floatrange Documentation

Release 1.0

Laurent Pointal

September 13, 2015

Contents

floatrange - range() like with float numbers.

author Laurent Pointal <laurent.pointal@laposte.net>

license MIT

version 1.0

CHAPTER 1

Usage example

(note computation approximation and display with floats)

```
>>> from floatrange import floatrange
>>> floatrange(5)
floatrange(0.0, 5.0, 1.0)
>>> list(floatrange(5))
[0.0, 1.0, 2.0, 3.0, 4.0]
>>> list(floatrange(3.2,5.4,0.2))
[3.2, 3.4000000000000004, 3.6, 3.8000000000000003, 4.0, 4.2, 4.4,
4.6000000000000005, 4.800000000000001, 5.0, 5.2]
>>> 6 in floatrange(1,8)
True
>>> 6.1 in floatrange(1,8,1,prec=0.2)
True
>>> 6.1 in floatrange(1,8,1,prec=0.05)
False
>>> list(reversed(floatrange(5)))
[4.0, 3.0, 2.0, 1.0, 0.0]
>>> list(floatrange(10.1,9.7,-0.1))
[10.1, 10.0, 9.9, 9.799999999999999]
```

CHAPTER 2

floatrange class

class `floatrange.`**`floatrange`**`(`[*start*], *stop*[, *step*[, *prec*]])` → floatrange object

Build a virtual sequence of floating point numbers from `start` to `stop` by `step`. Set float equality precision with `prec` (default 0 for strict equality).

Unlike Python range(), generated values are floating point. Like Python range(), the `stop` value is *not* included (to have the stop value included, just add step*0.1 to it).

A *`floatrange`* object is usable like Python `range`, to iterate on the values, to test for its length, to get the value at an index, to get length of values sequence, to search for the index of a value, to count the number of occurrences of a value.

As floating point computation may lead to slightly different values, you should take care:

- Test for presence of a value in the range use `prec`, which is by default zero for *exact* equality.
- When reverting the *`floatrange`*, the new object is created to *generate* reverse values, this can build values with computational differences. To have exact same values in reverse order, cast to a `list` and reverse it.

CHAPTER 3

Other solutions

If you have the scipy's numpy package installed, see `numpy.arange()` and `numpy.linspace()` functions (docs for arange and linspace). If you have heavy computation, this is the way to go.

Note these differences between numpy functions and floatrange:

- floatrange generate values when needed, like `range()`.
- floatrange has a precision parameter `prec` to deal with inaccurate floating point computation.
- floatrange is a pure Python module.

And searching for `Python float range`, or `frange` will give you other solutions (see also comments on these solutions when available).

f

F