CSSE1001

Semester 2, 2012
Assignment 1
10 marks

Due Thursday 30 August, 2012, 5pm

Managing student marks

1 Introduction

For this assignment you will write a program for managing student marks for a course stored in comma separated values (CSV) files. The program you will write is a simple text-based interactive program that allows the user to manages these CSV files.

For this assignment you can assume all relevant files are readable and writeable and have the correct format, and arguments to the functions you need to write are "sensible". Later in the course we will look at error handling.

You can assume that, when the course starts, a CSV file has been created. Each line of this file contains the student number of an enrolled student and entries for marks for each assessable item in the course. The file `marks.csv` is an example of such a file for a course that has five assessable items. The contents of the file is given below.

```
s111111,,,,,
s222222,,,,,
s333333,,,,,
s444444,,,,,
s555555,,,,,
s666666,,,,,
s777777,,,,,
```

There are two activities that need to be performed. One is to merge the results of a given piece of assessment and the other is to update the marks for one or more students.

When a piece of assessment is complete, another tool creates a CSV file for

that piece of assessment. Each row contains a student number and a result. An example file `assign1_marks.csv` file is given below.

```
s555555,7
s333333,10
s666666,9
s111111,10
s999999,9
```

Note that the student numbers are not necessarily in the same order as in the original marks file and that there might be student numbers that don't appear in, or are missing from, the original marks file.

One of the jobs of your program is to merge that data from a file like `assign1_marks.csv` with the data in a file like `marks.csv` to produce a file whose contents is as follows.

```
s111111,10,,,,
s222222,,,,,
s333333,10,,,,
s444444,,,,,
s555555,7,,,,
s666666,9,,,,
s777777,,,,,
```

Below are four examples of using the program. The first merges a results file, the second one attempts to merge the same results file, the third updates some student results and the forth one shows what happens when an illegal command is entered.

```
>>> interact()


Welcome to Results Manager

Commands:
M - merge results files.
```

```
U - update result.


Enter Command: M
Enter results file: marks.csv
Enter new results file: assign1_marks.csv
Enter column: 1
s999999 cannot be merged - no match found.
Enter updated results file (empty to ignore merge): marks_a1.csv
Done.

>>> interact()


Welcome to Results Manager

Commands:
M - merge results files.
U - update results.


Enter Command: M
Enter results file: marks_a1.csv
Enter new results file: assign1_marks.csv
Enter column: 1
s555555 already has a value in column 1 - update ignored.
s333333 already has a value in column 1 - update ignored.
s666666 already has a value in column 1 - update ignored.
s111111 already has a value in column 1 - update ignored.
s999999 cannot be merged - no match found.
Enter updated results file (empty to ignore merge):
Merge ignored.
>>> interact()


Welcome to Results Manager

Commands:
```

```
M - merge results files.
U - update result.


Enter Command: U
Enter results file: marks_a1.csv
Student number (empty to finish): s999999
Enter column: 1
New result: 4
s999999 cannot be merged - no match found.
Student number (empty to finish): s555555
Enter column: 1
New result: 8
Student number (empty to finish): s111111
Enter column: 2
New result: 9
Student number (empty to finish):
Enter updated results file (empty to ignore updates): marks2.csv
Done.

>>> interact()


Welcome to Results Manager

Commands:
M - merge results files.
U - update results.


Enter Command: Q
Unknown command

>>>
```

After these interactions the contents of the file `mark2.csv` is

```
s111111,10,9,,,
```

```
s222222,,,,,
s333333,10,,,,
s444444,,,,,
s555555,8,,,,
s666666,9,,,,
s777777,,,,,
```

Note that in both a merge and an update, when an unknown student result is entered, a message appears and that mark is ignored. Also note that, when merging, an updated result for a known student is ignored and a warning is printed. In both cases, entering an empty output results file will cause the program to terminate without writing the results to a file.

2 Assignment Tasks

For each function that you write you need to provide a suitable comment giving a description, the type and any preconditions. You should use the triple-quote commenting style.

2.1 Download files

The first task is to download the following files:
`assign1.py`, `test_assign1.py`, `marks.csv`, `marks1.csv` and `assign1_marks.csv`.

We suggest you create a folder in which to write your solution and put these files in that folder.

The file `assign1.py` is for your assignment. Add your name and student number in the space provided. **Do not otherwise modify this comment block or the block of code at the end. Add your code after the initial comment block**. When you have completed your assignment you will submit the file `assign1.py` containing your solution to the assignment (see Section 4 for submission details).

The file `test_assign1.py` contains some simple tests you can use on your code. Please read the comments at the beginning of the file.

2.2 Write the code

Finally, write your solution to the assignment making sure you have included suitable comments. There are several functions you need to write and these are described below. **Do not use global variables in your code.**

2.2.1 Get Marks From a File

`get_marks_from_file(filename)` takes the name of a CSV file containing marks as described in the introduction and returns a list of lists representing the marks for students. The marks information should be in the same order as in the file. For example (the actual value displayed in IDLE will be on one line rather than over several lines as below):

```
>>> get_marks_from_file('marks2.csv')
[['s111111', '10', '9', '', '', ''],
 ['s222222', '', '', '', '', ''],
 ['s333333', '10', '', '', '', ''],
 ['s444444', '', '', '', '', ''],
 ['s555555', '8', '', '', '', ''],
 ['s666666', '9', '', '', '', ''],
 ['s777777', '', '', '', '', '']]
>>> get_marks_from_file('assign1_marks.csv')
[['s555555', '7'],
 ['s333333', '10'],
 ['s666666', '9'],
 ['s111111', '10'],
 ['s999999', '9']]
>>>
```

NOTE: In the example above `get_marks_from_file` returns the list of lists and this result is printed by the interpreter.

When reading information from files please use `open(filename, 'rU')` in order to get operating system independent processing of newlines.

2.2.2 Updating a Mark for a Student

`update_mark(all_marks, stud_num, mark, column, check_result_exists)` takes a list of lists of student marks (as returned by `get_marks_from_file`), a student number, a mark for that student, the column to update and a boolean value to determine if updating an existing mark is allowed.

For example :

```
>>> marks = get_marks_from_file('marks2.csv')
>>> update_mark(marks, 's111111', '10', 2, True)
s111111 already has a value in column 2 - update ignored.
>>> update_mark(marks, 's111111', '10', 2, False)
>>> marks
[['s111111', '10', '10', '', '', ''],
 ['s222222', '', '', '', '', ''],
 ['s333333', '10', '', '', '', ''],
 ['s444444', '', '', '', '', ''],
 ['s555555', '8', '', '', '', ''],
 ['s666666', '9', '', '', '', ''],
 ['s777777', '', '', '', '', '']]
>>>
```

2.2.3 Merge Marks

`merge_marks(all_marks, new_marks, column)` takes, as arguments, a list of lists representing the current marks, a list of lists representing the marks to be merged and the column of `all_marks` that is to be updated. The list of lists `all_marks` is updated.

For example:

```
>>> marks = get_marks_from_file('marks.csv')
>>> new_marks = get_marks_from_file('assign1_marks.csv')
>>> merge_marks(marks, new_marks, 1)
s999999 cannot be merged - no match found.
>>> marks
[['s111111', '10', '', '', '', ''],
 ['s222222', '', '', '', '', ''],
```

```
 ['s333333', '10', '', '', '', ''],
 ['s444444', '', '', '', '', ''],
 ['s555555', '7', '', '', '', ''],
 ['s666666', '9', '', '', '', ''],
 ['s777777', '', '', '', '', '']]
>>>
```

2.2.4 Save Marks to File

`save_marks_to_file(records, filename)` takes, as arguments, a list of lists of marks and a file name in which to save the marks. No whitespaces should appear in the file except the newline after each students information.

For example:

```
>>> marks = get_marks_from_file('marks.csv')
>>> save_marks_to_file(marks, 'marks1.csv')
```

The file `marks1.csv` should have the same content as `marks.csv`.

2.2.5 The Top-Level Interface

`interact()` is the top-level function that defines the text-base user interface as described in the introduction.

2.2.6 Hints

For file and string processing: `open, strip, join`
For user interaction: `raw_input`

3 Assessment and Marking Criteria

In addition to providing a working solution to the assignment problem, the assessment will involve discussing your code submission with a tutor. This discussion will take place in the practical session you have signed up to in

week 7. You **must** attend that session in order to obtain marks for the assignment.

In preparation for your discussion with a tutor you may wish to consider:

- any parts of the assignment that you found particularly difficult, and how you overcame them to arrive at a solution;
- whether you considered any alternative ways of implementing a given function;
- where you have known errors in your code, their cause and possible solutions (if known).

It is also important that you can explain to the tutor how each of the functions that you have written operates (for example, if you have used a for loop or a while loop in a function, why this was the right choice).

Marks will be awarded based on a combination of the correctness of your code and on your understanding of the code that you have written. A technically correct solution will not elicit a pass mark unless you can demonstrate that you understand its operation.

We will mark your assignment according to the following criteria.

Criteria	**Mark**
Your code is mostly complete, correct, clear, succinct and well commented. You are able to explain your code.	8 - 10
Your code has some problems OR you have some problems explaining your code.	4 - 7
Your code is clearly incomplete, incorrect, too complex or hard to understand OR you have major problems explaining your code.	1 - 3
Your work has little or no academic merit.	0

A partial solution will be marked. If your partial solution causes problems in the Python interpreter please comment out that code and we will mark that.

Please read the section in the course profile about plagiarism.

4 Assignment Submission

You must submit your completed assignment electronically through Blackboard.

Please read
`http://www.library.uq.edu.au/ask-it/blackboard-assessment`
for information on submitting through Blackboard.

You should electronically submit your copy of the file `assign1.py` (use this name - all lower case).

You may submit your assignment multiple times before the deadline - only the last submission will be marked. After each submission please use Blackboard to check that the file you submitted was the one you intended to submit. Make sure the file is called `assign1.py` and not, for example, `assign1.py.py`

Late submission of the assignment will not be accepted. In the event of exceptional personal or medical circumstances that prevent you from handing in the assignment on-time, you should contact the lecturer in charge and be prepared to supply appropriate documentary evidence. You should be prepared to submit whatever work you have completed at the deadline, if required. Requests for extensions should be made as soon as possible, and preferably before the assignment due date.