# CS1210 Lecture 37 Nov. 17, 2021

- HW 8 due tomorrow
- Quiz 4 Friday in class
- HW 9 due Monday, Nov. 29, 8pm
- Two more HW and one more DS assignment during the final two weeks of class. My goal is for no one to want/need to take the final exam. *Work hard on the homework - that's where you'll learn the material - and for many people that will be enough for the grade they want*

## Last time

- Started GUIs

## Today

- *Discussion of an incorrect solution for HW8 Q1*
- More basic GUI examples
- Discussion of HW9 specification and demo of solution
- Comments on Quiz 4

# Links/resources for learning tkinter

- Chapter 15 of the interactive text has a LOT of info – more extensive that most of the other chapters.
- Tkinter info on the official Python site:
  https://docs.python.org/3.9/library/tkinter.html
- This tutorial - http://www.tutorialspoint.com/python/python_gui_programming.htm - does a good job of explaining and demonstrating the basics and includes several good small examples. It seems like a good place to start.
- If you want a "real textbook chapter", Chapter 6 of Kent Lee's book "Python Programming Fundamentals" is pretty good. The book is electronically available to UI students through the UI library (you need to be on the campus network to access it). http://link.springer.com/chapter/10.1007%2F978-1-84996-537-8_6
- This site can also very helpful. It's usually the first hit when I do Google search (though I'm not sure all info there is fully up-to-date): http://effbot.org/tkinterbook/tkinter-index.htm *(AS OF 11/15/21, THIS SITE IS NO LONGER WORKING - NOT SURE IF IT WILL COME BACK)*
- Some of the explanations here can be helpful (e.g. explains "pack" better than many others): http://thinkingtkinter.sourceforge.net

- tkinter is big – there are long chapters and even whole books about it. You will use Tkinter in DS10, and HW9, HW10, and HW11, but not a lot of the features.
- You just need to understand the basics of a few widget types (Label, Entry, Button, Frame) and how to respond to events like button presses.
- Make sure you understand these small examples
  - minimal.py covered last time
  - minimal2.py covered last time
  - simplegui1.py covered last time
  - simplegui2.py covered last time. Review today
  - simplegui2wClass.py quickly covered last time.  
    Covered more carefully today

## Friday

- Quiz 4. Probably
  - 1 simple problem on sorting and algorithm complexity:
    - Some very basic questions relating to algorithmic complexity (Big-O, etc.) of sorting algorithms we covered - selection, insertion, merge sort, quicksort. Binary search algorithmic efficiency vs. linear search. Difficulty/algorithmic complexity of other problems mentioned: shortest path in undirected graph (efficient! Bfs takes care of it in HW8) vs. longest path, traveling salesperson, …
  - 2 problems on graphs
    - Write down dictionary representation given a drawing/diagram of a graph (or the other way around)
    - Implement a function manipulating a simple graph representation (see next slide for a good example)
  - 1 function implementation problem involving basics that were covered in Quiz 1 and 2: loops, strings, lists, dictionaries.

b. Write function createComplementGraph that takes as input a dictionary representation of a graph (as in part a) and returns a new dictionary representing the complement graph of the input graph. For a graph G, the complement graph has the same vertices, and a set of exactly those edges *not* present in G. Remember that for a directed graph, a node can have an edge to itself.

For example, if G is

```
{"R" : ["G", "B"],
 "G" : ["G", "R", "B"],
 "B" : ["B"]}
```

```
createComplementGraph(G) might return:
```

```
{"R" : ["R"],
 "G" : [],
 "B" : ["R", "G"]}
```

Note: I say "might return" because the specific order of the nodes in the adjacency lists is not important. Note: you can do this problem without really knowing anything about graphs. It is really just a dictionary problem.

```
def createComplementGraph(g):
```