**EPPS/GISC 4317/6317: Social Science Programming Fundamentals**

**Instructor: Dr. Bryan Chastain**

**Lab 7 - Create a Guessing Game Application**

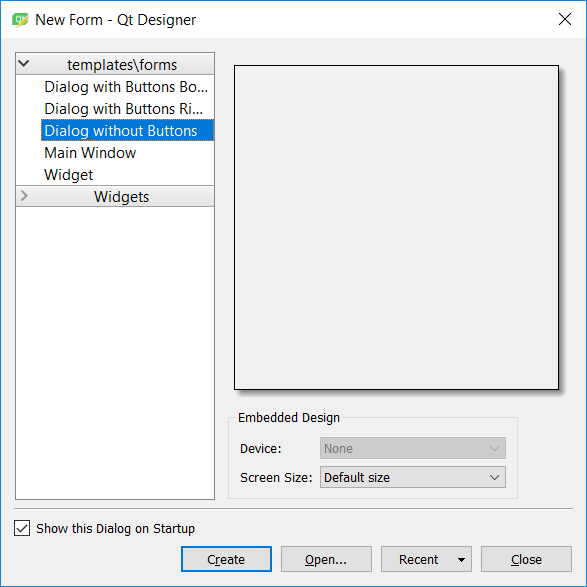
In this exercise, you will create a guessing game application using PyQt5. The game will pick a random number and allow players to guess the number. You'll also create a hint button to give players some clues about the correct number.

**View the finished application**

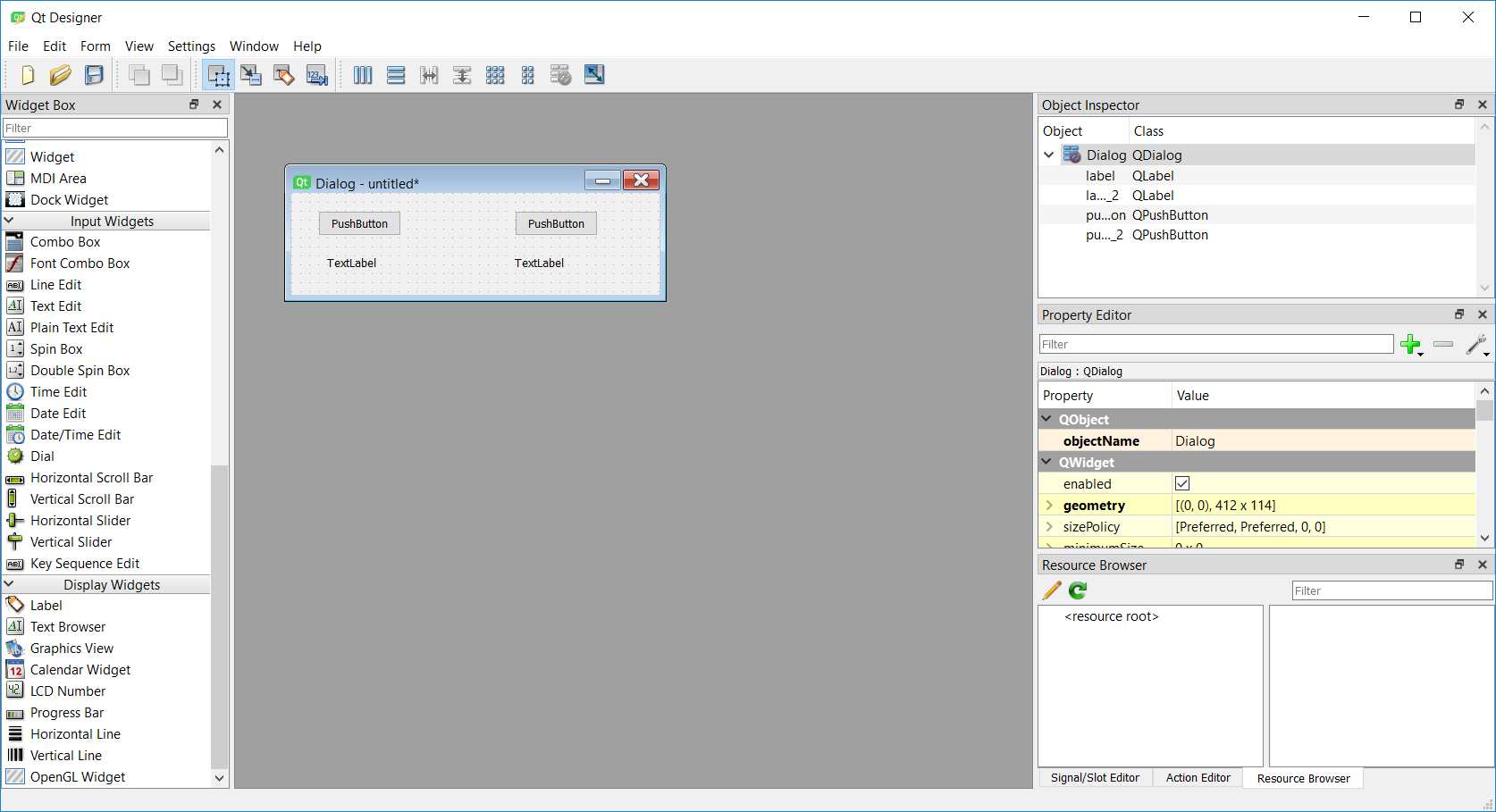
* Download the lab files and run the Lab10-**Guess.exe** executable. The Guessing application opens. Click the **Guess** button. The number of the guess is randomly generated and recorded. You get an input box, asking you to enter a number between 1 and 100. Enter a number and click OK. A message box displays to tell you whether your number is too big or too small.
* Dismiss the message box and click the Guess button to enter more guesses.
* Click the Give me a hint button. You're told that the correct number is within (plus or minus) 5 of a certain number. Notice that when you click the Give me a hint button, the number of guesses increases by 5. This is the price of a hint.
* Keep guessing until you get the correct number. How many tries did it take you? Notice how the game informs you that you're right.
* After you dismiss the message box, the number of guesses is reset to zero and you can start all over again. Close the guessing game application when you are finished.

**Design the form**

* In this lab, we will use QtDesigner & PyQt5 to design a Graphical User Interface (GUI) for the guessing game application. While GUI forms can be programmed directly with Python code, QtDesigner provides a visual interface for design, making the experience much easier and more intuitive.
* Open QtDesigner
  + Browse to either %localappdata%\anaconda3\Library\bin OR C:\Users\<YOUR USER NAME>\Anaconda3\Library\bin\ (depending on your computer) and open designer.exe
  + *Note: if you are on your personal computer, the Anaconda directory may be in a different location*
* Select a “Dialog without Buttons” project and press Create.



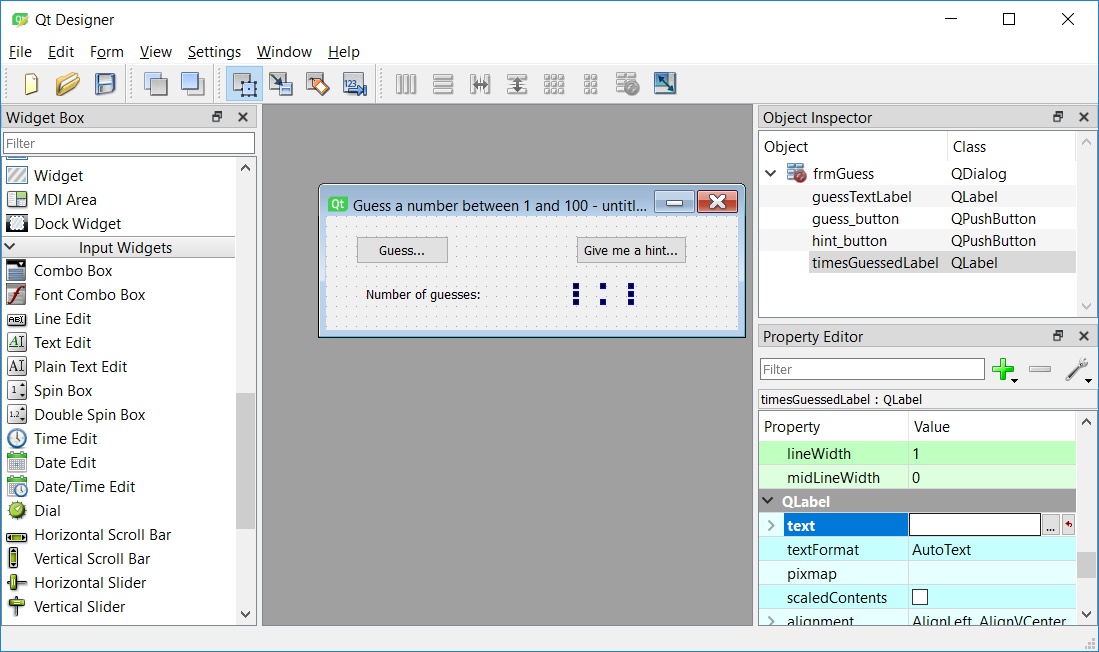
* Now we have our base form window to build our application on. Resize the window to the same approximate size as that of Lab10-Guess.exe
* Drag 2 Push buttons and 2 Labels on to the form, like so:



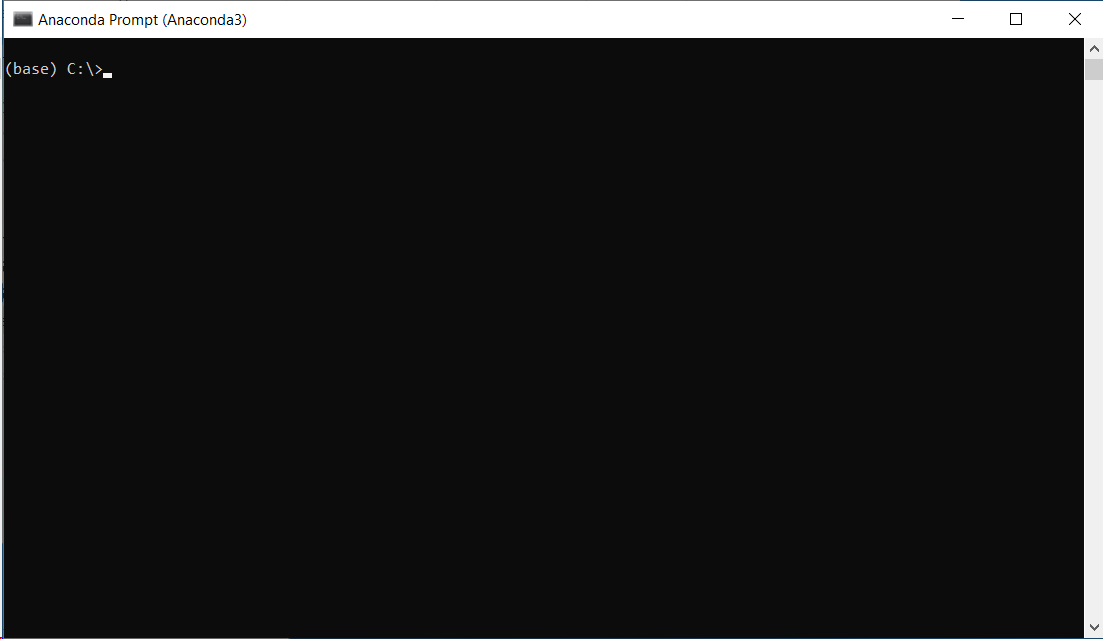
* Assign properties to the controls according to the following table.

|  |  |  |
| --- | --- | --- |
| **Control** | **Name** | Text |
| Form1 | frmGuess | Guess a number between 1 and 100 |
| Command1 | guess\_button | Guess... |
| Command2 | hint\_button | Give me a hint... |
| Label1 | guessTextLabel | Number of guesses: |
| Label2 | timesGuessedLabel | (blank) |

* It should end up looking like this:



* In QtDesigner, go to File->Save As and save it as lab7.ui in your lab 7 Spyder directory.
* This .ui file is just an XML file, but we can convert it to Python using the pyuic5 tool that comes installed with PyQt.
* In Windows, open Anaconda Prompt. It should open a window like this:



* In this window, type:

python –m PyQt5.uic.pyuic <mylab7directory>/lab7.ui -o <mylab7directory>/frmGuess.py

* You can close the command prompt and QtDesigner. Go back to Spyder and open the new frmGuess.py from your directory.
* The lab7.py file should contain a class with two functions: setupUi and retranslateUi. We will call this file from a new python file.
* Create a new Python file called lab7.py
* Set up the basic form as follows:
* In addition to the PyQt Widgets library, also import sys, random and our frmGuess file.

**from** PyQt5.QtWidgets **import** \*  
**import** random  
**import** sys  
  
**from** frmGuess **import** Ui\_frmGuess

* Declare variables

*# Declare global variables to store random number and times of guesses.*

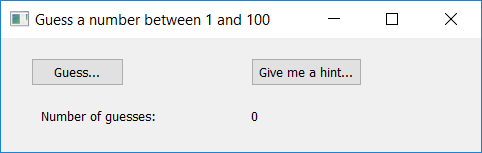
times\_guessed = 0  
num\_to\_guess = random.randrange(1, 101)

* Build the structure of the Mainwindow

**from** PyQt5.QtWidgets **import** \*  
**import** random  
**import** sys  
  
**from** frmGuess **import** Ui\_frmGuess  
  
*# Global variables to store random number and times guessed*times\_guessed = 0  
num\_to\_guess = random.randrange(1, 101)  
  
  
*# Create Main Window***class** Mainwindow(QWidget):  
 **def** \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.ui = Ui\_frmGuess()  
 *# Call UI set up in QtDesigner/pyuic5* self.ui.setupUi(self)  
 *# Reset times guessed label* self.ui.timesGuessedLabel.setText(str(times\_guessed))  
 *# Show form* self.show()  
  
  
*# Run main function on*

**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 app = QApplication(sys.argv)  
 ex = Mainwindow()  
 sys.exit(app.exec())

Run the program and you should get a window that looks like this:



It won’t function yet, as we haven’t programmed the Guess button code, but you’ve designed the basic look of it using PyQt.

**Declare and assign variable for the Guess button.**

* The code for the Guess button should accept an initial random number between 1 and 100 and allow the user to guess at the value, and report whether the guess was too low or too high. If the user's guess is correct, inform the user and allow the user to guess a new random number.
* We will need to create a new function to feed into the “command” parameter of the Guess button.
* In Python, functions cannot see variables outside of their scope by default. In order for the guess\_click() function to see these variables and to pass values between functions, we have to use the “global” statement to inform Python that we are referring to variables outside the function.
* You will need three global variables in the guess\_click function: one to store the random number from 1 to 100, one to store the times guessed, and the third to show the times guessed label. The times\_guessed and timesGuessedLabel change every time the user clicks the Guess button; therefore, their values are not preserved between the Guess button command function. The num\_to\_guess should preserve its value between the Guess button command function. Until the user guesses the correct number, the random number stays the same, while the times guessed increases by 1 for each guess.
* In the class Mainwindow() define a new function as follows:

**def** guess\_click(self):  
 **global** num\_to\_guess  
 **global** times\_guessed

* The global variable times\_guessed will store the count of the number of guesses the user has tried. The variable num\_to\_guess will store the random number range from 1 to 100.

guess, okPressed = QInputDialog.getInt(self, **"Enter Guess"**, **"Guess 1-100"**, 1, 1, 100, 1)  
**if** okPressed:  
 times\_guessed = times\_guessed + 1

* The QInputDialog.getInt function will accept user input from an input box and store the input to the local variable *guess*.
* When you click the Guess button, the value of okPressed becomes true. The number of guesses should be incremented by 1 and the result should be displayed in the UI function. We need to update the timesGuessedLabel by setting the times\_guessed variable to the setText parameter.

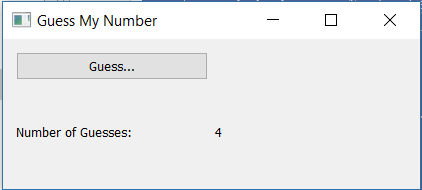
guess, okPressed = QInputDialog.getInt(self, **"Enter Guess"**, **"Guess 1-100"**, 1, 1, 100, 1)  
**if** okPressed:  
 times\_guessed = times\_guessed + 1

self.ui.timesGuessedLabel.setText(str(times\_guessed))

* Next modify \_\_init\_\_() to link the guess\_click function to guess\_button.

*# Call UI set up in QtDesigner/pyuic5*self.ui.setupUi(self)   
*# Connect Guess button to guess\_click function*self.ui.guess\_button.clicked.connect(self.guess\_click)

* Run the application. Click the Guess button a few times and notice the number change on the form.



**Evaluate the guesses**

* Now you will compare the local variable *guess* with *num\_to\_guess*. If they are the same, inform the user so, then generate a new random number and reset num\_guesses to 0. Otherwise, inform the user whether the guess is too big or too small.
* First, check to see if the *guess* variable is larger than the num\_to\_guess variable, display a message box informing the user the guessed number is too big. If the *guess* variable is less than the num\_to\_guess variable, inform the user the guessed number is too small. Otherwise, *guess* must be equal to num\_to\_guess, so you should inform the user the guess is right, reset times\_guessed and num\_to\_guess to 0.
* Second, try to write the code to generate a number between 1 and 100 after the correct guess. Remember, you do not want to generate a random number every time you click the Guess button, only when the correct guess is made.
* Your code should look like this.

guess, okPressed = QInputDialog.getInt(self, **"Guess"**, **"Enter guess 1-100"**, 1, 1, 100, 1)  
**if** okPressed:  
 times\_guessed = times\_guessed + 1  
 timesGuessedLabel.setText(str(times\_guessed))  
 **if** guess == num\_to\_guess:  
 QMessageBox.question(self, **"Congratulations"**, **"You guessed the right number in "** +

str(times\_guessed),QMessageBox.Ok)  
 *# Reset number to guess* num\_to\_guess = random.randrange(1, 101)  
 *# Reset times guessed* times\_guessed = 0  
 *#Reset label to 0* self.ui.timesGuessedLabel.setText(str(times\_guessed))  
  
 **elif** guess < num\_to\_guess:  
 QMessageBox.question(self, **"Low"**, **"Your guess is too low."**, QMessageBox.Ok)  
  
 **elif** guess > num\_to\_guess:  
 QMessageBox.question(self, **"High"**, **"Your guess is too high."**, QMessageBox.Ok)

* Run the application. You should be able to click the Guess button and guess at the answer. Message boxes should appear following each guess informing you whether your guess is too big or too small. When you get the correct answer, you should see a new message box stating "Congratulations, You guessed the right number in \*". When you dismiss the message box, you should see that the number of guesses is reset to 0, and you can start the next round of guessing.
* End the application and save the project.

**Homework:**

**Undergraduate:**

**a. Code the hint button**

In this step, you will code the **Give me a hint** button. The hint button will display a number in a message box that is within five of the actual value.

**Hint:** To do this, you need to generate a random number between -5 and 5. The sum of the new random number and your previous random number will be the hint. And make sure it works even if user click Give me a Hint button first! Remember, you want to keep your random hint number the same before the right guess and change the random hint number after the right guess.

**Graduate:**

**Do (a) as well as:**

**b. Add a penalty for getting hints**

The user will be penalized for clicking the **Give me a hint** button. Add code to increase the total number of guesses by five when the user clicks this button.