CHAPTER

FIVE

DATA STRUCTURES

This chapter describes some things you've learned about already in more detail, and adds some new things as well.

5.1 More on Lists

The list data type has some more methods. Here are all of the methods of list objects:

`list.`**`append`**`(`*x*`)`
Add an item to the end of the list. Equivalent to `a[len(a):] = [x]`.

`list.`**`extend`**`(`*iterable*`)`
Extend the list by appending all the items from the iterable. Equivalent to `a[len(a):] = iterable`.

`list.`**`insert`**`(`*i*, *x*`)`
Insert an item at a given position. The first argument is the index of the element before which to insert, so `a.insert(0, x)` inserts at the front of the list, and `a.insert(len(a), x)` is equivalent to `a.append(x)`.

`list.`**`remove`**`(`*x*`)`
Remove the first item from the list whose value is equal to *x*. It raises a `ValueError` if there is no such item.

`list.`**`pop`**`(`[*i*]`)`
Remove the item at the given position in the list, and return it. If no index is specified, `a.pop()` removes and returns the last item in the list. (The square brackets around the *i* in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)

`list.`**`clear`**`()`
Remove all items from the list. Equivalent to `del a[:]`.

`list.`**`index`**`(`*x*[, *start*[, *end*]]`)`
Return zero-based index in the list of the first item whose value is equal to *x*. Raises a `ValueError` if there is no such item.

The optional arguments *start* and *end* are interpreted as in the slice notation and are used to limit the search to a particular subsequence of the list. The returned index is computed relative to the beginning of the full sequence rather than the *start* argument.

`list.`**`count`**`(`*x*`)`
Return the number of times *x* appears in the list.

`list.`**`sort`**`(`*, *key*=*None*, *reverse*=*False*`)`
Sort the items of the list in place (the arguments can be used for sort customization, see `sorted()` for their explanation).

`list.`**`reverse`**`()`
Reverse the elements of the list in place.

`list.`**`copy`**`()`
Return a shallow copy of the list. Equivalent to `a[:]`.