Unit 14 Exercises – Exceptions

Aims and objectives

- Learning about exceptions and error-handling
- Learning how to implement a complete interactive application

This week's exercises

These exercises involve *files* and *exceptions*. The former was covered last term, and the latter was explained in the latest lecture. The concepts are also covered in Chapters 13 and 19 of the book. This week's tasks build a complete application called "Birthday book" that manages birthday records using files as storage.

Task 0 – Predicting the outcome of python code snippets (optional)

Go to http://142.93.32.146/index.php/971581?newtest=Y&lang=en and work through the exercises there.

- these exercises are not part of the course.
- they are entirely optional, yet relevant if you want to improve your programming skills.
- the results are anonymous.
- a group of academics (Fionnuala Johnson, Stephen McQuistin, John O'Donnell and John Williamson) hope to publish anonymized findings based on the collected results.

Task 1 – Re-read Chapters 13 and 19

Read Chapters 13 and 19. Work through the examples with the Python interpreter, and try some similar examples of your own.

Task 2 – Birthday book

The idea of this exercise is to store people's birthdays and produce reminders of birthdays that are coming up. A birthday consists of a month and a date, which can be represented by a dictionary such as

```
{ "month":"Sep", "day":17 }
```

The birthday book is a dictionary in which the keys are people's names, and the values are birthdays, represented as dictionaries as above. The functions you have to define are described below; they should all take a birthday book as a parameter, as well as any other parameter specified below. Before you start coding, make sure you have a clear idea of how to produce the desired information.

Step 1: Working in a file called *birthday.py*, write some code to set up a birthday book with several people and their birthdays, for testing purposes.

Step 2: Define a function which, given a person's name, prints his or her birthday.

Step 3: Define a function which, given a month, prints a list of all the people who have birthdays in that month, with the dates.

Step 4: Define a function which, given a month and a date, prints a list of all the people who have birthdays within the next week, with the dates. Don't forget that some of these birthdays might be in the next month, and if the given date is in December, some of them might be in January.

Task 3 – 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 which 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 Unit14/lab 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
  `line.split(",")`
  will be useful
- the function *int* converts a string into an integer

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

Task 4 – Menu

The task now is to combine the functions you have implemented so far 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 Task 2 and 3, as well as a command "quit" which terminates the program. Later in the course we will see how to build a graphical user interface instead of using keyboard input.

Also add a command (and a function) allowing a new birthday to be entered.

Task 5 – Handling errors

This exercise is to make the birthday book program robust by detecting and/or handling 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; the dates can 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.

Modify the birthday book 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 6 – Pickling (Optional)

In Task 4 you read birthdays from a text file. This task explores the use of Python's *pickling* facility, which allows a Python object to be stored in a file while maintaining its structure in a serialized form (without converting to and from human readable strings).

Start by reading how *pickle* works on this link. 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.