**Assignment - 20**

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

Ans: The float and Decimal classes both represent floating-point numbers, but they differ in their precision and handling of arithmetic operations. Floats are implemented using binary floating-point arithmetic and have limited precision, leading to potential rounding errors. Decimals, on the other hand, offer arbitrary precision decimal arithmetic, making them suitable for financial and high-precision calculations. However, Decimals are typically slower and consume more memory compared to floats.

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') are two distinct objects representing the same value. While they represent the same numerical value, they are different objects with potentially different internal states, as they may store different representations of the number internally.

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

Ans: Checking the equality of Decimal('1.200') and Decimal('1.2') would result in False, as they are two distinct Decimal objects with potentially different internal representations, despite representing the same numerical value.

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

Ans: It is preferable to start a Decimal object with a string rather than a floating-point value to avoid any potential precision loss or rounding errors associated with floating-point representations. Starting with a string ensures that the Decimal object accurately represents the specified value without any loss of precision.

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

Ans: Combining Decimal objects with integers in arithmetic operations is straightforward and simple, as Decimal objects support arithmetic operations with integers without loss of precision.

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

Ans: Decimal objects and floating-point values can be combined, but it's important to note that arithmetic operations between Decimal objects and floats may result in potential loss of precision due to the inherent limitations of floating-point arithmetic.

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

Ans: Using the Fraction class, an example of a quantity that can be expressed with absolute precision is 1/3, as it cannot be represented exactly using a finite number of decimal digits but can be represented accurately as a fraction.

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

Ans: A quantity that can be accurately expressed by the Decimal or Fraction classes but not by a floating-point value is the result of dividing 1 by 3 (1/3), as it requires an infinite number of decimal places to represent it accurately, which is not possible with finite-precision floating-point arithmetic.

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: The internal state of Fraction(1, 2) and Fraction(5, 10) objects is the same. Fractions are reduced to their simplest form, so both objects represent the same value of 1/2.

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

Ans: The Fraction class and the integer type (int) are related through containment. Fractions can represent integers accurately, and integers can be used as arguments to create Fraction objects. However, Fraction objects are not inherited from the integer type.