CS2321 Lab 3

Lab Instructions:
Save the code you write for each exercise in this lab as a *library* -- that is, a textfile with a .py extension containing only executable python code (i.e. no angle-bracket prompts, etc). Name each file according to the exercise number (e.g. ex1.py, ex2.py, etc.) and save them to a directory containing the report file, when completed, compress them together in a single zip file to be submitted on D2L.

Each function should have a docstring explaining what the function does.
Any Follow-up Questions and their Answers should be included in a **docstring** in the `main()` function.

e.g. the structure for a module contains a class should look like:

```
'''
   ex1.py
   Doc-string explaining what this module does
'''
# Other imports, such as math, random, etc., as needed

class MyProperClassName(Object):
   '''
       This is the doc-string explaining what this class does.
   '''
   def __init__(self, param1, param2, etc. ):
       '''
            Doc-string for the constructor.
       '''

       #Create instance variables using the parameters.

   #definitions of methods, make sure the first parameter is self.
   def method1(self, ......):
       '''
            Doc-string for each of the methods.
       '''
def main():
      myObject = MyProperClassName()
      myObject.method1(param1, param2, ...)
      ...
if __name__ == "__main__":
      main()
```

Lab Deliverable: Once all your programs run correctly, collect their code and the results of their test-cases in a nicely-formatted **PDF** file exported from Word Processing document (e.g. MS Word or LibreOffice are fine) to be included in the submission on D2L.

This **report** should consist of each lab exercise, clearly **labeled** in order, consisting of code, then copy/pasted text output, or, for GUI, screen-captured, of its four test-cases.
In this lab, take series of screen captures of your GUIs and insert them into the report.

Paired Programming:
We will work today's lab assignments in pairs -- on a single computer in one partner's account.
One person will start out as the *typist*, the other as the *verifier*. These roles will switch.
For each problem, partners should decide upon their proposed algorithm to solve the given problem *before* the typist begins to type. Sketch it out on a sheet of paper, perhaps.
For **ten** minute periods, the typist will type the code, while the other verifies and suggests corrections (typist has final decision). Under no circumstances may the verifier ever touch the mouse or keyboard. (Note: the *instructor* may not touch your input devices either!)
On the instructor's ten-minute signal, partners will trade responsibilities. This should allow both partners to benefit from each other's strengths. Future paired-programming labs will be with different partners, to spread the gained experience around.
Each partner should post the resulting code in their own D2L folder. You may transmit partnership-generated code to the other partner (only!) by email or thumb-drive.

Exercises

1. Implement Project #1 from page 349.

Add **three(3)** methods to the **Student** class that compare two(2) **Student** objects. One method should test for ***equality***. A second method should test for ***less than***. The third method should test for ***greater than or equal to***. In each case, the method returns the result of the comparison of the two students' names. Include a **main** function that tests all the comparison operators.

2. Implement Project #2 from page 349.

This project assumes that you have completed Project 1. Place several **Student** objects into a list and shuffle it. Then run the **sort** method with this list and display all of the students' information.