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

Floating point data type represent number values with fractional parts. Decimal accurately represent any number within the precision of the decimal format, whereas Float cannot accurately represent all numbers.

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?

Decimal “is based on a floating-point model which was designed with people in mind, and necessarily has a paramount guiding principle – computers must provide an arithmetic that works in the same way as the arithmetic that people learn at school.” – excerpt from the decimal arithmetic specification.

Decimal numbers can be represented exactly. In contrast, numbers like 1.1 and 2.2 do not have exact representations in binary floating point. End users typically would not expect 1.1 + 2.2 to display as 3.3000000000000003 as it does with binary floating point.

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

The Decimal class provides some mathematical operations such as sqrt and log. However, it doesn’t have all the functions defined in the math module.

When you use functions from the math module for decimal numbers, Python will cast the Decimal objects to floats before carrying arithmetic operations. This results in losing the precision built in the decimal objects.

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

Decimal objects cannot generally be combined with floats or instances of fractions.

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.

Both binary and decimal floating point are implemented in terms of published standards. While the built-in float type exposes only a modest portion of its capabilities, the decimal module exposes all required parts of the standard. When needed, the programmer has full control over rounding and signal handling. This includes an option to enforce exact arithmetic by using exceptions to block any inexact operations.

The decimal module was designed to support “without prejudice, both exact unrounded decimal arithmetic (sometimes called fixed-point arithmetic) and rounded floating-point arithmetic.” – excerpt from the decimal arithmetic specification.

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?

In Python the Fraction module supports rational number arithmetic. Using this module, we can create fractions from integers, floats, decimal and from some other numeric values and strings.

There is a concept of Fraction Instance. It is formed by a pair of integers as numerator and denominator.

The class fractions.Fractionis used to create a Fraction object. It takes Numerator and Denominator. The default value of the numerator is 0 and denominator is 1. It raises ZeroDivisionError when the denominator is 0.