srange Documentation

Release 0.0.4

Jon Tischler and Christian Schlepuetz

September 15, 2014

Contents

Contents:

CHAPTER 1

srange Package

Contents:

1.1 srange Module

class srange.srange.**srange**(*r=''*, *auto_reset=True*)

String-range class.

This class provides functions to convert a string representing integers and ranges of integers to an object which can be iterated over all the values contained in the string and a list of individual values or subranges can be retrieved.

EXAMPLE::

```
>>> sr = srange("1,3,4-5,8-11,12-13,20")
>>> for val in sr:
                        print("%d," % val),
1, 3, 4, 5, 8, 9, 10, 11, 12, 13, 20,
```

NOTE: String ranges can only contain integer numbers and must be strictly monotonically increasing. Multiple identical values are not allowed. addition of __resort_list() now allows for mis-ordered simple ranges, but they still must not overlap.

TODO: The issues of the above note should be addressed to make the code more robust and forgiving. There is no reason one should not specify sub- ranges in arbitrary orders.

The range is checked to be monotonic, it returns None if no more values last is the last number obtained from this range, use -Inf to get start of range, it returns the next

variables and methods that you may be interested in

variables of interest	description
self.r	the input string, after formatting and compacting
self.previous_item	the previous value produced, initially set very low
self.auto_reset	if True (default), then previous_item is reset to min at each call to __iter__

methods of interest	action
next()	returns next value, updates previous_item too
reset_previous()	reset the iterator so it starts with the first value
after(prev)	returns value that follows prev, without changing the current point in iteration
first()	returns the first number in the range, for self.r="3,5,9-20", self.first() returns 3
last()	returns the last number in the range, for self.r="3,5,9-20", self.last() returns 20
len()	returns number of points in the range, for self.r="3,5,9-20", self.len() returns 14
is_in_range(m)	returns True if m is in self.r, otherwise False
index(ipnt)	return the ipntth number from range, first number is ipnt==0, returns None if ipnt negative or too big
val2index(m)	returns index into r that corresponds to m. e.g. for r='3,5,9-20', m=5 returns 1.
sub_range(start,n,...)	returns a new range that is a sub range of current one, setLast=False
list(self)	returns a list where each element is a value in the range, CAUTION this can make a VERY big list

special methods	command	result using: sr = srange('1-4')
__getitem__(n)	print sr[2]	3
__len__()	print len(sr)	4
__str__()	print str(sr)	1-4
__repr__()	print repr(sr)	srange('1-4', len=4, previous=0, auto_reset=True)

after(*val*)

Return the value or the element that follows after the given value.

EXAMPLE::

```
>>> sr = srange("3,5,9-20")
>>> print sr.after(5)
9
```

first()

Return the number of the first item in the range. This method uses but does not change any internal variables, e.g. no self.xxxx

EXAMPLE::

```
>>> sr = srange("3,5,9-20")
>>> print sr.first()
3
```

index(*n*)

Return the n-th element from the string range. This method uses but does not change any internal variables, e.g. no self.xxxx

is_in_range(*item*)

Return True if item is in string range self.r, False otherwise. This method uses but does not change any internal variables, e.g. no self.xxxx

last()

Return the value of the last item in the range. This method uses but does not change any internal variables, e.g. no self.xxxx

EXAMPLE::

```
>>> sr = srange("3,5,9-20")
>>> print sr.last()
20
```

len()
Return the number of items in the string range. This method uses but does not change any internal variables, e.g. no self.xxxx

EXAMPLE::

```
>>> sr = srange("3,5,9-20")
>>> print sr.len()
14
```

list()
Expand a string range into a standard python list. This method uses but does not change any internal variables, e.g. no self.xxxx

EXAMPLE::

```
>>> print srange("3,5,9-13").list()
[3, 5, 9, 10, 11, 12, 13]
```

CAUTION: The following statement:

```
>>> list("1-100000",";")
```

will produce a list with 100000 elements!

Max list length for a 32 bit system is (2^32 - 1)/2/4 = 536870912 on my computer I get a MemoryError for lengths > 1e8, so limit to 1e7

next()
Return the next value in the string range. Also update self.previous_item.

reset_previous()
Reset previous_item to the lowest possible integer value.

sub_range(*start*, *n*, *set_last=False*)
Return a sub range from the original range as a new range string.

The new range starts at with the value start and has up to n elements. If start is not an element in the range, then it begin with first element after start. If set_last is True, then self.previous_item is set to the new end, otherwise no change is made. This method only changes an internal variable "self.previous_item" when set_last is True.

EXAMPLE::

```
>>> sr = srange('3,5,9-20')
>>> print sr.sub_range(start = 5, n = 3)
5,9-10
```

val2index(*val*)
Return the index into the srange that corresponds to val. This method uses but does not change any internal variables, e.g. no self.xxxx

EXAMPLE::

```
>>> r = '3, 5, 9-20'
>>> print val2index(5)
1
```

1.2 symrange Module

class srange.symrange.**symrange**(*endVal*, *negativeFirst=False*, *auto_reset=True*)

symrange class.

This class provides functions to loop symmetrically outward from 0 i.e., 0,1,-1,2,-2,3,-3,... endVal can only be a positive integer or 0

EXAMPLE::

```
>>> for i in symrange(2): print i
prints:
0 1 -1 2 -2 3 -3

>>> for i in symrange(2,True): print i
prints:
0 -1 1 -2 2 -3 3
```

NOTE: symrange can only contain integers.

variables and methods that you may be interested in:

variables	
self.endVal	the highest value +/- returned, this is always >= 0
self.negativeFirst	if True then 0,-1,+1,-2,+2,... Otherwise 0,+1,-1,+2,-2,...
self.auto_reset	if True (default), then previous is reset to None at each call to __iter__
self.length	total number of items in range, you can also get this from len(symrange(n)) or symrange(n).len()
self.previous	last value returned by the iterator, when previous==None, then a call to next() returns 0

methods	
next()	returns next value, updates previous too
last()	returns the last number in the range, for self.r="3,5,9-20", self.last() returns 20
len()	returns number of points in the range, for self.r="3,5,9-20", self.len() returns 14
first()	returns first value of iteration, always returns 0
after(prev)	returns value that follows prev, without changing the current point in iteration
index(ipnt)	return the ipntth number from range, first number is ipnt==0, returns None if ipnt negative or too big, same as symrange(2)[ipnt]
val2index(m)	returns index into range that corresponds to m. e.g. for r='0,-1,1,-2,2', m=1 returns 2.
list(self)	returns a list where each element is a value in the range, CAUTION this can make a VERY big list if n is large

special methods	command	result using: syr = symrange(5)
__getitem__(n)	print syr[3]	2
__len__()	print len(syr)	11
__str__()	print str(syr)	symrange starting from 0, going to 5, doing positives first, current value = "initialized to start"
__repr__()	print repr(syr)	symrange[endVal=5, negativeFirst=False, previous=None, len=11, auto_reset=True]

after(*val*)

Return the value of the element that follows after val (which is given).

first()

Return the value of the first item in the range. This is just for completeness, it always returns 0

index(*n*)
Returns the n-th element from the range, zero based, n==0 is first element. This method uses but does not change any internal variables, e.g. no self.xxxx This functionality also available by symrange(2)[n], which calles __getitem__() below

last()
Return the value of the last item in the range. This method uses but does not change any internal variables, e.g. no self.xxxx

len()
Return the number of items in the symrange. Usage: as symrange(3).len()

list()
Expands the symrange into a standard python list. This method uses but does not change any internal variables, e.g. no self.xxxx

EXAMPLE::

```
>>> print symrange(2).list()
[0, 1, -1, 2, -2]
```

CAUTION: The following statement:

```
>>> symrange(100000).list()
```

will produce a list with 200001 elements!

Max list length for a 32 bit system is (2^32 - 1)/2/4 = 536870912 on my computer I get a MemoryError for lengths > 1e8, so limit to 1e7

next()
Return the next value in the symrange.

val2index(*val*)
Return the index into the symrange that produces val. This method uses but does not change any internal variables, e.g. no self.xxxx

EXAMPLE::

```
>>> sr = symrange(4)
>>> print sr.val2index(3)
5
```

CHAPTER 2

License

CHAPTER 3

Indices and tables

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

S