

Python Cheat Sheet: Set Methods

“A puzzle a day to learn, code, and play” → Visit finxter.com

Method	Description	Example
<code>set.add(x)</code>	Add an element to this set	<pre>>>> s = {1, 2, 3} >>> s.add(4) # {1, 2, 3, 4}</pre>
<code>set.clear()</code>	Remove all elements from this set	<pre>>>> s = {1, 2, 3} >>> s.clear() # set()</pre>
<code>set.copy()</code>	Create and return a flat copy of this set	<pre>>>> s = {1, 2, 'Alice'} >>> s.copy() # Returns: {1, 2, 'Alice'}</pre>
<code>set.difference(x)</code>	Return a new set with elements of this set except the ones in the given set arguments.	<pre>>>> {1, 2, 3}.difference({1, 2}) {3}</pre>
<code>set.difference_update(iter)</code>	Remove all elements from this set that are members of any of the given set arguments.	<pre>>>> s = {1, 2, 3} >>> s.difference_update({1, 2}) # s == {3}</pre>
<code>set.discard(x)</code>	Remove an element from this set if it is a member, otherwise do nothing.	<pre>>>> s = {'Alice', 'Bob', 'Cloe'} >>> s.discard('Bob') # s == {'Alice', 'Cloe'}</pre>
<code>set.intersection()</code>	Return a new set of elements that are members of this and the set argument(s).	<pre>>>> {1, 2, 3, 4}.intersection({3, 4, 5}) {3, 4}</pre>
<code>set.intersection_update()</code>	Removes all elements from this set that are not members in all other specified sets.	<pre>>>> s = {1, 2, 3, 4} >>> s.intersection_update({3, 4, 5}) # s == {3, 4}</pre>
<code>set.isdisjoint(x)</code>	Return True if their intersection is the empty set.	<pre>>>> {1, 2, 3, 4}.isdisjoint({'Alice', 'Bob'}) True</pre>
<code>set.issubset()</code>	Return True if all elements of this set are members of the specified set argument.	<pre>>>> t = {'Alice', 'Bob', 'Carl', 'Liz'} >>> {'Alice', 'Bob'}.issubset(t) True</pre>
<code>set.issuperset()</code>	Return True if all elements of the specified set argument are members of this set.	<pre>>>> {'Alice', 'Bob', 'Carl'}.issuperset({'Alice'}) True</pre>
<code>set.pop()</code>	Remove and return a random element from this set. <code>KeyError</code> if set is empty.	<pre>>>> s = {'Alice', 'Bob', 'Carl'} >>> s.pop() 'Alice'</pre>
<code>set.remove()</code>	Remove and return a specific element from this set as defined in the argument. If the set doesn't contain element, raise <code>KeyError</code> .	<pre>>>> s = {'Alice', 'Bob', 'Cloe'} >>> s.remove('Bob') # s == {'Alice', 'Cloe'}</pre>
<code>set.symmetric_difference()</code>	Return new set with elements in either this or the specified set argument, but not both.	<pre>>>> {1, 2, 3}.symmetric_difference({2, 3, 4}) {1, 4}</pre>
<code>set.symmetric_difference_update()</code>	Replace this set with the symmetric difference, i.e., elements in either this set or the specified set argument, but not both.	<pre>>>> s = {1, 2, 3} >>> s.symmetric_difference_update({2, 3, 4}) >>> s {1, 4}</pre>
<code>set.union()</code>	Create and return new set with all elements in this or any of the specified sets.	<pre>>>> {1, 2, 3, 4}.union({3, 4, 5}) {1, 2, 3, 4, 5}</pre>
<code>set.update()</code>	Update this set with all elements that are in this or any of the specified set arguments.	<pre>>>> s = {1, 2, 3, 4} >>> s.update({3, 4, 5}) # s == {1, 2, 3, 4, 5}</pre>