Python Cheat Sheet: Set Methods

Method	Description	Example
`set.add(x)`	Add an element to this set	`>>> s = {1, 2, 3}` `>>> s.add(4)  # {1, 2, 3, 4}`
`set.clear()`	Remove all elements from this set	`>>> s = {1, 2, 3}` `>>> s.clear()  # set()`
`set.copy()`	Create and return a flat copy of this set	`>>> s = {1, 2, 'Alice'}` `>>> s.copy()  # Returns: {1, 2, 'Alice'}`
`set.difference(x)`	Return a new set with elements of this set except the ones in the given set arguments.	`>>> {1, 2, 3}.difference({1, 2})` `{3}`

`set.difference_update(iter)`	Remove all elements from this set that are members of any of the given set arguments.	`>>> s = {1, 2, 3}`
		`>>> s.difference_update({1, 2})`
		`# s == {3}`

`set.discard(x)`	Remove an element from this set if it is a member, otherwise do nothing.	`>>> s = {'Alice', 'Bob', 'Cloe'}`
		`>>> s.discard('Bob')`
		`# s == {'Alice', 'Cloe'}`

`set.intersection()`	Return a new set of elements that are members of this and the set argument(s).	`>>> {1, 2, 3, 4}.intersection({3, 4, 5})` `{3, 4}`

`set.intersection_update()`	Removes all elements from this set that are not members in all other specified sets.	`>>> s = {1, 2, 3, 4}`
		`>>> s.intersection_update({3, 4, 5})`
		`# s == {3, 4}`

`set.isdisjoint(x)`	Return True if their intersection is the empty set.	`>>> {1, 2, 3, 4}.isdisjoint({'Alice', 'Bob'})` `True`

`set.issubset()`	Return True if all elements of this set are members of the specified set argument.	`>>> t = {'Alice', 'Bob', 'Carl', 'Liz'}`
		`>>> {'Alice', 'Bob'}.issubset(t)`
		`True`

`set.issuperset()`	Return True if all elements of the specified set argument are members of this set.	`>>> {'Alice', 'Bob', 'Carl'}.issuperset({'Alice'})` `True`

`set.pop()`	Remove and return a random element from this set. KeyError if set is empty.	`>>> s = {'Alice', 'Bob', 'Carl'}`
		`>>>s.pop()`
		`'Alice'`

`set.remove()`	Remove and return a specific element from this set as defined in the argument. If the set doesn't contain element, raise KeyError.	`>>> s = {'Alice', 'Bob', 'Cloe'}`
		`>>> s.remove('Bob')`
		`# s == {'Alice', 'Cloe'}`

`set.symmetric_difference()`	Return new set with elements in either this or the specified set argument, but not both.	`>>> {1, 2, 3}.symmetric_difference({2, 3, 4})` `{1, 4}`

`set.symmetric_difference_update()`	Replace this set with the symmetric difference, i.e., elements in either this set or the specified set argument, but not both.	`>>> s = {1, 2, 3}`
		`>>> s.symmetric_difference_update({2, 3, 4})`
		`>>> s`
		`{1, 4}`

`set.union()`	Create and return new set with all elements in this or any of the specified sets.	`>>> {1, 2, 3, 4}.union({3, 4, 5})` `{1, 2, 3, 4, 5}`
`set.update()`	Update this set with all elements that are in	`>>> s = {1, 2, 3, 4}`

	this or any of the specified set arguments.	`s.update({3, 4, 5})` `# s == {1, 2, 3, 4, 5}`