**Python Advance Assignment-20**

**1. Compare and contrast the advantages and disadvantages of the float and Decimal classes.**

Ans.

**Decimal class**

* Provides support for fast correctly-rounded decimal floating point arithmetic.
* It gives the provison to represent the number in a representative way so that it is clearly undertood by the user.
* give you the exact arithmatic operation result like we use to do normally,
* Object exposes all required parts of the standard. When needed, we can change the rounding and signal handling. This includes an option to enforce exact arithmetic by using exceptions to block any inexact operations.
* Objects share many properties with the other built-in numeric types such as float and int. All of the usual math operations and special methods apply.
* It can be constructed from integers, strings, floats, or tuples. Construction from an integer or a float performs an exact conversion of the value of that integer or float.
* It is faster way to get precise value which we human work and have a smaller range than the float
* Decimal types are better suited for financial and monetary calculations
* The Decimal has a default precision of 28 places

**Floating class**

* It do not represent the exact precise value value.
* It gives the value close to the result.
* The float object exposes only a modest portion of its capabilities
* Some values cannot be exactly represented in a float data type. For instance, storing the 0.1 value in float (which is a binary floating point value) variable we get only an approximation of the value. Similarly, the 1/3 value cannot be represented exactly in decimal floating point type.
* Float types is best suited for scientific calculations
* The float has default precision of 18 places.

**2. Consider two objects: Decimal('1.200') and Decimal('1.2'). 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 [154]:

**from** decimal **import** Decimal

d1**=**Decimal('1.200')

d2**=**Decimal('1.2')

In [155]:

d1 **==** d2

Out[155]:

True

In [156]:

d1 **is** d2

Out[156]:

False

In [157]:

print(d1)

1.200

In [158]:

print(d2)

1.2

By looking at the value it is not equal. This can be seen if we use **is** operator to check for d1 and d2. If we print the d1 and d2 we do not get the same value but still it will shwo the equality this is because the normalised value of both the object is same. But both the objects are same as they both give the same normalised value.

Another way to represent them is in the form of **normlaization**

**3. What happens if Decimal('1.200') and Decimal('1.2') are tested for equality**

In [63]:

**from** decimal **import** Decimal

d1**=**Decimal(str(float('1.200')))

d2**=**Decimal('1.2')

We can test for the normalised value of both te objects. In this way we could get the ans

In [148]:

d11**=**d1**.**normalize()

print(d11)

1.2

In [149]:

d22**=**d2**.**normalize()

print(d22)

1.2

In [162]:

d11 **==** d22

Out[162]:

True

**4. Why is it usually better to initialize a Decimal object from a string than from a floating-point value**

Ans.

Because while initilising with a string, this will give the exact value of the number available in the quotes ' '. While in case of floating point value, it first has to convert from float then using the round operation we will get the result with the desired precision

**5. How easy is it to combine Decimal objects with integers in an arithmetic expression?**

Ans.

It is that much easy as we do normal calculation. How ever if we do not specify the number as a string then in order to get in decimal form we fisrt need to convert the value from float into decimal object and use the round method to get the precise value.

In [77]:

o1**=**Decimal('10.00')

print(o1**+**10)

20.00

In [99]:

d**=**Decimal(10)

print(round(d**+**10,2))

20.00

In [86]:

f**=**100

print(round(f,4))

*#here we don not have specified the Decimal class so here we do not get the precise value*

100

**6. How easy is it to combine Decimal objects and floating-point values?**

In [105]:

d1**=**Decimal(10)

f**=**10.0

print(d1**+**f)

**---------------------------------------------------------------------------**

**TypeError** Traceback (most recent call last)

**<ipython-input-105-1d41578a3c2e>** in <module>

1 d1**=**Decimal**(10)**

2 f**=10.0**

**----> 3** print**(**d1**+**f**)**

**TypeError**: unsupported operand type(s) for +: 'decimal.Decimal' and 'float'

As we see in the above example combining decimal object with floating value gives an error. To remove this erro we need to first convert the floating point value into decimal object by using using Decimal class and then we can use round function to get a value precised upto given number.

example is given below

In [104]:

d1**=**Decimal(10)

f**=**10.0

print(round(d1**+**Decimal(f),1))

20.0

**7. Give an example of a quantity that can be represented with absolute precision by using the Fraction class but not the Decimal class.**

In [106]:

d**=**0.1**+**0.1**+**0.1

print(d)

0.30000000000000004

In [107]:

d**=**Decimal(0.1)**+**Decimal(0.1)**+**Decimal(0.1)

print(d)

0.3000000000000000166533453694

Here we see that the quantity that can be represemted with abosulte precision by using fraction value is 18 while decimal class it has 28

**8. Give an example of a quantity that can be represented exactly by either the Decimal or Fraction class but not by a floating-point value.**

In [182]:

d1**=**Decimal('1.000000')

print(d1)

1.000000

In [184]:

d2**=**1.000000

print(d2)

1.0

**9. Consider two Fraction objects: Fraction(1, 2) and Fraction(5, 10). Do these two objects have the same internal state? Why or why not?**

In [188]:

**from** fractions **import** Fraction

In [191]:

f1**=**Fraction(1,2)

print(f1)

1/2

In [193]:

f2**=**Fraction(5,10)

print(f2)

1/2

Internal state of both the objects are not same. Though they give the same out output. It is because the fraction class will calculate for the further simplication till the result remains in fraction form

**10. What is the relationship between the Fraction class and the integer type (int)? Containment or inheritance?**

Ans. Fraction class contains the pair of intgers values as numerator and denomenator and gives the result in the form of fraction. Since the values are belong to integer class an these class object are using by another object of fraction class. So here Containment is happening.

In [196]:

n**=**10

d**=**7

f**=**Fraction(n,d)

print(f)

10/7