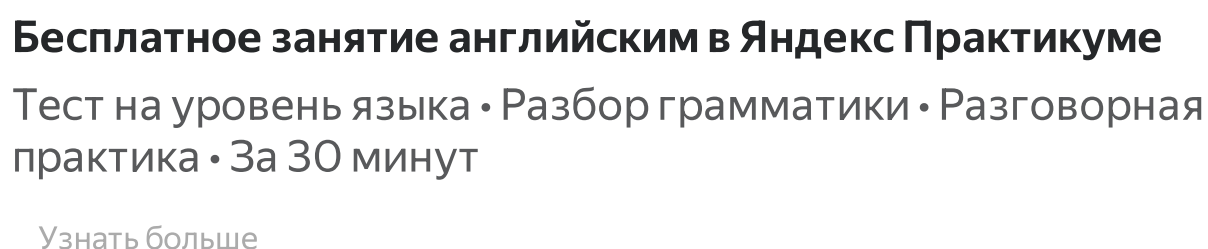


РЕКЛАМА • 18+



Узнать больше

/ Модуль decimal в Python, десятичная арифметика

поддержку быстрой правильно округленной десятичной арифметики с плавающей запятой. Он быстрее, чем `double`, по сравнению с [типом данных float](#):

о на модели с плавающей точкой, которая была разработана с учетом потребностей людей и
гепенный руководящий принцип - компьютеры должны обеспечивать арифметику, которая
метика, которую люди изучают в школе.

ть представлены точно. Напротив, числа типа float 1.1 и 2.2 не имеют точных формате. Конечные пользователи обычно не ожидают, что $1.1 + 2.2$ будет отображаться как

симметрию. В десятичном формате с плавающей запятой $0,1 + 0,1 + 0,1 - 0,3$ точно равно
результат равен $5.5511151231257827e-017$. Хотя значения близки к нулю, различия мешают
э равенство и различия могут накапливаться. По этой причине десятичная дробь

предпочтительнее в бухгалтерских приложениях, которые имеют строгие инварианты равенства.

- Десятичный модуль включает в себя понятие значимых мест, так что $1,30 + 1,20$ составляет $2,50$. Конечный ноль сохраняется для обозначения значимости. Это обычное представление для денежных приложений. Например $1,3 * 1,2$ дает $1,56$, а $1,30 * 1,20$ дает $1,5600$.
- В отличие от аппаратной двоичной плавающей запятой, модуль `decimal` имеет изменяемую пользователем точность, по умолчанию до 28 разрядов, которая может быть настолько большой, насколько это необходимо для вычислений:

```
from decimal import *
getcontext().prec = 6
Decimal(1) / Decimal(7)
# Decimal('0.142857')
getcontext().prec = 28
Decimal(1) / Decimal(7)
# Decimal('0.1428571428571428571428571428571429')
```

- И двоичные и десятичные числа с плавающей запятой реализованы в терминах опубликованных стандартов. В то время как [встроенный тип float](#) предоставляет лишь скромную часть своих возможностей, десятичный модуль `decimal` предоставляет все необходимые части стандарта. При необходимости программист полностью контролирует округление и обработку сигналов. Это включает в себя возможность применения точной арифметики с помощью исключений для блокировки любых неточных операций.
- Модуль `decimal` был разработан для поддержки без ущерба как точной необоснованной десятичной арифметики (иногда называемой арифметикой с фиксированной точкой), так и округленной арифметики с плавающей точкой.

Дизайн модуля основан на трех понятиях: десятичное число, контекст для арифметики и сигналы.

Десятичное число является неизменным. У него есть знак, цифры коэффициента и показатель степени. Для сохранения значимости цифры коэффициента не усекают конечные нули. Десятичные числа также включают специальные значения, такие как Infinity, -Infinity и NaN. Стандарт также отличает -0 от +0.

Контекст для арифметики - это среда, определяющая точность, [правила округления](#), ограничения на экспоненты, флаги, указывающие результаты операций, и средства активации, которые определяют, будут ли сигналы рассматриваться как истинные.

Сигналы - это группы исключительных условий, возникающих в процессе вычислений. В зависимости от потребностей приложения сигналы могут игнорироваться, рассматриваться как информационные или рассматриваться как исключения. Смотрите раздел "[Сигналы в модуле decimal](#)".

Для каждого сигнала есть флаг и активатор ловушек. Когда сигнал встречается, его флаг устанавливается в единицу, затем, если активатор прерывания установлен в единицу, возникает исключение. Флаги являются липкими, поэтому пользователь должен сбросить их перед мониторингом расчетов.

Содержание раздела:

- [КРАТКИЙ ОБЗОР МАТЕРИАЛА.](#)
- [Краткое руководство по модулю decimal](#)
- [Ошибки округления с повышенной точностью](#)
- [Способы работы с классом Decimal](#)
- [Класс Decimal\(\) модуля decimal](#)
- [Методы объекта Decimal\(\)](#)
- [Контексты модуля decimal](#)
- [Класс Context\(\) модуля decimal](#)
- [Режимы округления модуля decimal](#)
- [Сигнальные флаги модуля decimal](#)