

Table 2-4. Methods implemented in `list` or `deque` (those that are also implemented by object are omitted for brevity)

	list	deque	
<code>s.__add__(s2)</code>	●	●	<code>s + s2</code> – concatenation
<code>s.__contains__(e)</code>	●	●	<code>e in s</code>
<code>s.__copy__()</code>		●	Support for <code>copy.copy</code> (shallow copy)
<code>s.__delitem__(p)</code>	●	●	Remove item at position <code>p</code>
<code>s.__getitem__(p)</code>	●	●	<code>s[p]</code> – get item or slice at position
<code>s.__iadd__(s2)</code>	●	●	<code>s += s2</code> – in-place concatenation
<code>s.__imul__(n)</code>	●	●	<code>s *= n</code> – in-place repeated concatenation
<code>s.__iter__()</code>	●	●	Get iterator
<code>s.__len__()</code>	●	●	<code>len(s)</code> – number of items
<code>s.__mul__(n)</code>	●	●	<code>s * n</code> – repeated concatenation
<code>s.__reversed__()</code>	●	●	Get iterator to scan items from last to first
<code>s.__rmul__(n)</code>	●	●	<code>n * s</code> – reversed repeated concatenation ^a
<code>s.__setitem__(p, e)</code>	●	●	<code>s[p] = e</code> – put <code>e</code> in position <code>p</code> , overwriting existing item or slice
<code>s.append(e)</code>	●	●	Append one element to the right (after last)
<code>s.appendleft(e)</code>		●	Append one element to the left (before first)
<code>s.clear()</code>	●	●	Delete all items
<code>s.copy()</code>	●	●	Shallow copy of the list or deque
<code>s.count(e)</code>	●	●	Count occurrences of an element
<code>s.extend(i)</code>	●	●	Append items from iterable <code>i</code> to the right
<code>e.extendleft(i)</code>		●	Append items from iterable <code>i</code> to the left
<code>s.index(e)</code>	●	●	Find position of first occurrence of <code>e</code>
<code>s.insert(p, e)</code>	●	●	Insert element <code>e</code> before the item at position <code>p</code>
<code>s.pop()</code>	●	●	Remove and return last item ^b
<code>s.popleft()</code>		●	Remove and return first item
<code>s.remove(e)</code>	●	●	Remove first occurrence of element <code>e</code> by value
<code>s.reverse()</code>	●	●	Reverse the order of the items in place
<code>s.rotate(n)</code>		●	Move <code>n</code> items from one end to the other
<code>s.sort([key], [reverse])</code>	●		Sort items in place with optional keyword arguments <code>key</code> and <code>reverse</code>

^aReversed operations are explained in Chapter 16.

^b`a_list.pop(p)` allows removing from position `p`, but `deque` does not support that option.