Setting lists

The values in a regular list or a fixed list tuple can be repeated in its elements, but a list of unique values can be created where duplication is not allowed. A restrictive Python list of unique values is known as a "set" and is created by assigning values as a comma-separated list between curly brackets (braces) like this:

phonetic_set = { 'Alpha' , 'Bravo' , 'Charlie' }

Unlike regular lists or tuples, individual elements in a set cannot be referenced using the set name followed by square brackets containing an index number. Instead, sets have powerful methods that can be dot-suffixed to the set name for manipulation and comparison of values:

Hot tip

A set may not contain items that are not unique to its other elements.

Set Method:	Description:
***set*.add(*x*)**	Adds item *x* to the set
***set*.update(*x,y,z*)**	Adds multiple items to the set
***set*.copy()**	Returns a copy of the set
***set*.pop()**	Removes one random item from the set
***set*.discard(*x*)**	Removes item *x* if found in the set
***set1*.intersection(*set2*)**	Returns items that appear in both sets
***set1*.difference(*set2*)**	Returns items in ***set1*** but not in ***set2***

Don't forget

More set methods can be found in the Python documentation online at **docs.python.org**

The built-in Python **type()** function can be used to reveal the data class type and the built-in **len()** function can be used to return the length of the set. Additionally, the Python built-in membership operator **in** can be used to find values in a set.

Typically, a set is used to store unique values that are a collection of changeable values, which can easily be searched and compared using the powerful set methods. Although you cannot access set element values by index, a set can be converted to a regular list using the Python built-in **list()** function to allow element access.