Python Basics

S.R. Doty

August 27, 2008

Contents

```
>>> x=range(0,20,2)
>>> x
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
>>> x[2:5]
[4, 6, 8]
>>> x[0:5]
[0, 2, 4, 6, 8]
```

When taking a slice, either parameter `start` or `end` may be omitted: if `start` is omitted then the slice consists of all items up to, but not including, the one at index position `end`, similarly, if `end` is omitted the slice consists of all items starting with the one at position `start`. For instance, with the list `x` as defined above we have

```
>>> x[:5]
[0, 2, 4, 6, 8]
>>> x[2:]
[4, 6, 8, 10, 12, 14, 16, 18]
```

In this case, `x[:5]` is equivalent to `x[0:5]` and `x[2:]` is equivalent to `x[2:len(x)]`.

There is an optional third parameter in a slice, which if present represents an increment, just as in the range function. For example,

```
>>> list = range(20)
>>> list
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
>>> list[0:16:2]
[0, 2, 4, 6, 8, 10, 12, 14]
>>> list[0:15:2]
[0, 2, 4, 6, 8, 10, 12, 14]
```


Notice that one may cleverly use a *negative* increment to effectively reverse a list, as in:

```
>>> list[18::-1]
[17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

In general, the slice `x[len(x)::-1]` reverses any existing list `x`.

3.8.3 Joining two lists

Two existing lists may be *concatenated* together to make a longer list, using the `+` operator:

```
>>> [2,3,6,10] + [4,0,0,5,0]
[2, 3, 6, 10, 4, 0, 0, 5, 0]
```

3.8.4 List methods

If `x` is the name of an existing list, we can `append` an item `item` to the end of the list using

```
x.append(item)
```

For example,

```
>>> x = [3, 6, 8, 9]
>>> x.append(999)
>>> x
[3, 6, 8, 9, 999]
```