COM110 Fall 2019: Lab 9
Creating our own Classes (and using them)

In lab today we will be creating a dice roller program, as discussed in Chapter 10 of your text. Start by downloading and testing diceroller.pyc to see a sample of what we will be creating. (you can find all the necessary files for this lab in the downloaded lab9_username folder). For now, put it on your desktop along with a copy of graphics.py. On a Windows machine, just double-click the diceroller.pyc file. On a Mac, open the Terminal application (use the search option at the top-right of your Mac window - start typing Terminal to search), and in the Terminal window type `cd ~/Desktop` and hit enter to change the current directory to your Desktop, then type `python3 diceroller.pyc` then hit Enter on your keyboard.

Notice that the Quit button of the dice roller is initially disabled or "deactivated." If it is clicked on initially, nothing happens. Only when the "Roll" button is clicked does the Quit button become enabled or "activated." Then the user can click it to close the program.

1) **Button class.** We have written a function in class for drawing buttons, but it is useful to instead have a Button *class* that includes features/methods like checking if the Button object is clicked, "activating" the button, and "deactivating" the button (i.e., making it seem disabled), etc.
   a) Open the button.py file, part of lab9_username. It is a slightly more complete version of what we started in class today. *You should use this class for all button drawing that you may do for the rest of the semester. (E.g., it should be used for programming assignment 5.)* Read the code carefully. Notes or places you must add code for this lab checkpoint are marked with ##.
   b) Start by creating a graphical window in which to test the Button class in the main() function.

☺Get check 1☺

   c) Test the Button constructor method by creating two buttons, as specified by the ## comments. Also test the activate() method on the Roll button.

☺Get check 2☺

   d) Now complete the code for the deactivate() method. See the code for the activate() method as a reference. Test it on the Quit button you made above.

☺Get check 3☺

   e) Complete and test the boolean method .isClicked().

☺Get check 4☺

   f) Remove your test code from check 4. Now add code to continuously take user mouse clicks until the Quit button is clicked. Upon a click of the Roll button, the Quit button should be activated, as in the sample diceroller.pyc program demo. When the Quit button is clicked, the window should close. To accomplish this, follow the structure of the ## comments.

☺Get check 5☺

2) The DieView class:

a) Open dieview.py.

b) Read the documentation and comments so that you understand how to use the class. FYI, a "pip" is just one of the dots on a die. There is nothing in the code that you don't already have the skills to understand completely, so call us over if you have *any* questions about what *any* of the code does. (BTW, notice how unwieldy the setValue method is! So much code repetition, and giant if blocks. There is a much more streamlined version of this code… see bonus (a) below…)

c) Add a main function at the bottom of the module to test out the DieView class. In the main function create a graphical window, and create two DieView objects. These will look like two dice.

☺Get check 6☺

d) Play around with setting the value of these dice to certain numbers of your choice using the `.setValue()` method. For example, see if you can set the two dice to show two fives.

e) Then set them to two randomly generated integer values between 1 and 6.

☺Get check 7☺

3) The diceroller program. Create a new module called diceroller.py that imports the Button class from the button.py module and the DieView class from the dieview.py module. It should of course also import graphics.py. The rest of this new module diceroller.py consists of only a main() function in which you create the dice roller program you downloaded in diceroller.pyc. To do this, combine the test code you wrote in the main() functions of the previous two modules.

☺Get check 8☺

4) Bonus:

a) The Dieview class can be written much more compactly… see, for example, dieview2.py, which is from Zelle 11.4, page 383 (359 in 2nd eds.). It does the same exact thing as dieview.py, but it is written with better style… note the absence of the huge clunky if statement in the .setValue() method! (Zelle's explanation of this more streamlined code starts on page 379 (355 in 2nd eds.) at the start of 11.4.) Read this more streamlined version of the code (and Zelle's explanation if you'd like), then explain to us how the code works.

b) Extend your graphical dice roller program so that it can be used to play Craps. (See Chapter 9, programming exercise 7 on page 310 (294 in 2nd eds.) of Zelle, for the rules of the game.)

c) **Only if you've already completed bonus (a) above, you may also now take a shot at any bonuses from previous labs**.

☺Get bonus check☺