ASSIGNMENT 20

# Q1

The float class in Python represents floating-point numbers and uses binary floating-point arithmetic. It has the benefit of being efficient and widely supported by hardware and software. However, it can suffer from precision and rounding issues, leading to small inaccuracies in calculations. On the other hand, the Decimal class from the decimal module provides decimal floating-point arithmetic with adjustable precision. It allows for precise decimal calculations and avoids rounding errors but can be slower and consumes more memory compared to float.

# Q2

The expressions Decimal('1.200') and Decimal('1.2') represent the same value, but they correspond to different internal states. Internally, the Decimal class stores the exact decimal representation of the number, including trailing zeros. When comparing them for equality, they would be considered equal since their values are the same.

# Q3

If the equality of Decimal('1.200') and Decimal('1.2') is checked using the equality operator (==), the result will be True. The Decimal class compares the values, taking into account the internal precision and trailing zeros.

# Q4

It is preferable to start a Decimal object with a string rather than a floating-point value to ensure precision and avoid any potential rounding errors. When a floating-point value is converted to a Decimal object directly, it may introduce small inaccuracies due to the limitations of binary floating-point representation.

# Q5

Combining Decimal objects with integers in arithmetic operations is straightforward. The Decimal class supports arithmetic operations with integers, and the result will be a Decimal object with the desired precision.

# Q6

Decimal objects and floating-point values can be combined in arithmetic operations, but there can be precision issues. When a Decimal object is combined with a floating-point value, the result will be a Decimal object, but the precision may be affected by the limitations of binary floating-point representation.

# Q7

The Fraction class can express quantities with absolute precision. For example, Fraction (1, 3) represents one-third precisely without any rounding or approximation issues.

# Q8

A quantity that can be accurately expressed by the Decimal or Fraction classes but not by a floating-point value is a repeating decimal. For example, the value of 1/3 as a decimal (0.3333...) cannot be represented exactly by a float, but it can be represented precisely using the Decimal or Fraction class.

# Q9

No, the internal state of the two fraction objects is not the same. The first fraction object is created with the constructor Fraction(1, 2), which creates a fraction with a numerator of 1 and a denominator of 2. The second fraction object is created with the constructor Fraction(1, 2). (5, 10), which creates a fraction with a numerator of 1 and a denominator of 10.

The internal state of a fraction object is a tuple that contains the numerator and denominator of the fraction. The first fraction object has an internal state of (1, 2), while the second fraction object has an internal state of (1, 10).

The reason why the internal state of the two fraction objects is not the same is because the second fraction object is created using the constructor Fraction(numerator, denominator). (new\_numerator, new\_denominator), which creates a new fraction object with the same numerator as the original fraction object, but with the new denominator.

In this case, the new denominator is 10, so the second fraction object has an internal state of (1, 10).

# Q10

The Fraction class and the integer type (int) are related through inheritance. The Fraction class is a subclass of the int type, meaning it inherits the properties and methods of the int type while adding additional functionality specific to fractions.