CS2321 Lab 2

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.
For this lab, make sure you include breezypythongui.py together with your python module(s) in the zip file.

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 GUI class should look like:

```
'''
   ex1.py
   Doc-string explaining what this module does
'''

from breezypythongui import EasyFrame

# Other imports, such as math, random, etc., as needed

class MyProperClassName(EasyFrame):
   '''
       This is the doc-string explaining what this class does.
   '''
   def __init__(self):
       '''
            Doc-string for each of the methods, including the constructor.
       '''
       EasyFrame.__init__(self, title = 'A proper title to be displayed')

       #Create and place widgets properly.

   #definitions of event-handling methods, make sure the first parameter is
      self.
   def method1(self, ......):
       '''
            Doc-string for each of the methods, including the constructor.
       '''
   def method2(self, ......):
       '''
            Doc-string for each of the methods, including the constructor.
       '''
```

```
def main():
      MyProperClassName().mainloop()

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 #3 from pages 291.
Write a GUI-based program that allows the user to convert temperature values between degrees Fahrenheit and degrees Celsius. The interface should have labeled entry fields for these two values. These components should be arranged in a grid where the labels occupy the first row and the correspobding fields occupy the second row. At start-up, the Fahrenheit field should contain 32.0, and the Celsius field should contain 0.0. The third row in the window contains two command buttons, labeled >>>> and <<<<. When the user presses the first button, the program should use the data in the Fahrenheit field to compute the Celsius value, which should then be output to the Celsius field. The second button should perform the inverse function. Format the floating point numbers to have one(1) digit after the decimal point before they are displayed.

2. Implement Project #5 from pages 291.
Write a GUI-based program that plays a guess-the-number game in which the roles of the computer and the user are the reverse of what they are in the Case Study of this chapter. In this version of the game, the computer guesses a number between 1 and 100 and user provides the

responses. The window should display the computer's guesses with a label. The user enters a hint in response, by selecting one of a set of command buttons labeled **Too small**, **Too large**, and **Correct**. When the game is over, you should disable these buttons and wait for the user to click **New game**, as before.