

Computer Science 112 - Fundamentals of Computer Science II

Lab Project 1

Due on Github 11:59 PM Monday 19 September

The Python program files you submit should have a format similar to the code in the lecture notes. In particular, your files must include docstrings for all modules, classes, and method definitions.

You should put your name(s), the file name, and project number in a Python triple-quote comment at the beginning of each file that you edit (this is also the module's docstring). You'll lose credit if you don't do this.

This work will consist of several files, including **ninecells.py**, **numberguessapp.py**, **numberguessgui.py**, and **numberguess.py**, with the names of collaborating students at the top of each file.

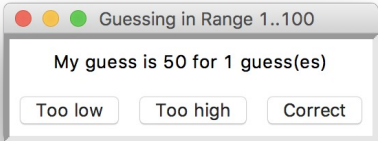
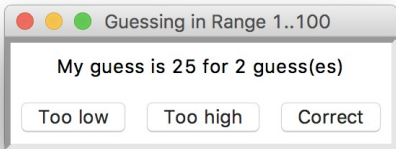
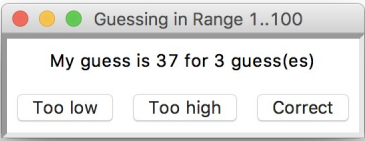
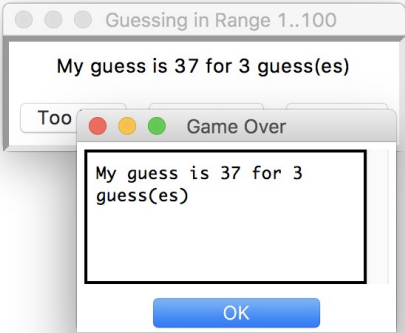
1. This folder contains some GUI programs that you can run as a warmup in exercises 2-8.
2. Open the **helloworld.py** file in IDLE3 and study its contents.
3. Edit the file to include your name and project number in the appropriate place.
4. Press F5 to load the module into the shell. Resize the program window, zoom it, and then minimize it. Restore the window and close it.
5. Return to the editor window and modify the program so that it centers the label in the window. Run the program again to test your changes. Then close the program window.
6. Return to the editor window and modify the program so that the label's color is red, its font size is 24 points, and its font weight is bold. Test to confirm the changes.
7. Repeat Exercise 4 with the files **layoutdemo.py** and **imagedemo.py**. Now edit the layout demo so that the labels are centered in their grid cells (resize the window to verify this).
8. Write a new program, whose file name is **ninecells.py**. This program's window should display the positions of a 3 by 3 grid, similar to the **LayoutDemo** program. You must use a loop to create the labels, rather than nine separate statements! The labels should be centered in their grid cells. ***Hint:*** use a nested **for** loop over the rows and columns of the grid.
9. The file **numberguesstui.py** contains code for launching a terminal-based game of guess the number. The computer thinks of a number between a lower bound and an upper bound (1 and 100 are the defaults), and prompts you for a guess until you guess correctly. You receive hints after incorrect guesses that allow you to guess intelligently, by going to the midpoint of

the remaining range of numbers each time. Study the code for this program and run it from IDLE or the terminal with the default range and then with a user-supplied range.

Ideally, you should need no more than $\log_2(\text{low} + \text{high} - 1)$ number of guesses to guess correctly. Modify the program so that the computer stops with a message if you exceed this number of guesses.

10. Now think about a program that switches the roles of the players in the guessing game of the previous exercise. In this new program, you, the user, think of a number between a given lower bound and a given upper bound, and the computer provides the guesses. The computer should be able to guess intelligently, too, so the user must provide hints after incorrect guesses.

A GUI-based program provides the following interface, reading from left to right:

	
At startup with the default range	After pressing Too high
	
After pressing Too low	After pressing Correct

The computer provides its first guess at startup, and the user provides responses via the command buttons. When the user presses **Correct**, a message box pops up and the program exits when the user closes this box.

The program's structure is similar to the GUI-based counter application discussed in lecture in included in this folder. The application startup code is in the file **numberguessapp.py** (similar to **counterapp.py**). You should complete the code for the classes **NumberGuess** and **NumberGuessGUI** in their respective module files, **numberguess.py** and **numberguessgui.py**. You should be able to complete all of the code for the **NumberGuess** class and test its methods before you connect it to the GUI.

***Be sure your name, the file name, and project number are in each file.** To submit your files, you can use the trick I showed you in class: Create a new empty file **CSCI112/Lab1/helloworld.py**, drag-n-drop **helloworld.py** to the drag-n-drop area, then simply upload all your other files into the same folder.*