DS10 goal: gain some experience with tkinter

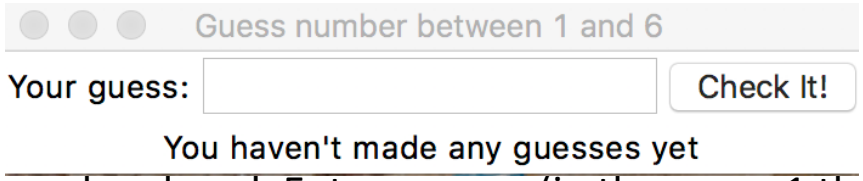
In this assignment, you'll start with code we provide, and we'll provide step-by-step instructions to add things to the program to make it a little better.

1. From http://www.cs.uiowa.edu/~cremer/courses/cs1210/#ds download
   - ds10.pdf, the pdf of this presentation
   - ds10game.py
2. Open ds10game.py and load it into Python

3. Enter the following at the Python prompt

   >>> startGameGUI(6)

   You should see a snall window like:



4. Now, a tedious number guessing game can be played. Enter a guess (in the range 1 through 6) in the Entry form in the GUI, and then press the Check It! button. If the GUI indicates that that is not the right answer, you can type in a new number and check again. Etc.

5. If you kill the window and restart the game via startGameGUI(20), the range will be [1,20] and it will be been even more annoying to enter numbers until you find the right one!

Next, we'll modify the code to make the game a little more playable/fun.

Modifying ds10game.py to make it a little more fun

We will improve the game in two ways:

1. We'll give better feedback to the user about the guess they've entered, displaying (in the GUI) whether the guess is too high or too low
2. When the user guesses correctly, the "Check It!" button will change behavior so that the user can press it to start a new game. (With the initial version of the game, you have to reload the code to play again.)

1. Giving better feedback about the user's guess

When the user enters a guess and presses the Check It! button, the "callback" function checkGuess is called.

Find that function in the Python file.

In the original version of the game, the user's guess is checked and simple feedback given via:

```
if guess == numberToGuess:
    statusLabel.configure(text = "That's it - you win!")
  else:
    statusLabel.configure(text="That guess wasn't right. Try again.")
```

1. Giving better feedback about the user's guess

Replace the if-else with an if-elif-else that gives better feedback:

```
if guess == numberToGuess:
    statusLabel.configure(text = str(guess) + " is it - you win!")

elif guess > numberToGuess:
    statusLabel.configure(text = str(guess) + " is too high. Try again.")
else:
    statusLabel.configure(text = str(guess) + " is too low. Try again.")
```

1. Giving better feedback about the user's guess

Now test the game. It should already be easier to play.

Now you can use 10 or even a larger number because with the new feedback you can use binary search to guess the answer much more quickly!

Before adding the second feature (ability to start new game), let's add one more little improvement.

So far, every time the guess is incorrect, the user needs to delete/erase the previous guess before typing a new one.

Instead, we can erase it for them at the end of checkGuess. Just add the line:

```
guessEntry.delete(0, tkinter.END)
```

(*Not* inside any part of the if-elif-else. After it, at the same level)

Test the code again. The game should be a little easier to use now!

2. Making it possible to start a new game

We'll add a feature that allows starting a new game from the GUI. We'll only allow a new game to start after the user successfully guesses.

How? When the user successfully guesses the number, we will temporarily change the GUI's only button. We will

1) Change the button's text so that it says "New Game"
2) Change the button's behavior so that it calls a newGame function instead of the checkGuess function

Then, when the user presses the button, the newGame function (which we must define) will be called, which will do several things:

1) Initialize a new game (by picking a new random number for the player to guess)
2) Change the text of the button back to "Check It!"
3) Change the behavior of the button so that it again will call checkGuess when pressed.

initializeGame already does all of these things. We just need to call it.

Once done, the new game is ready to be played.

2. Making it possible to start a new game

We said we need to change the button's text and behavior when the user's guess is correct. So, you need to add this line to your code:

```
button1.configure(text = "New game", command = newGame)
```

Where does this line go? *In the 'if' part of the if-elif-else of checkGuess.*

This code won't quite work yet because "newGame" has not been define.

What is commad "newGame"? It is simply a tiny function calls initializeGame with the existing maximum guessable number value stored in maxNumberToBeGuessed. Thus, we are not allowing the player to change the range of the values to be guessed. Each game will have the same range (until the player closes the game window and re-executes startGameGUI)

So, finally, add the newGame function:

```
def newGame():
    initializeGame(maxNumberToBeGuessed)
```

NOW TEST. You should be able to guess, get reasonable feedback, and start a new game after you win. AND, if there's still time, experiment a little – look at tkinter documentation and trying changing some colors or layout or preventing illegal guesses or …

Submission

Don’t forget to submit your final .py file as the DS10 assignment for your discussion section on ICON