# Python Chapter 6’s Assignment – Student Records System

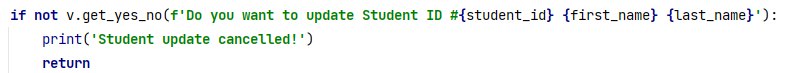
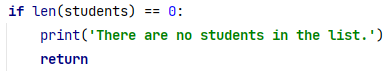
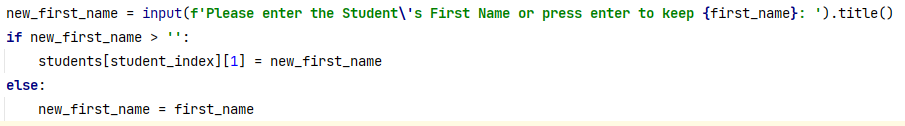
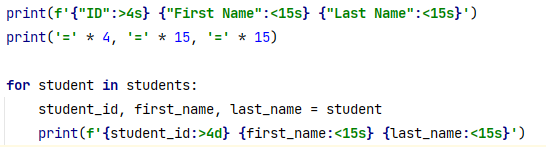
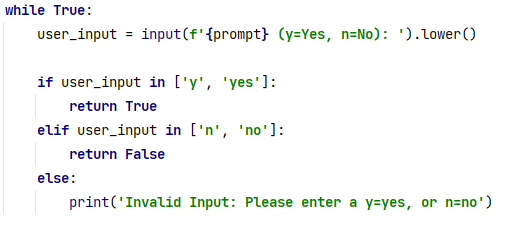
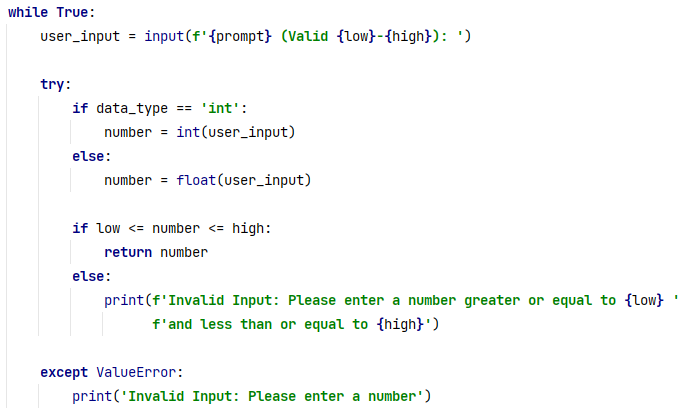
## Instructions

As a program developer, it’s your job to follow the design specifications given you to create a program or application that is exactly what was requested for by the end user. For this assignment, you must create a program that will store student’s IDs, plus first and last names in a two-dimensional list. You will want to use book\_apps\ch06\movie\_list\_2d.py as a starting point.

Here is a list of some of the additional logic you will need to add to your program that wasn’t included in the movie\_list\_2d.py module:

1. Change the menu to use numbers instead of words for menu selection.
2. Store student id, first name, and last name (no reusing student ids after delete)
3. Add an update student option
4. Add data validation for all data inputs
   1. Use the title method to upper case the first and last name
   2. Allow the user to skip updating either the first or last name
5. Add confirmation to the update or delete, so the end user has to confirm the action
6. Use a main menu scripts, plus data validation and student maint modules
7. Add docstring to all functions PLUS inline documentation too
8. Make sure your code is easy to read and have no PyCharm warnings  
   This means adding blank lines to your code between specific statement groupings
9. UI is VERY important, so use blank lines, dash lines, and indentation for a more readable interface
10. Use GitHub with regularly switch driver/navigator commits & pushes

Assignment Tips:

* At the end of the while loop before displaying the menu again:  
  
* Avoid nested constructs when possible, examples:  
  NOTE: all methods (except add) should first check to see if this 2D list is empty  
  
* Do not force the user to re-enter the data if they don’t want to change all the data:  
  NOTE: use the title method for names  
  
* User f-strings whenever possible  
  
* If the user didn’t change any data, then display a more appropriate message:  
  
* User to confirm all changes:  
  
* All data input should have some validation. Example:  
  

## Output Example

**Student Menu**

**======================**

**1 - List all students**

**2 - Add a student**

**3 - Update a student**

**4 - Delete a student**

**0 - Exit program**

**Please enter a Menu # (Valid 0-4):** 5

Invalid Input: Please enter a number greater or equal to 0 and less than or equal to 4

**Please enter a Menu # (Valid 0-4):** 1

There are no students in the list.

Press Enter to continue...

**Student Menu**

**======================**

**1 - List all students**

**2 - Add a student**

**3 - Update a student**

**4 - Delete a student**

**0 - Exit program**

**Please enter a Menu # (Valid 0-4):** 2

**Add Student**

**-----------**

Please enter the Student's First Name: mary jean

Please enter the Student's Last Name: smith

Student ID #1 Mary Jean Smith was added.

Press Enter to continue...

**Student Menu**

**======================**

**1 - List all students**

**2 - Add a student**

**3 - Update a student**

**4 - Delete a student**

**0 - Exit program**

**Please enter a Menu # (Valid 0-4):** 3

**Update Student**

**--------------**

Please enter the Student ID to be updated: 1

Do you want to update Student ID #1 Mary Jean Smith (y/n): y

Please enter the Student's First Name or press enter to keep Mary Jean: \_

Please enter the Student's Last Name or press enter to keep Smith:­ \_

Student's name was not changed. Update was cancelled.

Press Enter to continue...

**Student Menu**

**======================**

**1 - List all students**

**2 - Add a student**

**3 - Update a student**

**4 - Delete a student**

**0 - Exit program**

**Please enter a Menu # (Valid 0-4):** 2

**Add Student**

**-----------**

Please enter the Student's First Name: sam

Please enter the Student's Last Name: johnson

Student ID #2 Sam Johnson was added.

Press Enter to continue...

**Student Menu**

**======================**

**1 - List all students**

**2 - Add a student**

**3 - Update a student**

**4 - Delete a student**

**0 - Exit program**

**Please enter a Menu # (Valid 0-4):** 2

**Add Student**

**-----------**

Please enter the Student's First Name: debbie

Please enter the Student's Last Name: dolittle

Student ID #3 Debbie Dolittle was added.

Press Enter to continue...

**Student Menu**

**======================**

**1 - List all students**

**2 - Add a student**

**3 - Update a student**

**4 - Delete a student**

**0 - Exit program**

**Please enter a Menu # (Valid 0-4):** 1

ID First Name Last Name

==== =============== ===============

1 Mary Jean Smith

2 Sam Johnson

3 Debbie Dolittle

Press Enter to continue...

**Student Menu**

**======================**

**1 - List all students**

**2 - Add a student**

**3 - Update a student**

**4 - Delete a student**

**0 - Exit program**

**Please enter a Menu # (Valid 0-4):** 3

**Update Student**

**--------------**

Please enter the Student ID to be updated: asdf

Invalid Input: Please enter a number

Please enter the Student ID to be updated: 10

Student ID #10 was not found.

Press Enter to continue...

**Student Menu**

**======================**

**1 - List all students**

**2 - Add a student**

**3 - Update a student**

**4 - Delete a student**

**0 - Exit program**

**Please enter a Menu # (Valid 0-4):**): 3

**Update Student**

**--------------**

Please enter the Student ID to be updated: 3

Do you want to update Student ID #3 Debbie Dolittle (y/n): y

Please enter the Student's First Name or press enter to keep Debbie: Debby

Please enter the Student's Last Name or press enter to keep Dolittle: ­\_

Student ID #3 Debbie Dolittle was update to Debby Dolittle

Press Enter to continue...

**Student Menu**

**======================**

**1 - List all students**

**2 - Add a student**

**3 - Update a student**

**4 - Delete a student**

**0 - Exit program**

**Please enter a Menu # (Valid 0-4):** 4

**Delete Student**

**--------------**

Please enter the Student ID to be deleted: 2

Please confirm deleting Student ID #2 Sam Johnson (y/n): y

Student ID #2 Sam Johnson was deleted.

Press Enter to continue...

**Student Menu**

**======================**

**1 - List all students**

**2 - Add a student**

**3 - Update a student**

**4 - Delete a student**

**0 - Exit program**

**Please enter a Menu # (Valid 0-4):** 1

ID First Name Last Name

==== =============== ===============

1 Mary Jean Smith

3 Debby Dolittle

Press Enter to continue...

**Student Menu**

**======================**

**1 - List all students**

**2 - Add a student**

**3 - Update a student**

**4 - Delete a student**

**0 - Exit program**

**Please enter a Menu # (Valid 0-4):** 0

Bye!

Help on module main\_menu:

NAME

main\_menu - Chapter 6 Assignment

DESCRIPTION

Student Data System for managing student information (student id, first name, and last name)

This module contains the functions for displaying the main menu and running the menu options

FUNCTIONS

display\_menu()

Displays a list of all the valid main menu options

It also handles for nonnumerical data and invalid menu option selected.

1 - List all students

2 - Add a student

3 - Update a student

4 - Delete a student

0 - Exit program

:return no value

:rtype none

main()

Main keeps the program looping until the user enters 0 to exit the program

then based on the user's selected, will call the corresponding function option

Local scoped students is a 2D list that is pass as an argument to each menu option function

Local scoped max\_student\_id is the last student id used, and is passed to the add\_student function,

and this function will return the last added student id

:return no value

:rtype none

VERSION

1.0

DATE

2021.10.21

AUTHOR

Debbie Johnson

Help on module student\_maintenance:

NAME

student\_maintenance

DESCRIPTION

Chapter 6 Assignment

This module contains the functions for adding, updating, and deleting student data

FUNCTIONS

add\_student(students, next\_student\_id)

Display the all student information stored in a 2D list. It will increment the last student id by one

and use it as the new student's id. It also, displays that the student was successfully added.

:param students: student data (id, first\_name, last\_name)

:type students: 2d list

:param next\_student\_id: the next student id to be used for the add function

:type next\_student\_id: int

:return no value

:rtype none

delete\_student(students)

It will first check to see if there is any student data, and notify the user if no data is found.

It will then prompt the user for a valid student ID to be deleted from the 2D list

It handles for non numeric data, and student IDs that do not exists via the find\_student\_index

It will prompt the user to confirm they want to delete the selected student, and then let the user know

if the user was successfully deleted.

:param students: student data (id, first\_name, last\_name)

:type students: 2d list

:return no value

:rtype none

find\_student\_index(students, student\_id)

Search the 2D list for a specific student ID.

CODE EXAMPLE:

for student in students:

if student\_id in student:

return students.index(student)

return -1

:param students: student data (id, first\_name, last\_name)

:type students: 2d list

:param student\_id: student id that they user wants to find

:type student\_id: int

:return the index of the student in the 2D list or -1 if not found

:rtype int

list\_students(students)

Display the all student information stored in a 2D list (id, first name, last name)

It will notify the student if there is no data found.

:param students: student data (id, first\_name, last\_name)

:type students: 2d list

:return no value

:rtype none

update\_student(students)

It will first check to see if there is any student data, and notify the user if no data is found.

It will then prompt the user for a valid student ID to be updated from the 2D list

It handles for non numeric data, and student IDs that do not exists via the find\_student\_index

It will prompt the user to confirm they want to update the selected student, and then let the user know

if the user was successfully updated.

:param students: student data (id, first\_name, last\_name)

:type students: 2d list

:return no value

:rtype none

VERSION

1.0

DATE

2020.10.06

AUTHOR

Debbie Johnson

Help on module validation:

NAME

validation - the validation module for all numeric input.

FUNCTIONS

get\_num(prompt, data\_type='int')

Display an input prompt to get a number from the user, and convert it to a int or float

and loop again if invalid

:param prompt: the text that will be used for the user's input prompt

:type prompt: string

:param data\_type: a string indicating how the user input will be converted to (default=int or float)

:type data\_type: string

:return the user's input as a number that will be either a int or float

:rtype int or float

get\_positive\_num(prompt, data\_type='int')

Display an input prompt to get a positive number from the user, and convert it to a int or float

and loop again if invalid

:param prompt: the text that will be used for the user's input prompt

:type prompt: string

:param data\_type: a string indicating how the user input will be converted to (default=int or float)

:type data\_type: string

:return the user's input as a number that will be either a int or float

:rtype int or float

get\_range(prompt, low, high, data\_type='int')

Display an input prompt to get a number from the user within an acceptable range, and convert it to a int or float

and loop again if invalid

:param prompt: the text that will be used for the user's input prompt

:type prompt: string

:param low: the lowest possible numeric value

:type low: int

:param high: the highest possible numeric value

:type high: int

:param data\_type: a string indicating how the user input will be converted to (default=int or float)

:type data\_type: string

:return the user's input as a number that will be either a int or float

:rtype int or float

get\_string(prompt)

Display an input prompt to get a string value, which can not be an empty string

and loop again if invalid

:param prompt: the text that will be used for the user's input prompt

:type prompt: string

:return the user's input as a string

:rtype string

get\_yes\_no(prompt='y=Yes, n=No')

Display an input prompt to get a yes/no answer from the user (valid lower cased inputs are y, yes, n, no)

and loop again if invalid

:param prompt: user's prompt if one is passed otherwise the default is

:type prompt: string

:return true for yes or false for no

:rtype bool

DATA

\_\_status\_\_ = 'Development'

VERSION

1.0

DATE

2021.09.30

AUTHOR

Debbie Johnson