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

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

[a] Reversed operators are explained in Chapter 16.

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