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

**Ans : Example of float value**

**a = 0.35**

**print(type(a))**

**###Float objects like this one are always represent internally up to a certain degree of accuracy only.This becomes evident when adding 0.1 to a:###**

**print(a+0.1)**

**###The reason for this is that floats are internally represented in binary format;that is, a decimal number 0 < n < 1 is represented by a series of the form .For certain floating-point numbers the binary representation might involve a large number of elements or might even be an infinite series. However, given a fixed number of bits used to represent such a number—i.e., a fixed number of terms in the representation series—inaccuracies are the consequence. Other numbers can be represented perfectly and are therefore stored exactly even with a finite number of bits available.###**

**# Example is  :**

**b=0.5**

**print(b.as\_integer\_ratio()) #i.e. 0.5 is exactly represented as 1/2**

**# incase of b=0.35**

**b=0.35**

**print(b.as\_integer\_ratio()) # Here teh b=0.35 is not exact**

**### The precision is dependent on the number of bits used to represent the number.Hence some error can creep in. It translates 15 digits accuracy for float.For example, the issue can be of importance when summing over a large set of numbers.**

**In such a situation, a certain kind and/or magnitude of representation error might,in aggregate, lead to significant deviations from a benchmark value.In order to overcome this we use the decimal module.The module decimal provides an arbitrary-precision object for floating-point numbers and several options to address precision issues when working with such numbers:###**

**import decimal**

**from decimal import Decimal**

**print(decimal.getcontext())**

**d = Decimal(1) / Decimal (55)**

**print("d=",d)**

**decimal.getcontext().prec = 5  # lower precision**

**e=Decimal(1)/Decimal(55)**

**print("e=",e)**

**###The precision can in this way be adjusted to the exact problem at hand and one can operate with floating-point objects that exhibit different degrees of accuracy.###**

**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 :**

**###Though they are identical value wise, internal representation or internal state of these two values are different, as they have decimal values of different precisions.###**

**print(Decimal('1.200'))**

**print(Decimal('1.2'))**

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

**Ans : It returns that the values are stored are True.**

**print(Decimal('1.200') == Decimal('1.2'))**

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

**Ans :**

**###Floating-point value is converted to Decimal format. Decimal can store float value with absolute precision. But when float value is given as Decimal object,it first has to be converted from floating point value which might already have rounding error.Hence it is preferable to start a Decimal object with a string.**

**Example is as follows : ###**

**import decimal**

**from decimal import Decimal**

**a=Decimal(0.3) #0.3 is a float**

**print(a) # It is not stored exactly as 0.3 but as printed.**

**b=Decimal('0.3') #'0.3' is a string**

**print(b) #It exactly prints 0.3, precisely, with correct precision.**

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

**Ans :**

**import decimal**

**from decimal import Decimal**

**val=2**

**print(type(val))**

**a=Decimal(val)**

**print(a)**

**b=a\*Decimal(val)**

**print(b)**

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

**Ans : Arithmetic operfations like Adding,subtracting or multiplying a Decimal object by a floating-point value is**

**generates an error. To do these operations, the floating point has to be converted to a Decimal object—for example, converting from a floating-point value and then rounding. Else, arithmetic operations between the two types cause runtime errors.**

**import decimal**

**from decimal import Decimal**

**val=0.3**

**b=Decimal(val)**

**print("Muliplication:",Decimal('0.2')\*b ) # Multiplying a decimal object with float value converted to Decimal**

**print(Decimal('0.2')\*val)# Error when Decimal obeject is multiplied by a float**

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

**Ans : Example**

**from fractions import Fraction**

**val=0.5**

**fr=Fraction(val)**

**print(fr)**

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

**Ans :**

**d=Decimal('0.1') \* Decimal('0.1')**

**print("decimal=",d)**

**frac=Fraction('1/10') \* Fraction('1/10')**

**print("fraction=",frac)**

**fl=0.1\*0.1**

**print("float value=",fl)**

**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 :**

**from fractions import Fraction**

**frac1=Fraction(1, 2)**

**print(frac1)**

**frac2=Fraction(5, 10)**

**print(frac2)**

**if (frac1 == frac2 ):**

**print('Fraction (1,2) and Fraction(5,10) are equal')**

**### The internal state of both are same as Fraction(5,10) is reduced to simplest form. Hence 1/2 is printed in both the cases.###**

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

**Ans : Fraction class and integer type(int) are related in form of a container.It contains two ints, one the numerator and the other the denominator.**

**from fractions import Fraction**

**frac = Fraction(1,2)**

**print('numerator is', frac.numerator,type(frac.numerator))**

**print('denominator is', frac.denominator,type(frac.numerator) )**