1. Compare and contrast the float and Decimal classes' benefits and drawbacks.

Ans: - Float and Decimal classes:

Float: It is a binary floating-point number that can represent values between integers and has a much greater range of values. However, operations with floats can be slightly slower than with integers, and precision can be lost1.
Decimal: It is designed for more precise decimal arithmetic, suitable for financial and monetary calculations. However, it might be slower than float for some operations.

2. Decimal ('1.200') and Decimal ('1.2') are two objects to consider. In what sense are these the same object? Are these just two ways of representing the exact same value, or do they correspond to different internal states?

Ans: - Decimal ('1.200') and Decimal ('1.2'): These are two different representations of the same numerical value. They are considered the same in terms of their numerical value, but they do have different internal states due to the different number of trailing zeros.

3. What happens if the equality of Decimal ('1.200') and Decimal ('1.2') is checked?

Ans- **Equality check**: When the equality of Decimal ('1.200') and Decimal ('1.2') is checked, it would return True because they represent the same numerical value, despite the difference in representation.

4. Why is it preferable to start a Decimal object with a string rather than a floating-point value?

Ans: - Starting a Decimal object with a string: It's preferable to start a Decimal object with a string rather than a floating-point value because converting directly from a float can cause the Decimal object to inherit all of the imprecision that we were trying to avoid in the first place.

5. In an arithmetic phrase, how simple is it to combine Decimal objects with integers?

Ans: - Combining Decimal objects with integers: Decimal objects can be easily combined with integers in arithmetic expressions. The integer is implicitly converted to a Decimal before the operation.

6. Can Decimal objects and floating-point values be combined easily?

Ans: - Combining Decimal objects and floating-point values: Combining Decimal objects and floating-point values is not straightforward because they are different types. One must be converted to the other type before they can be combined.

7. Using the Fraction class but not the Decimal class, give an example of a quantity that can be expressed with absolute precision.

Ans: - Example with Fraction class: An example of a quantity that can be expressed with absolute precision using the Fraction class is 1/2. This represents the exact value of one-half, which is not always exactly representable as a floating-point or decimal number due to the limitations of these formats.

8. Describe a quantity that can be accurately expressed by the Decimal or Fraction classes but not by a floating-point value.

Ans: - Decimal or Fraction classes vs floating-point value: A simple example is 1/3. While this can be exactly represented by the Fraction class (as Fraction (1, 3)) or as a Decimal given enough precision, it cannot be exactly represented as a floating-point value due to the limitations of binary representation.

Q9. Consider the following two fraction objects: Fraction (1, 2) and Fraction(1, 2). (5, 10). Is the internal state of these two objects the same? Why do you think that is?

Ans: - Fraction (1, 2) and Fraction (5, 10): These two objects represent the same numerical value (one-half), so in that sense, their internal state is the same. The difference lies in the specific numbers used for the numerator and denominator.

Q10. How do the Fraction class and the integer type (int) relate to each other? Containment or inheritance?

Ans:- Fraction class and integer type (int): The Fraction class in Python is not inherited from the integer type (int), but rather it contains integer values that represent the numerator and denominator of the fraction.