Objects Reference

Operation	Effect	Return value
`xs.append(value)`	Add one entry to `xs`	`None`
`xs.remove(value)`	Remove the first equal entry from `xs`	`None`; raises `ValueError` if absent
`xs.reverse()`	Reverse `xs` in place	`None`
`xs.sort()`	Sort `xs` in place	`None`
`list(reversed(xs))`	Leave `xs` unchanged	A new list in reverse order
`sorted(xs)`	Leave `xs` unchanged	A new sorted list
`xs[:]` or `copy.copy(xs)`	Copy the outer list	A new list sharing the original entries
`copy.deepcopy(xs)`	Copy the list and nested containers	An independent copy of that data

Classes and instances

Form	Meaning
`class Playlist:`	Define a type
`Playlist(1, 'Study')`	Create an instance and initialize it
`self.title = title`	Store a value on the current instance
`playlist.add_song(song)`	Call a method, supplying the instance as `self`
`__init__(self, ...)`	Initialize a new instance
`__str__(self)`	Return the object's text representation
`__eq__(self, other)`	Return the result of an equality comparison
`a is b`	Test whether two names refer to the same object
`isinstance(value, Playlist)`	Test whether a value is a Playlist instance
`self._songs`	An attribute intended for internal use