**1. Compare and contrast the float and Decimal classes&#39; benefits and drawbacks.**

**2. Decimal(&#39;1.200&#39;) and Decimal(&#39;1.2&#39;) 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(&#39;1.200&#39;) and Decimal(&#39;1.2&#39;) 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?**

***SOLUTIONS***

1. *The float and Decimal classes in Python are both used to represent decimal numbers, but they differ in terms of precision and accuracy. Floats are represented in binary format and have limited precision, which can lead to rounding errors. Decimal, on the other hand, has arbitrary precision and is represented in decimal format, allowing for more accurate calculations. The main drawback of using Decimal is that it is slower and consumes more memory than using floats.*
2. *Decimal('1.200') and Decimal('1.2') represent the same value but have different internal states. The Decimal class stores the exact value of the number and its context, which includes information about the precision and rounding rules. In this case, the two objects have different internal states because the first has a precision of three decimal places and the second has a precision of one decimal place.*
3. *The equality of Decimal('1.200') and Decimal('1.2') is checked based on their internal states, which are different in this case. Therefore, the result of the equality check will be False.*
4. *Starting a Decimal object with a string rather than a floating-point value is preferable because it ensures that the number is represented exactly as intended, without any rounding errors. When a floating-point value is used, it may be rounded to fit the available precision, leading to inaccuracies.*
5. *It is very simple to combine Decimal objects with integers in an arithmetic phrase, as they are both numeric types that can be added, subtracted, multiplied, or divided using standard arithmetic operators.*
6. *Decimal objects and floating-point values can be combined, but care should be taken to avoid rounding errors. When a Decimal object is combined with a float, the result will be a float, which may introduce inaccuracies.*
7. *An example of a quantity that can be expressed with absolute precision using the Fraction class is 1/3. This is because 1/3 has an infinite repeating decimal representation, but can be represented exactly as a fraction.*
8. *A quantity that can be accurately expressed by the Decimal or Fraction classes but not by a floating-point value is 0.1. This is because 0.1 has a repeating binary representation in floats, which can lead to rounding errors and inaccuracies.*
9. *The internal state of Fraction(1, 2) and Fraction(1, 2, 5, 10) is different because they represent different fractions, even though they have the same value. The second fraction has a denominator of 10, which is not reduced to its lowest terms, whereas the first fraction is already in its simplest form.*
10. *The Fraction class and the integer type (int) are related through inheritance. The Fraction class is a subclass of the abstract base class Rational, which defines the interface for rational numbers. Integers are a specific type of rational number with a denominator of 1, so they can be used wherever a rational number is expected.*