CICS-160 Spring 2023
Assignment 2: Abstract Data Types
Due on Sunday March 12 2023.

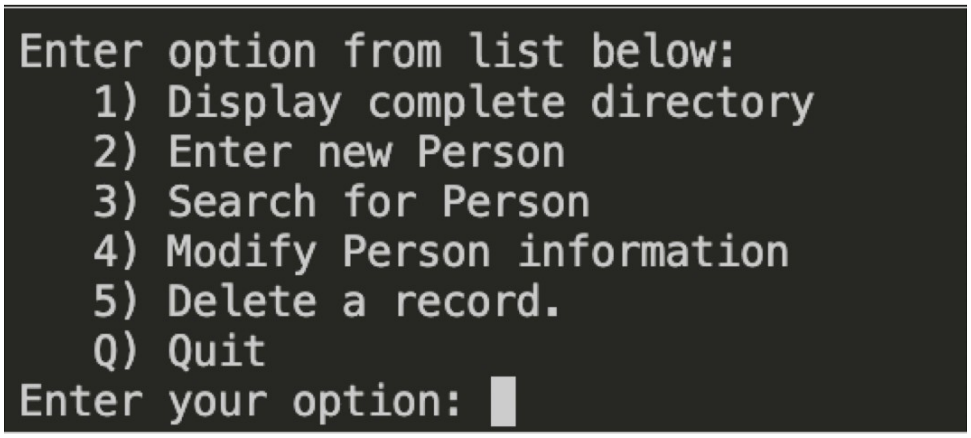
Learning Goals:

This assignment is designed to give you an opportunity to practice with the design of a series of classes that work together in order to solve a particular problem, and how to state the specifications of such a solution via Abstract Data Types (ADTs). This is an assignment centered on design, not implementation. You do not have to implement the classes you design, or the main menu system described further down. Do only have to design those parts. The only code you have to write relates to the process of testing mentioned at the end of the paragraph describing your tasks. Because you will not have the classes or the main program implemented, you will not be able to "test your testing code", so we'll be flexible with that in terms of syntax errors, etc. You will be turning in a typed document describing your design. This assignment will be 100% manually graded.

Overview

For this project, you will be designing a system that will present a user with a menu-driven version of an extension to the list of persons software we have been commonly using as an example during our course so far. The system needs to give the user the opportunity to add, delete, modify, and list the elements of a list of objects of type Person. If you want, you can use, as a starting point, any idea/concept/code we have used or talked about in class. Your design should include ADTs for class Person and class Persons, indicating the name and signature for each of the methods. Your design should also indicate, for class Persons, which methods of class Person it makes use of, if any. Finally, for either class, you should indicate which dunder methods (__eq__, __str__, __lt__, etc) you will be implementing, if any, and based on what they will do their work (i.e. if you decide you will implement a __lt__ method, you will need to indicate based on what that method will return different values. You do not have to implement any methods; you only have to indicate their name, signature, and which outside methods (from another class, or a dunder method) they will use. Finally, if your design uses any library operations, you should indicate which ones those will be. For example, indicate if your design makes use of random.randint(), or string.upper(). <u>You do not have to implement any of these functions or methods.</u>

The behavior of the system, when completed, will be as follows. The user will be presented with a menu of options from which to choose, each one of them implementing an operation on a list. The operations are: entering a new record; listing all records; displaying a particular record; modifying a particular record; and deleting records. A possible menu might look as displayed in the following image. Your code will start by validating the input entered by the user, asking them to enter another option if the value they enter does not match any of the options presented. All throughout the design, whenever letters are to be entered by the user, the program should behave correctly regardless of that input being uppercase or lowercase.



```
Enter option from list below:
   1) Display complete directory
   2) Enter new Person
   3) Search for Person
   4) Modify Person information
   5) Delete a record.
   Q) Quit
Enter your option: 
```

The search operation, operation #3, will ask for a string, and will then display every record that contains that string as part of their name. Operations #4 and #5 will show the user all records with a name that contain a particular string, asking the user if they want to modify/delete that record, as appropriate.

Your tasks

Based on the operations listed in the main menu shown above, design a class called Person and a class call Persons, both of which are continuations/expansions of the classes with those same names that we have been working with in class. Your design of these classes should include both what attributes and methods each of them will have. For the methods, you should indicate both what inputs they will take and what outputs they will return, if any. Finally, your design should include code to test two of the methods from each of the classes you are designing (two for class Person, two for class Persons, for a total of four). For now, these tests can be implemented either using the unittest class we have seen previously this semester, or by creating objects with specific data and printing them, as we have done in class, making sure that the data that gets displayed is the data we expect.

Upload three files to gradescope:

- person.pdf, which includes the ADT and other design specifications for class Person.
- persons.pdf, which includes the ADT and other design specifications for class Persons.
- tests.py, which implements the four tests you have chosen to provide tests for.