4.3 Making Python objects persistent using the shelve module

Serialization of Python objects using the shelve module

As you remember, the pickle module is used for serializing objects as a single byte stream. Both serializing and deserializing parties must abide by the order of all the elements placed into a file or database, or sent via a network.

There is another handy module, called shelve, that is built on top of pickle, and implements a **serialization dictionary** where objects are pickled and associated with a key. The keys must be ordinary strings, because the underlying database (dbm) requires strings.

Therefore, you can open your shelved data file and access your pickled objects via the keys the way you do when you access Python dictionaries. This could be more convenient for you when you're serializing many objects.

Sound familiar? An analogy to the rack with shelves in the pantry is a correct one. Looks delicious, doesn't it?

Using shelve is quite easy and intuitive.

First, let's import the appropriate module and create an object representing a file-based database:

```
import shelve
my_shelve = shelve.open('first_shelve.shlv', flag='w')
```

The meaning of the optional flag parameter:

Value	Meaning
'r'	Open existing database for reading only
'w'	Open existing database for reading and writing
'c'	Open database for reading and writing, creating it if it doesn't exist (this is a default value)
'n'	Always create a new, empty database, open for reading and writing

Now our shelve object is ready for action, so let's insert a few items and close the shelve object.

```
my_shelve['USD'] = {'code':'US dollar', 'symbol': '$'}
my_shelve['JPY'] = {'code':'Japanese yen', 'symbol': '¥'}
my_shelve.close()
```

Now let's open the shelve file to demonstrate direct access to the elements (contrary to the sequential access to items when we use pickles).

```
new_shelve = shelve.open(shelve_name)
print(new_shelve['USD'])
new_shelve.close()
```

The final code is presented in the right pane.

```
import shelve

shelve_name = 'first_shelve.shlv'

my_shelve = shelve.open(shelve_name, flag='c')
my_shelve['EUR'] = {'code':'Euro', 'symbol': '€'}
my_shelve['GBP'] = {'code':'Pounds sterling', 'symbol': '£'}
my_shelve['USD'] = {'code':'US dollar', 'symbol': '$'}
my_shelve['JPY'] = {'code':'Japanese yen', 'symbol': '¥'}
my_shelve.close()

new_shelve = shelve.open(shelve_name)
print(new_shelve['USD'])
new_shelve.close()
```

You should treat a shelve object as a Python dictionary, with a few additional notes:

- the keys must be strings;
- Python puts the changes in a buffer which is periodically flushed to the disk. To enforce an immediate flush, call the `sync()` method on your shelve object;
- when you call the `close()` method on an shelve object, it also flushes the buffers.

When you treat a shelve object like a Python dictionary, you can make use of the dictionary utilities:

- the `len()` function;
- the `in` operator;
- the `keys()` and `items()` methods;
- the `update` operation, which works the same as when applied to a Python dictionary;
- the `del` instruction, used to delete a key-value pair.

After running the code, you'll notice additionally that some files are created to support the database. Don’t try to alter those files with external utilities, because your shelve may become inconsistent, resulting in read/write errors.

The use of shelve is really easy and effective. Moreover, you should know that you could simulate the shelve by pickling the whole dictionary, but the shelve module uses the memory more efficiently, so whenever you need access to pickled objects, employ a shelve.

And the final remark is:

- because the shelve module is backed by pickle, it isn’t safe to load a shelve from an untrusted source. As with pickle, loading a shelve can execute arbitrary code.