D: Dictionaries and tuples

Gareth McCaughan Revision 1.8, May 14, 2001

Credits

Introduction

The worksheets that make up most of this course don't cover everything. Two important things they don't cover are "dictionaries" and "tuples". This sheet tells you a little about those.

Dictionaries

The easiest way to explain what a dictionary is is to give an example. So here is one.

```
>>> dict = {}                    An empty dictionary.
>>> dict[1] = 'boo!'             Add a piece of information to it.
>>> dict['ouch'] = 99            And another.
>>> dict                         Lets's see what's in there.
'ouch': 99, 1: 'boo!'            Two ``associations''.
>>> dict['ouch']                 What's associated with 'ouch' ?
99                               This is.
```

So, a dictionary is a little like a list, except that while an array has entries labelled by numbers (`a[0]`, `a[1]`, etc), the entries in a dictionary can be labelled by things other than numbers. (Most usefully, they can be labelled by numbers and by strings.)

It's sometimes helpful to think of a dictionary as being made up of "associations". An association is a piece of information like "`'ouch'` is associated with `99`". So the dictionary we built above contains two associations.

You might use a dictionary to hold an address list, with each association pairing a person's name up with their address. Or it might hold information about all the players in a multi-player game, or about all the computers connected to a network.

The two parts of an association are sometimes called the *key* and the *value*: you find the value by looking up the key. So in the dictionary we made above, the keys are `1` and `'ouch'`, and the corresponding values are `'boo!'` and `99`.

As we've seen, you use dictionaries in the same sort of way as you use lists: if `d` is a dictionary then `d[k]` is the value associated with the key `k`.

To change an association in a dictionary, just say something like `dict['ouch'] = 1000`. To remove an association entirely, say `del dict['ouch']`. (`del` is short for "delete".)

To test whether a dictionary has any association with key `k`, do something like this:

```
>>> dict = {'a': 123, 'b': 987}          Set up a dictionary
>>> dict.has_key('a')                    Any association for 'a' ?
1                                        Yes.
>>> dict.has_key('z')                    Any association for 'z' ?
0                                        Nope.
```

Keys

Not all objects are allowed as keys for dictionaries. We've already seen that numbers and strings are OK. Unfortunately, lists aren't. However, there's a data type very similar to the list that you *can* use: the "tuple". Tuples are the other subject of this sheet. . .

Tuples

A tuple is like a list. There are only two differences you need to care about:

1. Tuples are "immutable". In other words, once you've got a tuple you aren't allowed to change its contents. If `x` is a list then you can say `x[1]=99`; if it's a tuple, you can't.
2. Tuples are written a little differently: instead of `[1,2,3,4]` you say `(1,2,3,4)`.

You can use a tuple almost anywhere where you'd use a list. For instance, you can say `for x in (1,2,3,4):` instead of `for x in [1,2,3,4]` (see Sheet L (*Loops*) if you don't know what that's about).

If for some reason you want a tuple containing only one object, you can't write it as `(x)` (can you think why?). Instead, you say `(x,)`. Weird, I'm afraid . . .