CSCA20 Assignment 1

Due: July 20, 2014. 5:00pm

Some Notes Before We Begin

As with A0, the most important thing to mention about assignments is that they will **NOT** be pre-marked. That means that it is your responsibility to ensure that your code works properly with all functions named correctly, etc. If your code crashes the automarker, or the automarker cannot find your file/function/etc, there will be **NO** remarks.

Once again, style will count heavily towards your mark. Following the design recipe and providing good quality comments will be important for your grade on this assignment.

Your assignment

The University of Scarborough (Toronto Campus) was so pleased with your work in A0, they have another assignment for you. This time, they want a system to replace their very out-dated SORI[1] enrolment system. At the moment, every time they want to add a new student or look up a mark, they have to edit or read through large spreadsheets. Your job is to automate some of their tasks.

They have provided you with two files for this task, `enrolment.csv` and `grades.csv`. They assure you that the files are properly formatted `csv` files, so you don't have to worry about getting bad data.

The Dictionary

This assignment will revolve around using dictionaries, so this is a good time to make sure we fully understand the format of the dictionary we will be using. The `keys` to the dictionary will be a 9-10 character student id, stored as a string[2], and the `values` are lists, where the first 3 elements are strings representing the student's `last name`, `first name` and `department` (in that order). The rest of the elements in the list are tuples of the form `(course code, mark)`, stored as a string and an int respectively, representing the mark that the student achieved in that particular course. So a dictionary entry might look something like this:

```
student_data = {"1234567890":  ["Harrington", "Brian", "Computer Science", ("CSCA01",99),
("CSCA02",97), ("ENGA01",53)], "9876543210":["Smith","Alice","English",("ENGA01",87),
("PSYD03",85)] }
```

Functions

In order to complete this assignment, you must complete the following functions. You may use helper functions if you wish, but the following functions must be present.

add_student

`add_student` takes a dictionary and four strings as parameters. The dictionary is the student data dictionary as shown above, the strings are, in order, the student's last name, the student's first name, the student's student number, and the student's department. This function should add a new record for this student to the dictionary, overwriting any previous record. It should not return anything [3]

[1] Student Office Record Inventory

[2] They use numbers now, but they might decide to use letters in future, so we'll stick with a string

[3] We haven't gotten into mutability in this course, but dictionaries, sets and lists can be edited by functions, unlike strings and ints. You don't need to return the dictionary, just change it and it will stay changed. Try it for yourself.

get_name

`get_name` takes a dictionary and a string as parameters, the dictionary is the standard student data described above, the string is a student number. The function should return the name of the student whose student number was entered if such a record exists in the dictionary. The format of the name should be `first name last name`, with a single space in-between. If no such record exists, it should return the string "No such student"[4].

add_grade

`add_grade` takes a dictionary, 2 strings and an int as parameters. The dictionary is the standard student data dictionary, the first string is a student number, the second string is a course code and the int is that student's mark in the given course. The function should update the dictionary with this grade entry. If no such student exists, the function should leave the dictionary unchanged. If the student already has a mark for this course, the new mark should over-write the old. The function should not return anything (or, if you prefer, it should return `None`).

get_grade

`get_grade` takes a dictionary, and two strings as parameters. The dictionary is the standard student data dictionary, the first string is a student number and the second string is a course code. The function should return the grade that the student received in the given course code as an integer. If no such student exists, or the student didn't take the specified course, the function should return `-1`.

read_student_data

`read_student_data` takes two files as parameters[5]. The files will be `csv` files formatted exactly as `enrolment.csv` and `grades.csv`. The function should read these files, and return a dictionary representing the information contained in those files in the format specified above.

interview

The interview function takes no parameters, and acts as the interface for the system. When run it should ask the user what they want to do, options such as reading data from csv files, adding a student, checking a mark, etc should be included. This function is the only place in your code where you should use `print`, `input`, `open` or `close`. Your goal is to make this function as user-friendly as possible. You will be marked on your design, and so even if you can't get the other functions to work, you should still write the code to show how this function *would* work, even if it's printing dummy data.

What to hand in

All of your code should be in a file called `a1.py`. The code should not print anything or ask for any input when imported (any global code should be inside an `if(__name__ == "__main__")` block). The only module you may import is `csv`.

[4] Remember that capitalization and punctuation count

[5] As in the exercises, remember that these are file handles, not the names of the files. The files will be opened and closed by the `interview` function, or the automarker, this function should not open or close any files