

Table 2-3. Methods and attributes found in `list` or `array` (deprecated array methods and those also implemented by object are omitted for brevity)

	list	array	
<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>
<code>s.__deepcopy__()</code>		●	Optimized support for <code>copy.deepcopy</code>
<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>p</code>
<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
<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 after last
<code>s.buffer_info()</code>		●	Return a tuple (<code>address</code> , <code>length</code>) giving the current memory address and the length in elements of the buffer used to hold array's contents
<code>s.byteswap()</code>		●	Swap bytes of all items in array for endianness conversion
<code>s.clear()</code>	●		Delete all items
<code>s.copy()</code>	●		Shallow copy of the list
<code>s.count(e)</code>	●	●	Count occurrences of an element
<code>s.extend(it)</code>	●	●	Append items from iterable <code>it</code>
<code>s.frombytes(b)</code>		●	Append items from byte sequence interpreted as packed machine values
<code>s.fromfile(f, n)</code>		●	Append <code>n</code> items from binary file <code>f</code> interpreted as packed machine values
<code>s.fromlist(l)</code>		●	Append items from list; if one causes <code>TypeError</code> , none are appended
<code>s.fromunicode(d)</code>		●	Extends this array with data from the given unicode string
<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.itemsize</code>		●	Length in bytes of each array item
<code>s.pop([p])</code>	●	●	Remove and return item at position <code>p</code> (default: last)
<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.sort([key], [reverse])</code>	●		Sort items in place with optional keyword arguments <code>key</code> and <code>reverse</code>
<code>s.tobytes()</code>		●	Return items as packed machine values in a <code>bytes</code> object
<code>s.tofile(f)</code>		●	Save items as packed machine values to binary file <code>f</code>
<code>s.tolist()</code>		●	Return items as numeric objects in a <code>list</code>
<code>s.tounicode()</code>		●	Convert the array to a unicode string
<code>s.typecode</code>		●	One-character string identifying the C type of the items