pyinter Documentation

Release 0.1

Inti Ocean

December 13, 2013

Contents

Pyinter is a python interval library which deals with interval arithmetic and sets of intervals (discontinous ranges).

CHAPTER 1

Interval Class

class pyinter.**Interval**

Instances of `Interval` provide the following operations:

`Standard comparison operators: <=, <, ==, !=, >, >=`

Note: comparison is performed solely on the lower value of the `Interval`.

`x in i`

Tests x to see if it is in the range specified by the `Interval` i.

`interval | other`

Performs interval intersection just as `intersect()`

`Interval.`**`intersect`**(*other*)

Returns a new `Interval` representing the intersection of this `Interval` with the other `Interval`

`interval & other`

Performs interval union just as `union()`

`Interval.`**`union`**(*other*)

Returns a new Interval or an `IntervalSet` representing the union of this `Interval` with the other `Interval`.

If the two intervals are overlaping then this will return an `Interval`, otherwise this returns an `IntervalSet`.

1.1 *Interval* Construction helpers

`pyinter.interval.`**`closed`**(*lower_value*, *upper_value*)

Helper function to construct an interval object with closed lower and upper.

For example:

```
>>> closed(100.2, 800.9)
[100.2, 800.9]
```

`pyinter.interval.`**`closedopen`**(*lower_value*, *upper_value*)

Helper function to construct an interval object with a closed lower and open upper.

For example:

```
>>> closedopen(100.2, 800.9)
[100.2, 800.9)
```

pyinter.interval.**open**(*lower_value*, *upper_value*)

Helper function to construct an interval object with open lower and upper.

For example:

```
>>> open(100.2, 800.9)
(100.2, 800.9)
```

pyinter.interval.**openclosed**(*lower_value*, *upper_value*)

Helper function to construct an interval object with a open lower and closed upper.

For example:

```
>>> openclosed(100.2, 800.9)
(100.2, 800.9]
```

CHAPTER 2

IntervalSet Class

class pyinter.**IntervalSet**(*iterable=None*)

A class to hold collections of intervals, otherwise known as discontinous ranges

add(*other*)

Add an Interval to the IntervalSet by taking the union of the given Interval object with the existing Interval objects in self.

This has no effect if the Interval is already represented. :param other: an Interval to add to this IntervalSet.

intersection(*other*)

Returns a new IntervalSet which represents the intersection of each of the intervals in this IntervalSet with each of the intervals in the other IntervalSet. :param other: An IntervalSet to intersect with this one.

union(*other*)

Returns a new IntervalSet which represents the union of each of the intervals in this IntervalSet with each of the intervals in the other IntervalSet :param other: An IntervalSet to union with this one.

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

Python Module Index

p