.courses:
            print("\nCourses Teaching:")
            for i, course in enumerate(self.courses, 1):
                print(f"{i}. {course}")

        # Display publications
        self.display_publications()

def main():
    # Create a faculty member
    faculty = Faculty(
        name=input("Enter faculty name: "),
        age=int(input("Enter age: ")),
        gender=input("Enter gender: "),
        department=input("Enter department: "),
        designation=input("Enter designation: ")
    )

    # Menu-driven program
    while True:
        print("\n=== Faculty Management System ===")
        print("1. Add Course")
        print("2. Add Publication")
        print("3. Display Faculty Information")
        print("4. Exit")

        choice = input("\nEnter your choice (1-4): ")

        if choice == '1':
            course = input("Enter course name: ")
            faculty.add_course(course)
            print("Course added successfully!")

        elif choice == '2':
            title = input("Enter publication title: ")
            year = input("Enter publication year: ")
            journal = input("Enter journal name: ")
            faculty.add_publication(title, year, journal)
            print("Publication added successfully!")

        elif choice == '3':
            faculty.display_faculty_info()

        elif choice == '4':
            print("\nExiting program. Goodbye!")
            break

        else:
            print("\nInvalid choice! Please try again.")

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

```
Enter faculty name: Diksha Ma'am
Enter age: 32
Enter gender: Female
Enter department: AI
Enter designation: Asst. Teacher

=== Faculty Management System ===
1. Add Course
2. Add Publication
3. Display Faculty Information
4. Exit

Enter your choice (1-4): 3

Person Information:
--------------------
Name: Diksha Ma'am
Age: 32
Gender: Female

Faculty Information:
--------------------
Department: AI
Designation: Asst. Teacher

No publications available.

=== Faculty Management System ===
1. Add Course
2. Add Publication
3. Display Faculty Information
4. Exit

Enter your choice (1-4): 4

Exiting program. Goodbye!
```

Q5. Write a program that overloads the + operator so that it can add a specified number of days to a given date.

```python
class Date:
    # Number of days in each month (non-leap year)
    DAYS_IN_MONTH = [0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

    def __init__(self, day, month, year):
        if not self.is_valid_date(day, month, year):
            raise ValueError("Invalid date")
        self.day = day
        self.month = month
        self.year = year

    def is_leap_year(self, year):
        """Check if the given year is a leap year"""
        return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)

    def get_days_in_month(self, month, year):
        """Get the number of days in a given month"""
        if month == 2 and self.is_leap_year(year):
            return 29
        return self.DAYS_IN_MONTH[month]

    def is_valid_date(self, day, month, year):
        """Validate the date"""
        if not (1 <= month <= 12):
            return False
        if not (1 <= day <= self.get_days_in_month(month, year)):
            return False
        if year < 1:  # Assuming we don't handle dates before year 1
            return False
        return True

    def __add__(self, days):
        """Overload + operator to add days to the date"""
        if not isinstance(days, int):
            raise TypeError("Can only add integer number of days")
        if days < 0:
            raise ValueError("Cannot add negative days")

        new_day = self.day
        new_month = self.month
        new_year = self.year

        while days > 0:
            # Days left in current month
            days_in_current_month = self.get_days_in_month(new_month, new_year)
            days_left_in_month = days_in_current_month - new_day + 1

            if days < days_left_in_month:
                # If remaining days fit in current month
                new_day += days
                days = 0
            else:
                # Move to next month
                days -= days_left_in_month
                new_day = 1
                new_month += 1

                # Handle year rollover
                if new_month > 12:
                    new_month = 1
                    new_year += 1

        return Date(new_day, new_month, new_year)

    def __str__(self):
        """Return string representation of the date"""
        return f"{self.day:02d}/{self.month:02d}/{self.year}"

def main():
    while True:
        try:
            print("\n=== Date Calculator ===")
            print("Enter date (or press Enter to exit):")
            date_input = input("Date (DD/MM/YYYY): ")

            if not date_input:
                print("Goodbye!")
                break

            # Parse date input
            day, month, year = map(int, date_input.split('/'))
            date = Date(day, month, year)

            # Get number of days to add
            days = int(input("Enter number of days to add: "))

            # Calculate and display new date
            new_date = date + days
            print(f"\nOriginal date: {date}")
            print(f"After adding {days} days: {new_date}")

        except ValueError as e:
            print(f"Error: {e}")
        except Exception as e:
            print(f"An error occurred: {e}")

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

```
=== Date Calculator ===
Enter date (or press Enter to exit):
Date (DD/MM/YYYY): 25/11/2024
Enter number of days to add: 5

Original date: 25/11/2024
After adding 5 days: 30/11/2024

=== Date Calculator ===
Enter date (or press Enter to exit):
Date (DD/MM/YYYY):
Goodbye!
```

Q6. Write a program to overload -= operator to subtract two Distance objects

```
class Distance:
    def __init__(self, feet=0, inches=0):
        """Initialize distance with feet and inches"""
        self.feet = feet
        self.inches = inches
        self.normalize()

    def normalize(self):
        """Convert excess inches to feet"""
        # Convert excess inches to feet
        if self.inches >= 12:
            self.feet += self.inches // 12
            self.inches = self.inches % 12
        # Handle negative values
        elif self.inches < 0:
            while self.inches < 0:
                self.feet -= 1
                self.inches += 12

    def __isub__(self, other):
        """Overload -= operator to subtract two distances"""
        if not isinstance(other, Distance):
            raise TypeError("Can only subtract Distance objects")

        # Convert both distances to inches for easier subtraction
        total_inches1 = self.feet * 12 + self.inches
        total_inches2 = other.feet * 12 + other.inches

        # Perform subtraction
        result_inches = total_inches1 - total_inches2

        if result_inches < 0:
            raise ValueError("Result cannot be negative")

        # Convert back to feet and inches
        self.feet = result_inches // 12
        self.inches = result_inches % 12

        return self

    def __str__(self):
        """Return string representation of the distance"""
        if self.feet == 0:
            return f"{self.inches} inches"
        elif self.inches == 0:
            return f"{self.feet} feet"
        else:
            return f"{self.feet} feet {self.inches} inches"

def get_distance_input(prompt):
    """Helper function to get feet and inches input"""
    while True:
        try:
            print(prompt)
            feet = int(input("Enter feet: "))
            inches = int(input("Enter inches: "))
            if feet < 0 or inches < 0:
                print("Distance cannot be negative!")
                continue
            return Distance(feet, inches)
        except ValueError:
            print("Please enter valid numbers!")

def main():
    while True:
        print("\n=== Distance Calculator ===")
        print("1. Subtract Distances")
        print("2. Exit")

        choice = input("\nEnter your choice (1-2): ")
        if choice == '1':
            try:
                # Get first distance
                d1 = get_distance_input("\nEnter first distance:")
                print(f"First distance: {d1}")

                # Get second distance
                d2 = get_distance_input("\nEnter second distance:")
                print(f"Second distance: {d2}")

                # Perform subtraction using -=
                print("\nPerforming d1 -= d2...")
                d1 -= d2

                print(f"\nResult: {d1}")

            except ValueError as e:
                print(f"Error: {e}")
            except Exception as e:
                print(f"An error occurred: {e}")

        elif choice == '2':
            print("\nGoodbye!")
            break

        else:
            print("\nInvalid choice! Please try again.")

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

```
=== Distance Calculator ===
1. Subtract Distances
2. Exit

Enter your choice (1-2): 28

Invalid choice! Please try again.

=== Distance Calculator ===
1. Subtract Distances
2. Exit

Enter your choice (1-2): 1

Enter first distance:
Enter feet: 28
Enter inches: 5
First distance: 28 feet 5 inches

Enter second distance:
Enter feet: 20
Enter inches: 11
Second distance: 20 feet 11 inches

Performing d1 -= d2...

Result: 7 feet 6 inches

=== Distance Calculator ===
1. Subtract Distances
2. Exit

Enter your choice (1-2): 2

Goodbye!
```