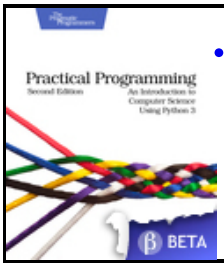
Tuples



- Chapter 11.2 Storing Data Using Tuples

Optional reading

Immutable Sequences

Tuples are immutable sequences: they cannot be modified. Tuples and lists have much in common, but lists are mutable sequences: they can be modified.

Tuples use parentheses instead of square brackets:

```
lst = ['a', 3, -0.2]
tup = ('a', 3, -0.2)
```

Once created, items in lists and tuples are accessed using the same notation:

```
>>> lst[0]
'a'
>>> tup[0]
'a'
```

Slicing can be used with both:

```
>>> lst[:2]
['a', 3]
>>> tup[:2]
('a', 3)
```

Tuples cannot be modified:

```
>>> tup[0] = 'b'
TypeError: 'tuple' object does not support item assignment
```

Tuples have fewer methods than lists. In fact, the only regular methods are `count` and `index`:

```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__doc__', '__eq__', '__format__',
'__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__',
'__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__',
'__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__',
'__subclasshook__', 'append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>> dir(tuple)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__iter__',
'__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__',
'__repr__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'count', 'index']
```

The rest of the list methods are not available for tuple because they modify the object, and tuples, being immutable, cannot be modified.

A `for` can be used to iterate over the values in a tuple:

```
>>> tup = ('a', 3, -0.2)
>>> for item in tup:
    print(item)

a
3
-0.2
```

A tuple can be passed as an argument to the built-in function `len`:

```
>>> len(tup)
3
```

It is also possible to iterate over the indices of a tuple:

```
>>> for i in range(len(tup)):
        print(tup[i])

a
3
-0.2
```

Jennifer Campbell • Paul Gries
University of Toronto
