

Python_advance_assignment_20

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

In []:

Ans: Both the `float` and decimal types store numerical values `in` Python.

Use floats when convenience `and` speed matter. A `float` gives you an approximation of the number you declare.

Use decimals when precision matters. Decimals can suffer `from` their own precision issues, but generally, decimals are more precise than floats. The performance difference between `float` and decimal, `with` Python 3, `is not` outlandish, `and in` my experience, the precision benefits of a decimal outweigh the performance benefits of a `float`.

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 ?

In []:

Ans: Both values are same but internal representation at storage is different. Precision differs, `Decimal('1.200')` gives internally `1.200` and `Decimal('1.2')` gives `1.2`.

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

In []:

Ans: Both values are checked to be equal , they only differ `in` precision.

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

In []:

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.

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

In []:

Ans: We can do it **with** use of Decimal().

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

In []:

Ans: Arithmetic operations 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.

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

In []:

Ans: Value of **0.5** will be represented **as** $\frac{1}{2}$.

8. 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 ?

In []:

Ans: Both will be reduced to $\frac{1}{2}$

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

In []:

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.