Programming Assignment 4

Getting started

Complete the reading and practice assignments posted on the course schedule. Review class handouts and examples. This assignment is designed to emphasize working with files and directories, use of dictionaries, tuples, sets and lists.

Programming Project: Classlists **worth: 20 points**

Estimate number of students eligible for a course.

Data and program overview

In this assignment you will be working with data on student enrollment in classes of the Boston School of Magic. The data includes

- A list of courses included in two certificate programs of the school (descriptions of programs provided in *program1.txt* and *program2.txt*)
- Course prerequisites structure (*prereqs.txt*)
- Class lists, including grades, for students who took these courses in the past two semesters (2017-9 and 2018-1) as well as those who are taking the courses now (2018-9). Each class list comes in a separate file, the title of which includes the course number and the semester designation (1 –Spring, 9 -- Fall). For instance, file *c1250-2018-9.txt* stores records of students enrolled in course number 1250 in the Fall of 2018.

There is a lot of useful information that can be derived from this data, such as, how many students are eligible for a diploma in each of the two programs, how many courses per semester students take on average, etc. In this assignment, you will be creating a program that uses this data to estimate the number of students who would be eligible for taking a specific course. An *eligible* student:

- has completed all prerequisites for the course, and
- has not taken it in any prior semester.

Boston School of Magic allows students to enroll in any course, regardless of the program.

I provide two data sets for this assignment, in zip files called *data-small.zip* and *data-large.zip*. Download and unzip the files in your project folder. Unzipping should result in two folders added to your project folder: *files-small* and *files-large*. Review the contents of the files before you start, to familiarize yourself with their structure.

The program that you write must work as follows.

1. Ask the user to specify the subfolder in the current working directory, where the files are stored.
2. Ask the user for a course number, outputting the number of eligible students in response. Incorrect course numbers should be ignored and the program should go on. When the user does not specify any course number, pressing enter instead, the program should stop.

The sample interaction demonstrates the running of the program.

Sample interaction

```
Please enter the name of the subfolder with files: files-small
Enter course number or press enter to stop: 1100
There are  17  students who could take course  1100 Charms.
Enter course number or press enter to stop: 2300
There are  2  students who could take course  2300 Care Of Magical Creatures.
Enter course number or press enter to stop: 2470
There are  19  students who could take course  2470 Magical Theory.
Enter course number or press enter to stop: 45
There are  0  students who could take course  45 None
Enter course number or press enter to stop:
```

Important Notes and Requirements

1. Your program should not use any global variables and should have no code outside of function definitions, except for a single call to `main`.
2. All file related operations should use device-independent handling of paths (use `os.getcwd()` and `os.path.join()` functions to create paths, instead of hardcoding them).
3. You must define and use functions specified below in the **Required Functions** section. You may and should define other methods as appropriate.
4. You should use data structures effectively to efficiently achieve the goals of your functions and programs.

Required Functions

You **must** define and use the following functions, plus define and use others as you see fit:

a. Function `processProgramFile()` with a single parameter of type str, providing the path to a program description file. The function should construct a dictionary based on the information in the file. The dictionary should have keys equal to the course numbers, and values for keys equal to the corresponding course titles, e.g.

```
{'1001': 'Transfiguration.', '1100': 'Charms.', '1250': 'Defense Against The
Dark Arts.', '1380': 'Potions.', '1420': 'Arithmancy.', '2075': 'Flying.'}
```

The function must return a tuple, consisting of the program name and the created dictionary.

b. Function `processPrereqsFile()` with a single parameter of type str, providing the path to a file defining prerequisites. The function should construct a dictionary based on the information in the file. A line in the file of the form `1250: 1001 1100` indicates that 1250 has two prerequisite courses: 1001 and 1100. The dictionary should have keys equal to the course number, and values for keys equal to the corresponding prerequisite courses. Only courses that have prerequisites should be included in this dictionary.

The function must return the constructed dictionary.

c. Function `processClassFiles()` with a single parameter, defining the subfolder with the class list files, as outlined in the **Data** section. The function should construct a dictionary with keys corresponding to course numbers. The value for each key must be equal to the set of students who have taken the course designated by the key.

The function must return the constructed dictionary.

d. Function `initFromFiles(),` with a single parameter, defining the subfolder with the files. The method should call functions specified in items a-c above to process data provided in the files. The function should return a tuple with the constructed dictionaries for program courses, class lists and prerequisites.

e. Function `estimateClass(),` which will be passed a course number and other parameters of your choosing, and which will return a set of students who would be eligible to take the course, specified by the parameter, in the next semester. If the parameter does not refer to a valid course number, the function should return an empty set. Note: this function should not be working with files, but should get all needed information from the appropriate data structures created from files.

f. Function main(), which will be called to start the program and works according to your design.

Grading

The grading schema for this project is roughly as follows:

Your program should compile without syntax errors to receive any credit. If a part of your program is working, you will receive partial credit, but only if the program compiles without syntax errors. Your program will be graded on the correctness of the output, and conformance to the requirements, including requirements on functions. 2 points will be awarded for good programming style, as defined in handout 1

Created by Tamara Babaian on October 23, 2018