******

Инструкцию по использованию модулей fractions, decimal.

**Модуль fractions**

Для начала в заголовке файла from fractions import \*, from decimal import \*.

Модуль fractions предоставляет поддержку рациональных чисел.

class **fractions.Fraction**(numerator=0, denominator=1)

class **fractions.Fraction**(other\_fraction)

class **fractions.Fraction**(float)

class **fractions.Fraction**(decimal)

class **fractions.Fraction**(string)

Класс, представляющий собой рациональные числа. Экземпляр класса можно создать из пары чисел (числитель, знаменатель), из другого рационального числа, числа с плавающей точкой, числа типа decimal. Decimal, и из строки, представляющей собой число.

**Fraction.limit\_denominator**(max\_denominator=1000000) - ближайшее рациональное число со знаменателем не больше данного.

Также, помимо класса рациональных чисел, модуль fractions предоставляет функцию для нахождения наибольшего общего делителя.

**fractions.gcd**(a, b) - наибольший общий делитель чисел a и b.

**Модуль decimal**

Проблему может решить использование функции round(), которая округлит число. Однако есть и другой способ, который заключается в использовании встроенного модуля decimal.

Ключевым компонентом для работы с числами в этом модуле является класс Decimal. Для его применения нам надо создать его объект с помощью конструктора. В конструктор передается строковое значение, которое представляет число:

from decimal import Decimal

number = Decimal("0.1")

После этого объект Decimal можно использовать в арифметических операциях:

from decimal import Decimal

number = Decimal("0.1")

number = number + number + number

print(number)

В операциях с Decimal можно использовать целые числа:

number = Decimal("0.1")

number = number + 2

Однако нельзя смешивать в операциях дробные числа float и Decimal:

number = Decimal("0.1")

number = number + 0.1 # здесь возникнет ошибка

С помощью дополнительных знаков мы можем определить, сколько будет символов в дробной части числа:

number = Decimal("0.10")

number = 3 \* number

print(number) # 0.30

Строка "0.10" определяет два знака в дробной части, даже если последние символы будут представлять ноль. Соответственно "0.100" представляет три знака в дробной части.

Округление чисел

Объекты Decimal имеют метод quantize(), который позволяет округлять числа. В этот метод в качестве первого аргумента передается также объект Decimal, который указывает формат округления числа:

from decimal import Decimal

number = Decimal("0.444")

number = number.quantize(Decimal("1.00"))

print(number) # 0.44

number = Decimal("0.555678")

print(number.quantize(Decimal("1.00"))) # 0.56

number = Decimal("0.999")

print(number.quantize(Decimal("1.00"))) # 1.00

Используемая строка "1.00" указывает, что округление будет идти до двух знаков в дробной части.

По умолчанию округление описывается константой ROUND\_HALF\_EVEN, при котором число округляется в большую сторону, если оно нечетное, а предыдущее перед ним больше 4. Например:

from decimal import Decimal, ROUND\_HALF\_EVEN

number = Decimal("10.025")

print(number.quantize(Decimal("1.00"), ROUND\_HALF\_EVEN)) # 10.02

number = Decimal("10.035")

print(number.quantize(Decimal("1.00"), ROUND\_HALF\_EVEN)) # 10.04

Стратегия округления передается в качестве второго параметра в quantize.

Строка "1.00" означает, что округление будет идти до двух чисел в дробной части. Но в первом случае "10.025" - вторым знаком идет 2 - четное число, поэтому, несмотря на то, что следующее число 5, двойка не округляется до тройки.

Во втором случае "10.035" - вторым знаком идет 3 - нечетное число, поэтому оно округляется до 4.

Данное поведение при округлении, возможно, не всем покажется желательным, и в этом случае его можно переопределить, использовав одну из следующих констант:

ROUND\_HALF\_UP: округляет число в сторону повышения, если после него идет число 5 или выше

ROUND\_HALF\_DOWN: округляет число в сторону повышения, если после него идет число больше 5

number = Decimal("10.026")

print(number.quantize(Decimal("1.00"), ROUND\_HALF\_DOWN)) # 10.03

number = Decimal("10.025")

print(number.quantize(Decimal("1.00"), ROUND\_HALF\_DOWN)) # 10.02

ROUND\_05UP: округляет только 0 до единицы, если после него идет 5

number = Decimal("10.005")

print(number.quantize(Decimal("1.00"), ROUND\_05UP)) # 10.01

number = Decimal("10.025")

print(number.quantize(Decimal("1.00"), ROUND\_05UP)) # 10.02

ROUND\_CEILING: округляет число в большую сторону вне зависимости от того, какое число идет после него

number = Decimal("10.021")

print(number.quantize(Decimal("1.00"), ROUND\_CEILING)) # 10.03

number = Decimal("10.025")

print(number.quantize(Decimal("1.00"), ROUND\_CEILING)) # 10.03

ROUND\_FLOOR: не округляет число вне зависимости от того, какое число идет после него

number = Decimal("10.021")

print(number.quantize(Decimal("1.00"), ROUND\_FLOOR)) # 10.02

number = Decimal("10.025")

print(number.quantize(Decimal("1.00"), ROUND\_FLOOR)) # 10.02