

Don't forget

Like index numbering with lists, the items in a tuple sequence are numbered from zero.

Beware

There must be the same number of variables as items to unpack a tuple.

Hot tip

More set methods can be found in the Python documentation.

Restricting lists

Tuple

The values in a regular list can be changed as the program proceeds (they are “mutable”) but a list can be created with fixed “immutable” values that cannot be changed by the program. A restrictive immutable Python list is known as a “tuple” and is created by assigning values as a comma-separated list between parentheses in a process known as “tuple packing”:

```
colors-tuple = ( 'Red' , 'Green' , 'Red' , 'Blue' , 'Red' )
```

An individual tuple element can be referenced using the tuple name followed by square brackets containing that element’s index number. Usefully, all values stored inside a tuple can be assigned to individual variables in a process known as “sequence unpacking”:

```
a , b , c , d , e = colors-tuple
```

Set

The values in a regular list can be repeated in its elements, as in the tuple above, 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):

```
phonetic-set = { 'Alpha' , 'Bravo' , 'Charlie' }
```

Individual set elements cannot be referenced using the set name followed by square brackets containing an index number, but instead sets have methods that can be dot-suffixed to the set name for manipulation and comparison:

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