CS1210 Lecture 40 Dec. 1, 2021

- HW 10 due Saturday
- HW 11 will be posted tomorrow, due Sunday, Dec. 12

Today

- Final course grade scales are on the next slide
- One more sample widget, perhaps useful for HW10 and/or 11
- Continue with introduction to accessing web services such as Google Static Maps and Geocoding APIs
- Structure of HW10

Final Exam is optional. The default is that you are NOT taking it. You must OPT IN and notify me if you want to take it

Grade scales the are same percentage-wise except for rounding differences

Without final

Possible points: 168

Grade	Points	approx %
A+	163	97
A	148	88.1
A-	141	83.9
B+	133	79.2
B	122	72.6
B-	117	69.6
C+	109	64.9
C	92	54.8
C-	83	49.4
D+	80	47.6
D	73	43.5
D-	67	39.9

So far, maximum possible points: 156

HW: 8 x 6 = 48

DS: 10 x 3 = 30

Quizzes: 18 + 3 * 10 = 48

Maximum without final is: 168

HW9: 6 HW10: 5 HW11: up to 7 additional

DS11: up to 3 additional

I will provide a with-final scale in a few days. The percentages to achieve grade levels will be very nearly the same.

NOTE: **POINTS** ARE THE OFFICIAL SCALE – NOT PERCENTAGE

Structure of HW10start.py

- Classes
  - Globals. A class but we won't ever create instances. Just use the properties of the class as global variables. "Holding" them all in a Class object keep things a little bit better organized
    - properties include:
      - rootWindow
      - mapLabel
      - others you will add for new widgets you add to GUI
      - mapFileName, a string that never needs to change
      - mapSize, an integer that you can change if you want but doesn't need to change during execution of program
      - mapLocation, a string that you change via GUI
      - zoomLevel, an integer that you will change via GUI
      - other properties you will add such as mapType

Structure of HW10start.py

- Functions
  - geocodeAddress(addressString)
    - Calls Google geocoding service to get lat, lng for address specified in input string. Note: returns (0,0) if Google failed to yield lat/lng for the string
  - getMapUrl()
    - Uses values in Globals such as mapLocation, zoomLevel, etc. to construct a string that can be sent to Google to produce desired map.
  - retrieveMapFromGoogle()
    - Calls getMapUrl and then makes request to Google with that URL. Google returns an image that is stored in file Globals.mapFile
  - displayMap()
    - Calls retrieveMap… then then update GUI stuff to make map (which was stored in an image file by retrieveMap…) appear on Globals.mapLabel widget
  - readEntryAndDisplayMap()
    - *should* read string from Entry widget you add, save value in Globals.mapLocation, then display map. Initial version just displays map based on setting Globals.maplocation to "Beijing"
  - initializeGUIetc()
    - Creates GUI, sets values of Globals properties so we can access GUI widgets in other code.
  - HW10()
    - initializes GUI, displays map, starts Tkinter loop

geocodeAddress function: using the Geocoding API from Python

Input: string for a location. Return value: a latitude, longitude pair

1. Create a URL string describing the geocoding info you want.
   - first few lines of geocodeAddress
2. Send the URL to Google and receive the results
   - urlopen(…) line of geocodeAddress
3. Google returns a JSON-formatted string.
   - JSON is a commonly used open standard for transmitting data as text. http://en.wikipedia.org/wiki/JSON Perhaps most commonly, JSON is used to transmit data in dictionary form – i.e. before transmitting, data is encoded into a long string that "looks" (to human reader) like a dictionary representation. The receiver of this string can then "decode" the string into a dictionary data structure and extract the items of interest out of it.
   - Json.loads(…) line decodes the JSON result, yielding a Python dictionary
   - Remaining lines look in the dictionary to extract the info we need (here, latitude and longitude). Study the Google Geocoding API - https://developers.google.com/maps/documentation/geocoding/intro to see what information is in the dictionary (much more than you'll likely use!)

Similarly, using the Static Maps service from Python

1. Create a long URL string describing the map you want. Use the documentation - https://developers.google.com/maps/documentation/maps-static/intro to learn the URL format
   - see getMapUrl in hw10start.py
   - You will need to modify getMapURL for the pin and maptype parts of HW10
2. Send the URL to Google and receive the results
   - see retrieveMapFromGoogle

HW10 todo list

- understand the use of class Globals as “nice” way of handling global values
- add Entry widget so you can change location
  - Upon button press, callback should read Entry, set Globals.mapLocation property, and call displayMap()
- enable zooming
  - Upon use of widget (button or whatever you choose - consider +/-buttons as in simplegui2.py), set Globals.zoomLevel, call displayMap()
- enable changing map type
  - add a new Globals property for mapType
  - modify getMapURL to so that string representing request to Google Static maps API specifies desired map type
  - upon use of widget, set Globals.mapType value, and call displayMap() Consider radio buttons as in radioButton.py
- display pin at map center
  - modify getMapURL so that string representing request to Google Static maps API includes specification of pin at map center

- Demo: basic retrieving of Web pages via urlopen, etc. – getWebPage.py

Needed for HW11: DO **SOON**!

Definitely do step 1 before Monday

1. GET A TWITTER DEVELOPER ACCOUNT! It's **FREE**
   - https://developer.twitter.com/
   - If you are worried about us seeing your personal tweets, make a different Twitter account just for this class
   - It used to be simpler. Now you need to answer a bunch of questions to get the access we need. Just say you things like "For a university class project, learning about Twitter API and writing a Python program to search for tweets based on keywords and general location". *The answers DO MATTER* – some students have had their requests denied based at least partly on these answers I think.
   - For any required URLs (website, callback, etc.) when you are in the app creation screen, can just use http://www.uiowa.edu or similar
   - *Some email addresses* (maybe particularly some international ones) *seem problematic* – result in delayed or denied approval.
2. Add required keys to twitteraccess.py. Test that searchTwitter() in twitteraccess.py works for you.
3. Then start working on HW11…
   - More important than ever: do not write many lines of code before testing! *This assignment has a lot of code and many little things can go wrong*. If you add a lot of lines and then it crashes/doesn't work, it can be very difficult to debug/find where the error is.
   - Add a few lines, test, add a few lines, test, …

Next time

- Details and important information about HW 11
  - Twitter accounts
  - How to access/authenticate with web services like Twitter that require authorization