Unit 12 Exercises – Files and exceptions

Aims and objectives

- Learning how to use files for long-term storage of data
- Learning how to implement a complete interactive application
- Learning about exceptions and error-handling

This week's exercises

These exercises involve *files* and *exceptions*. These concepts are explained in the latest lectures, and in Chapters 10 and 11 of the book. This week's tasks extend the birthday book exercise (Unit 11, Task 3); if you have not completed that exercise yet, you will have to do it now.

Task 1 – Re-read Chapter 10

Read Chapter 10. Work through the examples with the Python interpreter, and try some similar examples of your own.

Task 2 – Reading birthdays from a file

The aim of this task is to define a function *getBirthdays* which takes a file name as a parameter and reads birthdays from the file, storing them in a dictionary in the same way as in Unit 11. The dictionary should also be a parameter of the function. The first line of the function definition should therefore be

```
def getBirthdays(fileName,book):
```

The file should contain a number of lines with one birthday per line, in the following format:

```
John,Mar,23
Susan,Feb,16
```

and so on. The file *birthdays.txt* in the Unit12 folder contains some data that you can use for testing; you can also create your own files using the normal Python editor.

For this task, don't worry about handling errors: assume that the file exists, that it has the correct format, that every line gives a valid date, etc. The following points will be useful:

- remember to open the file and then call methods to read data
- the easiest way to read data from this file is to use the *readline* method, but note that it gives you a string *with a newline character at the end*, so you will need to discard the last character
- remember the *split* function from the *string* module: the call
  ```
     split(line,",")
  ```
  will be useful
- the function *int* converts a string into an integer

Test your function (how should you do this?).

Do the work for this task in the file *readfile.py*.

Task 3 – A complete birthday book application

Including your work from Unit 11, you now have functions to perform many useful operations in relation to the birthday book. The task now is to combine these functions into a complete application.

Write a program which repeatedly asks the user to enter a command, asks for further details if necessary, and carries out the corresponding operation. For example, one command could be "read"; in this case the program should ask the user to enter a filename, and then read birthdays from that file into the birthday book. There should be a textual menu with a command for each of the operations from Unit 11, as well as a command "quit" which terminates the program. The lecture on Friday will go over an example of this kind of application. Later in the course we will see how to build a graphical user interface instead of using keyboard input.

Also add a command allowing a new birthday to be entered.

Do the work for this task in the file *birthdaybook.py*.

Task 4 – Handling errors

This exercise is to make the birthday book program robust by detecting as many errors as possible and giving informative error messages to the user. There are many possibilities for errors in the input to the program. For example: the file of birthdays might not exist or might not have the correct format, or dates could be invalid (e.g. Feb 31); when finding a person's birthday, the person might not be in the birthday book; when asking for birthdays in a given month, the name of the month might be incorrect; the user might enter an incorrect command in response to the top-level prompt; and so on.

Start by copying the file *birthdaybook.py* to the file *robust.py*. Then modify the program so that as many errors as you can think of are detected. In some cases, for example trying to open a non-existent file, you should handle the exception raised by the built-in Python function. In other cases, you might like to raise and handle your own exceptions, or you might prefer to use other techniques (for example, checking that the top-level command is correct can be done easily with a series of *if* statements).

Test your program thoroughly, remembering that you are now checking that error cases are handled correctly.

Task 5 – Pickling (Optional)

In Task 2 you read birthdays from a text file. This task explores the use of Python's *pickling* facility, which allows data to be stored in a file without converting to and from strings.

Start by reading section 11.4 of the relevant PDF under Unit 12. Then add commands to your program allowing the birthday book to be written to and read from a file using the functions of the *pickle* module. Only way to create a pickled file is by using *pickle.dump*, so you will have to implement writing before reading and test them together.