

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

Ask Question



There is also the step value, which can be used with any of the above:

```
a[start:stop:step] # start through not past stop, by step
```

The key point to remember is that the <code>:stop</code> value represents the first value that is *not* in the selected slice. So, the difference between <code>stop</code> and <code>start</code> is the number of elements selected (if <code>step</code> is 1, the default).

The other feature is that start or stop may be a *negative* number, which means it counts from the end of the array instead of the beginning. So:

```
a[-1] # last item in the array
a[-2:] # last two items in the array
a[:-2] # everything except the last two items
```

Similarly, step may be a negative number:

```
a[::-1] # all items in the array, reversed
a[1::-1] # the first two items, reversed
a[:-3:-1] # the last two items, reversed
a[-3::-1] # everything except the last two items, revers
```

Python is kind to the programmer if there are fewer items than you ask for. For example, if you ask for a[:-2] and a only contains one element, you get an empty list instead of an error. Sometimes you would prefer the error, so you have to be aware that this may happen.

Relation to slice() object

The slicing operator [] is actually being used in the above code with a slice() object using the : notation (which is only valid within []), i.e.:

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

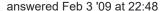
is equivalent to:

```
a[slice(start, stop, step)]
```

Slice objects also behave slightly differently depending on the number of arguments, similarly to <code>range()</code>, i.e. both <code>slice(stop)</code> and <code>slice(start, stop[, step])</code> are supported. To skip specifying a given argument, one might use <code>None</code>, so that e.g. <code>a[start:]</code> is equivalent to <code>a[slice(start, None)]</code> or <code>a[::-1]</code> is equivalent to <code>a[slice(None, None, -1)]</code>.

While the : -based notation is very helpful for simple slicing, the explicit use of <code>slice()</code> objects simplifies the programmatic generation of slicing.

edited Feb 24 at 19:26





Greg Hewgill 679k 147 1019
1172

- 74 Slicing builtin types returns a copy but that's not universal. Notably, <u>slicing NumPy arrays</u> returns a view that shares memory with the original. – Beni Cherniavsky-Paskin Sep 23 '13 at 0:13
- @RodriKing It means that your start and end are empty, and your step is -2. So you reverse (beause it's negative) and take by 2 elements, for the whole list because start and empty are not defined. – mbh86 Aug 9 '18 at 9:54
- 2 Another example: a = list(range(100)) # [0,1,2,, 99] and a[20::-3] It means that you will go reverse 3 by 3. You start from position 20 and go 3 backwards ... 20, 17,

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
of the step has priority to the list ends. I was expecting to get [99, 96, 93, ..., 24, 21] . – Ciprian Tomoiagă Aug 31 '18 at 10:39
```

@nodakai: Strings are immutable; lists are not. b[:] must be a different object from b, but a[:] and a can be the same. – Greg Hewgill Sep 13 '18 at 3:46 /



The <u>Python tutorial</u> talks about it (scroll down a bit until you get to the part about slicing).

461

The ASCII art diagram is helpful too for remembering how slices work:



One way to remember how slices work is to think of the indices as pointing *between* characters, with the left edge of the first character numbered 0. Then the right edge of the last character of a string of n characters has index n.



By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.