15 Extended Slices

Ever since Python 1.4, the slicing syntax has supported an optional third ``step'' or ``stride'' argument. For example, these are all legal Python syntax: `L[1:10:2]`, `L[:-1:1]`, `L[::-1]`. This was added to Python at the request of the developers of Numerical Python, which uses the third argument extensively. However, Python's built-in list, tuple, and string sequence types have never supported this feature, raising a `TypeError` if you tried it. Michael Hudson contributed a patch to fix this shortcoming.

For example, you can now easily extract the elements of a list that have even indexes:

```
>>> L = range(10)
>>> L[::2]
[0, 2, 4, 6, 8]
```

Negative values also work to make a copy of the same list in reverse order:

```
>>> L[::-1]
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

This also works for tuples, arrays, and strings:

```
>>> s='abcd'
>>> s[::2]
'ac'
>>> s[::-1]
'dcba'
```

If you have a mutable sequence such as a list or an array you can assign to or delete an extended slice, but there are some differences between assignment to extended and regular slices. Assignment to a regular slice can be used to change the length of the sequence:

```
>>> a = range(3)
>>> a
[0, 1, 2]
>>> a[1:3] = [4, 5, 6]
>>> a
[0, 4, 5, 6]
```

Extended slices aren't this flexible. When assigning to an extended slice, the list on the right hand side of the statement must contain the same number of items as the slice it is replacing:

```
>>> a = range(4)
>>> a
[0, 1, 2, 3]
>>> a[::2]
[0, 2]
>>> a[::2] = [0, -1]
>>> a
[0, 1, -1, 3]
>>> a[::2] = [0,1,2]
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
ValueError: attempt to assign sequence of size 3 to extended slice of size 2
```

Deletion is more straightforward:

```
>>> a = range(4)
>>> a
[0, 1, 2, 3]
>>> a[::2]
[0, 2]
>>> del a[::2]
>>> a
[1, 3]
```

One can also now pass slice objects to the `__getitem__` methods of the built-in sequences:

```
>>> range(10).__getitem__(slice(0, 5, 2))
[0, 2, 4]
```

Or use slice objects directly in subscripts:

```
>>> range(10)[slice(0, 5, 2)]
[0, 2, 4]
```

To simplify implementing sequences that support extended slicing, slice objects now have a method `indices(`*length*`)` which, given the length of a sequence, returns a `(`*start*`,` *stop*`,` *step*`)` tuple that can be passed directly to `range()`. `indices()` handles omitted and out-of-bounds indices in a manner consistent with regular slices (and this innocuous phrase hides a welter of confusing details!). The method is intended to be used like this:

```
class FakeSeq:
    ...
    def calc_item(self, i):
        ...
    def __getitem__(self, item):
        if isinstance(item, slice):
            indices = item.indices(len(self))
            return FakeSeq([self.calc_item(i) for i in range(*indices)])
        else:
            return self.calc_item(i)
```

From this example you can also see that the built-in `slice` object is now the type object for the slice type, and is no longer a function. This is consistent with Python 2.2, where `int`, `str`, etc., underwent the same change.

Release 1.01.
See [About this document...](#) for information on suggesting changes.