EPPS/GISC 4317/6317: Programming for Social Science

**Instructor: Dr. Bryan Chastain**

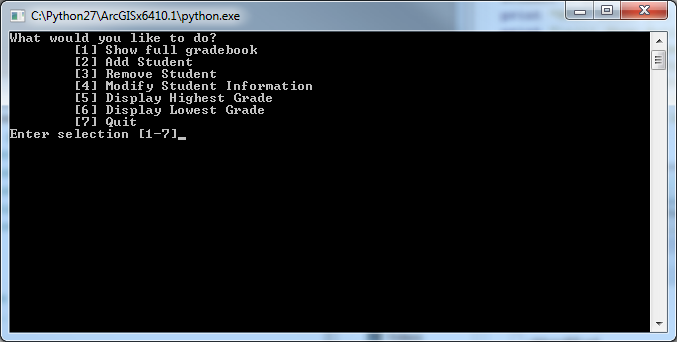
**Lab 03: Lists**

OBJECTIVES

In this project, you will create an application that utilizes a list-in-list structure to store the names and letter grades of students taking an exam.

View the finished application

* Download lab3soln.exe from eLearning, then double-click to run. The Grade application opens.



* Using option “2”, enter a student name and a letter grade. Enter a few more sets of names and grades. Try using option “1” periodically to check your work.
* Use option “4” to change student names and/or grades.
* Use option “3” to remove a student
* Use options “5” and “6” to display the student with the highest and lowest grades, respectively.

**Open student.py**

Download student.py from eLearning to your local directory. After copying, open this student.py with Spyder.

Inside this file is the code to print the menu as shown in the demo program, as well as prompt the user for input. There is also a nested if/elif statement to catch and perform code for each of the different menu entries.

**Create a while() loop**

* As it’s currently written, it will only prompt the user for menu selection once, perform that code, and then quit. However, in that setup, all data entered would be lost after the execution. There would be no way to go back and “Show full gradebook” after Adding a student.
* Instead, we need to change the structure so that it keeps looping until the user quits.
* Before the “menu=int(…)” line, create a new while loop, that keeps looping while the input is not “7”:
  + *#loop until the user decides to quit*
  + **while** menu != 7:
* However, the loop is still not set up, as Python requires loop bodies to be indented. Highlight all of the lines after the loop statement but before the “Terminating” statement and press Tab.
* Save and now try running the program. You should be able to keep going through the menu selection process until you press “7.”

**Create a list**

* The application will store the student names and grades in a list. Because you do not know how many students will be entered, it will need to be dynamic in nature. Because the values in the list will need to be accessed no matter what menu option is selected, you must declare the variable **outside the while() loop**.
* Somewhere at the top of the student.py file, initialize a new “student” variable as an empty list

student = []

**Store student names in the list**

* Next, you will write the code that adds the student name to the list. Later, you will add code to store the grade as well.
* The code to add a student will go in the “elif” code for option 2 (“Add Student”). Delete the “print menu” statement currently inside that codeblock.
* Prompt the user to enter the student’s name with the input() function.
* Prompt the user to enter the student’s grade with the input() function that is then converted to an integer.
* Append this name to the student list. (We will add the grade later.)

**elif** menu == 2:

*#prompt user for student name*

sname = input("Student Name?")

*#prompt user for student grade*

sgrade = int(input("Student Grade?"))

*#append student information to list*

student.append(sname)

**Print the gradebook**

* In this step, you will add code to display all student information
* The code to add a student will go in the “elif” code for option 1 (“Show full gradebook”). Delete the “print menu” statement currently inside that code block.
* Give the output column titles of “Name” and “Grade,” separated by a tab.
* Create a new For loop to loop through all students in your list
* Print the information for each student

**if** menu == 1:

**print**("Name\tGrade")

*#loop through all students*

**for** s **in** student:

**print**(s)

* Save and run the application. You should be able to add student names to the list and use the “Show full gradebook” option to inspect the contents. Notice that this list stores only student names; the grades are not shown.

Create a multi-dimensional list (a list of lists)

* Currently, “student” is a 1-D list; it stores only the student name. In this step, you will convert the student list to a 2-D list, so that the array can store both student names and grades. The first dimension will hold the two attributes for a given student--name and grade--while the second dimension will hold the index for that student. The first dimension always contains two elements (name and grade). The second dimension increases by one every time a new student is added to the list.
* Navigate to the “Add Student” code block (option 2). Edit the append statement so that instead of just storing a name, it stores a new list containing a student’s name and grade.

student.append([sname,sgrade])

**Jupyter for prototyping**

* We could continue developing this application completely in Spyder and it would be just fine. However, let’s try opening up Jupyter and using that for prototyping and testing our logic. Open Jupyter, create a new Python 3 notebook and name it student.
* For our testing we won’t need to print the menu text. Also, we won’t need to loop over the user selection. We will separate each of our menu item code blocks into their own cells in our notebook to be run independently. This will come in handy, especially while working on your homework.
* Let’s set up the first code block with some test data so we don’t have to keep entering it in every time we want to test out our code (you can use whatever names & grades you like):

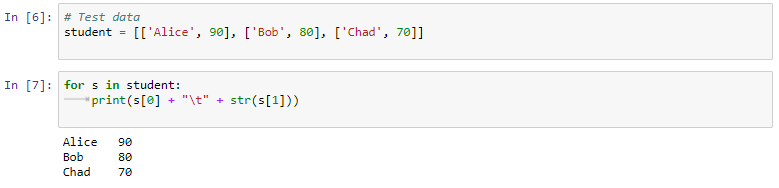
*# Test data*

student = [['Alice', 90], ['Bob', 80], ['Chad', 70]]

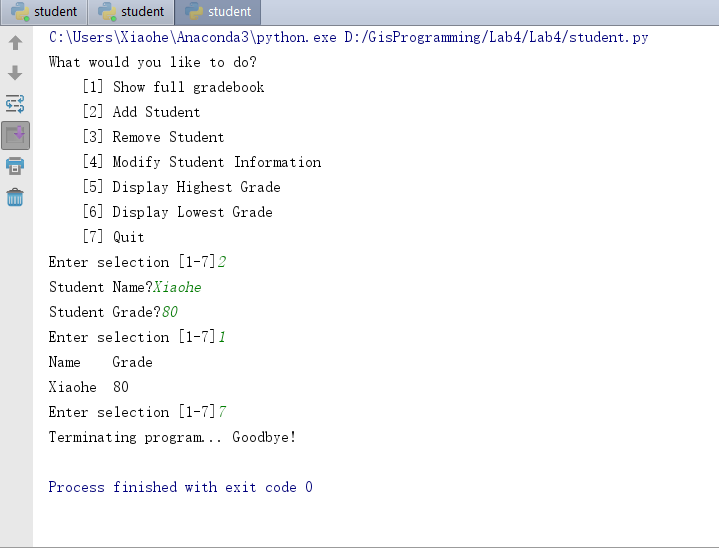
* Run that cell and then in the next cell we will test out how to print a multi-dimensional list:

**for** s **in** student:

**print**(s[0] + "\t" + str(s[1]))



* Try running this cell. It loops through all the students in the list and sets them to the “s” variable one at a time. Then for each student it prints out their name (0th sub-item) and grade (1st sub-item).
* Once this is working properly, copy these two lines from Jupyter and paste to replace the current “Show full gradebook” code back in Spyder.
* Save and run the application in Spyder.
* You should be able to add the student names and grades to the 2-dimensional list and see the results in option 1. Notice that this list now stores both the student names and grades.
* Close the application and save the project.



**Homework (use Jupyter to test out the logic for each of the option codeblocks before using in Spyder):**

**Undergraduate (GISC 4317)**

* Implement the “Remove Student” option as shown in the demo program (option 3). It should prompt the user to ask what student to remove, and then remove ONLY that student. If the user enters a name that is not in the database, it should give an error (e.g “your input is invalid”.)
* Implement the “Modify Student Information” option as shown in the demo program (option 4). It should prompt the user to ask what student to modify, and then modify ONLY that student. If the user enters a name that is not in the database, it should give an error. If the user doesn’t enter a value for name or if he/she doesn’t enter a value for grade, it should default to the original value.

Submit your python file (.py) with your name and course number (4317 or 6317) in comments at the top to eLearning.

**Graduate (GISC 6317)**

* Implement the “Remove Student” option as shown in the demo program (option 3). It should prompt the user to ask what student to remove, and then remove ONLY that student. If the user enters a name that is not in the database, it should give an error (e.g “your input is invalid”).
* Implement the “Modify Student Information” option as shown in the demo program (option 4). It should prompt the user to ask what student to modify, and then modify ONLY that student. If the user enters a name that is not in the database, it should give an error. If the user doesn’t enter a value for name or if he/she doesn’t enter a value for grade, it should default to the original value.
* Implement the “Highest Grade” option (option 5). It should report the name and grade of the student who has the highest grade. It should not crash if there are no students.
* Implement the “Lowest Grade” option (option 6). It should report the name and grade of the student who has the lowest grade. It should not crash if there are no students.
* Finally, check the menu selection input to ensure the user has given a valid response. If the user enters a letter or a number other than 1-7, it should give an error, but not crash.
* Check your code with pep8online.com to ensure PEP8 compliance.

Submit your python file (.py) with your name and course number (4317 or 6317) in comments at the top to eLearning.

Tip for both: Include a description of the code at the top of the script in what is known as a module docstring. For consistency, always use """triple double quotes""" around docstrings. Also, you can include your name/email in a metadata variable called \_\_author\_\_. Example:

