Navigation

- index
- modules |
- next |
- previous |
- Python v2.6.4 documentation »
- The Python Standard Library »
- 10. Numeric and Mathematical Modules »

10.5. `fractions` — Rational numbers¶

New in version 2.6.

The `fractions` module provides support for rational number arithmetic.

A Fraction instance can be constructed from a pair of integers, from another rational number, or from a string.

class `fractions.Fraction`(*numerator=0*, *denominator=1*)¶
class `fractions.Fraction`(*other_fraction*)
class `fractions.Fraction`(*string*)

The first version requires that *numerator* and *denominator* are instances of `numbers.Integral` and returns a new `Fraction` instance with value `numerator/denominator`. If *denominator* is `0`, it raises a `ZeroDivisionError`. The second version requires that *other_fraction* is an instance of `numbers.Rational` and returns an `Fraction` instance with the same value. The last version of the constructor expects a string or unicode instance in one of two possible forms. The first form is:

```
[sign] numerator ['/' denominator]
```

where the optional `sign` may be either '+' or '-' and `numerator` and `denominator` (if present) are strings of decimal digits. The second permitted form is that of a number containing a decimal point:

```
[sign] integer '.' [fraction] | [sign] '.' fraction
```

where `integer` and `fraction` are strings of digits. In either form the input string may also have leading and/or trailing whitespace. Here are some examples:

```
>>> from fractions import Fraction
>>> Fraction(16, -10)
Fraction(-8, 5)
>>> Fraction(123)
Fraction(123, 1)
>>> Fraction()
Fraction(0, 1)
>>> Fraction('3/7')
Fraction(3, 7)
[40794 refs]
>>> Fraction(' -3/7 ')
Fraction(-3, 7)
>>> Fraction('1.414213 \t\n')
Fraction(1414213, 1000000)
>>> Fraction('-.125')
Fraction(-1, 8)
```

The `Fraction` class inherits from the abstract base class `numbers.Rational`, and implements all of the methods and operations from that class. `Fraction` instances are hashable, and should be treated as immutable. In addition, `Fraction` has the following methods:

`from_float`(*flt*)¶
This class method constructs a `Fraction` representing the exact value of *flt*, which must be a `float`. Beware that `Fraction.from_float(0.3)` is not the same value as `Fraction(3, 10)`

`from_decimal`(*dec*)¶
This class method constructs a `Fraction` representing the exact value of *dec*, which must be a `decimal.Decimal`.

`limit_denominator`(*max_denominator=1000000*)¶

Finds and returns the closest `Fraction` to `self` that has denominator at most max_denominator. This method is useful for finding rational approximations to a given floating-point number:

```
>>> from fractions import Fraction
>>> Fraction('3.1415926535897932').limit_denominator(1000)
```

```
Fraction(355, 113)
```

or for recovering a rational number that’s represented as a float:

```
>>> from math import pi, cos
>>> Fraction.from_float(cos(pi/3))
Fraction(4503599627370497, 9007199254740992)
>>> Fraction.from_float(cos(pi/3)).limit_denominator()
Fraction(1, 2)
```

`fractions.gcd`(*a*, *b*)¶

Return the greatest common divisor of the integers *a* and *b*. If either *a* or *b* is nonzero, then the absolute value of `gcd(a, b)` is the largest integer that divides both *a* and *b*. `gcd(a,b)` has the same sign as *b* if *b* is nonzero; otherwise it takes the sign of *a*. `gcd(0, 0)` returns `0`.

See also

Module `numbers`

The abstract base classes making up the numeric tower.

Previous topic

10.4. `decimal` — Decimal fixed point and floating point arithmetic

Next topic

10.6. `random` — Generate pseudo-random numbers

This Page

- Show Source

Navigation

- index
- modules |
- next |
- previous |
- Python v2.6.4 documentation »
- The Python Standard Library »
- 10. Numeric and Mathematical Modules »

Last updated on Feb 26, 2010. Created using Sphinx 0.6.3.