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

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?

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

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

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

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

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

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

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?

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

1. The float and Decimal classes are both used to represent numbers with decimal places in Python. The float class uses binary floating-point arithmetic, which can result in inaccuracies due to the limitations of representing decimal numbers in binary. The Decimal class uses base-10 arithmetic and provides more control over rounding and precision. The benefits of the Decimal class are its greater precision and control over rounding, while its drawbacks include slower performance and higher memory usage compared to the float class.
2. Decimal('1.200') and Decimal('1.2') represent the same value, but they have different internal states because they are different objects. The first string specifies a precision of 3 decimal places, while the second string specifies a precision of 1 decimal place. When the two objects are compared using the == operator, they will be considered equal because they represent the same value.

3. If the equality of Decimal('1.200') and Decimal('1.2') is checked using the == operator, the result will be True because they represent the same value. However, if the identity of the objects is checked using the is operator, the result will be False because they are different objects.
4. It is preferable to start a Decimal object with a string rather than a floating-point value because floating-point values can have inaccuracies due to the limitations of binary representation. When a floating-point value is converted to a Decimal object, the inaccuracies can be propagated, whereas starting with a string representation ensures that the value is accurately represented in base-10.
5. It is simple to combine Decimal objects with integers in an arithmetic phrase, because Decimal objects can be implicitly converted to integers when necessary. For example, the expression Decimal('1.2') + 3 would result in a Decimal object with the value 4.2.
6. Decimal objects and floating-point values can be combined, but the result may not be accurate due to the limitations of binary representation. When a Decimal object is combined with a floating-point value, the result will be a float object, which may not accurately represent the original values.
7. One example of a quantity that can be expressed with absolute precision using the Fraction class is the fraction 1/3. This value has a repeating decimal representation and cannot be represented exactly as a decimal or float value, but can be represented exactly as a fraction with a numerator and denominator.
8. A quantity that can be accurately expressed by the Decimal or Fraction classes but not by a floating-point value is a monetary value that requires exact precision, such as $1.99. If this value is represented as a float, there may be rounding errors that result in a different value being used in calculations.
9. The internal state of Fraction(1, 2) and Fraction(1, 2, 5, 10) is not the same, even though they represent the same value of 1/2. The second fraction includes additional factors of 5 and 10 in the numerator and denominator, which can be cancelled out to simplify to 1/2. However, the internal state of the object still includes these factors, which can affect how the object is displayed and how it interacts with other objects.
10. The Fraction class and the integer type (int) are not related by containment or inheritance. However, the Fraction class can be used to represent exact fractions that may not be representable as integers, while integers can be used as inputs to create Fraction objects.