Lists

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

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

extend(*L*)
Extend the list by appending all the items in the given list; equivalent to a[len(a):] = *L*.

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*).

remove(*x*)
Remove the first item from the list whose value is *x*. It is an error if there is no such item.

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.

index(*x*)
Return the index in the list of the first item whose value is *x*. It is an error if there is no such item.

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

sort()
Sort the items of the list, in place.

reverse()
Reverse the elements of the list, in place.

An example that uses most of the list methods:

```
>>> a = [66.25, 333, 333, 1, 1234.5]
>>> print a.count(333), a.count(66.25), a.count('x')
2 1 0
>>> a.insert(2, -1)
>>> a.append(333)
>>> a
```

```
[66.25, 333, -1, 333, 1, 1234.5, 333]
>>> a.index(333)
1
>>> a.remove(333)
>>> a
[66.25, -1, 333, 1, 1234.5, 333]
>>> a.reverse()
>>> a
[333, 1234.5, 1, 333, -1, 66.25]
>>> a.sort()
>>> a
[-1, 1, 66.25, 333, 333, 1234.5]
```

Using Lists as Stacks

The list methods make it very easy to use a list as a stack, where the last element added is the first element retrieved (``last-in, first-out''). To add an item to the top of the stack, use append(). To retrieve an item from the top of the stack, use pop() without an explicit index. For example:

```
>>> stack = [3, 4, 5]
>>> stack.append(6)
>>> stack.append(7)
>>> stack
[3, 4, 5, 6, 7]
>>> stack.pop()
7
>>> stack
[3, 4, 5, 6]
>>> stack.pop()
6
>>> stack.pop()
5
>>> stack
[3, 4]
```

Using Lists as Queues

You can also use a list conveniently as a queue, where the first element added is the first element retrieved (``first-in, first-out''). To add an item to the back of the queue, use append(). To retrieve an item from the front of the queue, use pop() with 0 as the index. For example:

```
>>> queue = ["Eric", "John", "Michael"]
>>> queue.append("Terry")       # Terry arrives
>>> queue.append("Graham")      # Graham arrives
>>> queue.pop(0)
'Eric'
>>> queue.pop(0)
'John'
>>> queue
['Michael', 'Terry', 'Graham']
```

The del statement

There is a way to remove an item from a list given its index instead of its value: the del statement. This differs from the pop() method which returns a value. The del statement can also be used to remove slices from a list or clear the entire list (which we did earlier by assignment of an empty list to the slice). For example:

```
>>> a = [-1, 1, 66.25, 333, 333, 1234.5]
>>> del a[0]
>>> a
[1, 66.25, 333, 333, 1234.5]
>>> del a[2:4]
>>> a
[1, 66.25, 1234.5]
>>> del a[:]
>>> a
[]
```

del can also be used to delete entire variables:

```
>>> del a
```

Referencing the name a hereafter is an error (at least until another value is assigned to it). We'll find other uses for del later.

Esempi

```
list = ['larry', 'curly', 'moe']
# print(list)  #--> ['larry', 'curly', 'moe']

## append elem at end
list.append('shemp')
#print(list)   #--> ['larry', 'curly', 'moe', 'shemp']

## insert elem at index 1
list.insert(1, 'xxx')
#print(list)   #--> ['larry', 'xxx', 'curly', 'moe', 'shemp']

## add list of elems at end
list.extend(['yyy', 'zzz'])
#print(list)   #--> ['larry', 'xxx', 'curly', 'moe', 'shemp',
'yyy', 'zzz']

print(list.index('curly'))          #    -->  2

## search and remove that element
list.remove('curly')
#print(list)   #--> ['larry', 'xxx', 'moe', 'shemp', 'yyy', 'zzz']

## removes and returns 'xxx'
list.pop(1)
#print(list)   #--> ['larry', 'moe', 'shemp', 'yyy', 'zzz']
```