CS1210 Lecture 42 Dec. 6, 2021

- HW 11 due 8pm, Sunday, Dec. 12
- DS 11 is available now, due 5pm Thursday, Dec. 9. It is helpful practice for the "markers" part of HW11
- Once again: Course grading scales on next slide.

Today:

- HW11 and Twitter
- Some interesting CS theory

The rest of the week

- More on HW11
- Some interesting CS theory/advanced topics
  - Classification/intro machine learning, the Halting Problem, P vs NP, Turing machines, …

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

Without final max: 168

So far:

HW: 10 x 6 = 60

DS: 10 x 3 = 30

Quizzes: 18 + 3 * 20 = 78

HW11: 0-7 additional

DS11: 0-3 additional

depending on what HW &DS assignments you will drop

With final scale

Possible points: 200

Grade	Points	approx %
A+	194	97
A	176	88
A-	167	83.5
B+	158	79
B	145	72.5
B-	139	69.5
C+	130	65
C	110	55
C-	99	49.5
D+	95	47.5
D	87	43.5
D-	80	40

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

HW11 Full Assignment Todo list

1. Complete HW10. Do not do anything beyond this step until it is complete and working! Email me/TA when you can't figure things out!
2. Do the "enabling steps" – get Twitter account, keys, "install" Oauth related modules. Make sure twitteraccess.py functions work for you. (HW11 page provides a super *easy* way to install the necessary Oauth-related modules.)
3. add new Entry for search term (***don't*** add a new button for this! Original button should now read both entries, execute Twitter search, and retrieve and show map)
4. good practice: rename readEntryAndDisplayMap to, for example, readEntriesSearchTwitterAndDisplayMap
5. Before trying to actually display tweets on GUI and markers for tweets on map, test basic integration of twitter search. Upon button press, the readEntriesSearchTwitterAndDisplayMap callback function (command) should now read **both** Entry widgets, and *initially* just:
    - print tweet information the python shell
    - Maybe initially print whole tweet dictionary, but next extract just needed parts – text, name, screen_name and print those
    - Show the basic map, disregarding tweet information for now.
6. Once basic twitter integration has been checked, work on displaying tweet information in GUI and enabling "stepping through" tweets:
    - Save the retrieved tweets in a property of Globals. E.g. Globals.tweets
    - set a value to indicate which is the "current tweet". E.g. use another property such as Globals.currentTweetIndex. Set it to 0 when you first retrieve the tweets.
    - add widget(s) to GUI where tweet information can be displayed. Could be Label(s) but Text widget might be better (more on this later).
    - Implement a function displayTweet that updates relevant GUI widget(s) with information from 'current tweet"
    - Thus, readEntriesSearchTwitterAndDisplayMap should now read two Entries, call search Twitter, set some Globals properties, and call displayTweet *and* displayMap
7. Add GUI widget(s) to change "current tweet". If prior things are done well, this should be easy. Much like zoom buttons. Might have 'Next tweet', 'Prev tweet" buttons that simply change Globals.currentTweetIndex, and then call ???
8. Make sure to constrain tweet search by location. Pass the lat/lng of the specified location in your call to searchTwitter. Experiment with "radius" argument. Default radius is 2km. Large and very small radii don't seem to work well.
9. Add additional GUI widget(s) to display more tweet information, including the URLs within a tweet. This adds another layer of complication. Tweets can contain multiple URLs and you should be able to step through these as well. So, in addition to Globals.currentTweetIndex, you'll probably want another property such as Globals.currentTweetURLIndex, along with widgets to change this
10. Add code to allow opening of current tweet URL:
    - E.g. a button and callback/command function that simply executes webbrowser.open(relevant URL). Note: your code will need to import webbrowser module
11. Add markers (pins) for tweets - this is what DS11 helps you do.

Needed for HW11: DO VERY **SOON**!

1. GET A TWITTER DEVELOPER ACCOUNT! Free
   - First, get a Twitter account if you don't already have one. (if you are worried about us seeing your personal tweets, make a different Twitter account just for this class - but I promise we won't look!)
   - Follow the detailed steps at https://homepage.divms.uiowa.edu/~cremer/courses/cs1210/etc/getElevatedTwitterDevAccess.pdf to setup up a Twitter Developer account with "Elevated" access. It looks like a lot of little steps but it just takes a few minutes.
   - Then get the necessary API keys by following the steps at https://homepage.divms.uiowa.edu/~cremer/courses/cs1210/etc/accessKeysInstructions.pdf This also takes just a few minutes.

2. Add the keys you obtained at the end of step 1 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, …

Working with Twitter – twitteraccess.py

1. Twitter uses a very common authorization scheme called OAuth.
   - After you have a developer account, you need to create/register a Twitter app. Do that here: https://developer.twitter.com/en/apps
   - Once the app has been created, copy the four keys/tokens for the app and put them in the appropriate places in twitteraccess.py
     - API key, API secret, Access token, Access token secret
2. Before trying any other functions in twitteraccess.py, execute:
   - authTwitter() to enable access to the Twitter API by sending your authorization information
3. Try various functions in twitteraccess.py
   - Most importantly, learn to use the searchTwitter(…). It is key to retrieving Tweets for HW11
   - searchTwitter(…) constructs a string (like geocodeAddress and getMapURL but now for the Twitter Search API rather than Google APIs) that gets sent to Twitter.
   - Study the Search API docs here: https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets to understand the structure of the string sent to the Twitter Search API
   - The Twitter Search API returns a JSON object that contains as list of JSON Tweet objects as the value of its "statuses" key. The structure of a JSON Tweet object is detailed here: https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object

HW11: things to consider/keep in mind as you get to various parts

- *recommendation*: use Text widget rather than Label for displaying tweets.
  - in GUI initialization

    ```
    tweetText = tkinter.Text(width=…, height = …)
    tweetText.configure(state=tkinter.DISABLED)
    ```

  - in code to display tweet:

    ```
    tweetText.configure(state=tkinter.NORMAL)
    tweetText.delete(1.0, tkinter.END)
    tweetText.insert(…)
    tweetText.configure(state=tkinter.DISABLED)
    ```

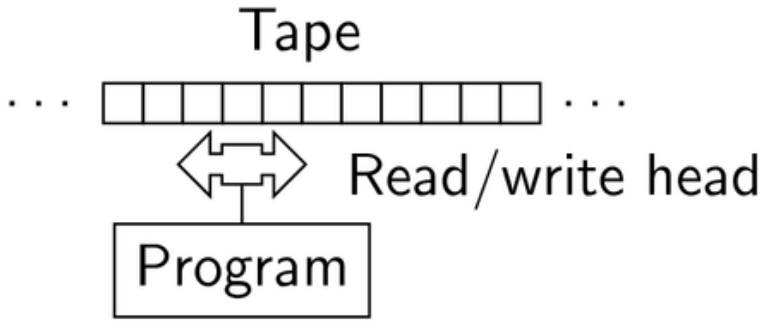
- Few tweets have non-null 'coordinates' field, so many of your markers will likely be in the middle of the maps. That's okay.
  - *recommendation*: to get more tweets *with* non-null coordinates field, it seems to help to use small search radius – e.g. 2km rather than 5 or 10.
  - Search in popular location of, say, large city. E.g. "Times Square" and "party"
- Handling Unicode characters in tweets:
  - Printing "raw" text of some tweets in Python shell will cause errors
  - see "printable" function in twitteraccess.py for "hacky" approach to eliminating unprintable characters

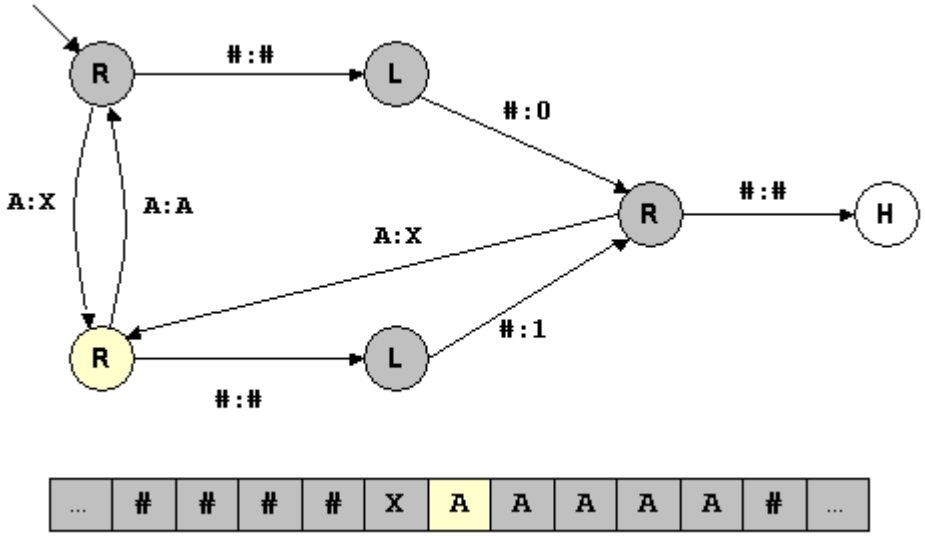
Some additional theory tidbits

- What if Python didn't have for/while loops, but had something else – e.g. "goto line i"
  - how does this affect what we can compute?
- Turing machines: important theoretical "universal computer"
  - memory: infinitely long "tape" of cells that can be blank or contain 0 or 1
  - program: a set of "states" and one fundamental operation:

    if state == q and tape cell == x, set cell to y, state to q', move

This simple computer can compute anything that is computable!





- A simulator
- A "real" one ☺

while vs. goto plus a simpler form of if statement that has no “body” – just a possible goto *linenum*

```
n = 0
while n < 100:
    sum = sum + n
    n = n + 1
print sum
```

```
1. n = 0
2. If ??? goto ???
3. sum = sum + n
4. n = n + 1
5. ???
6. print sum
```

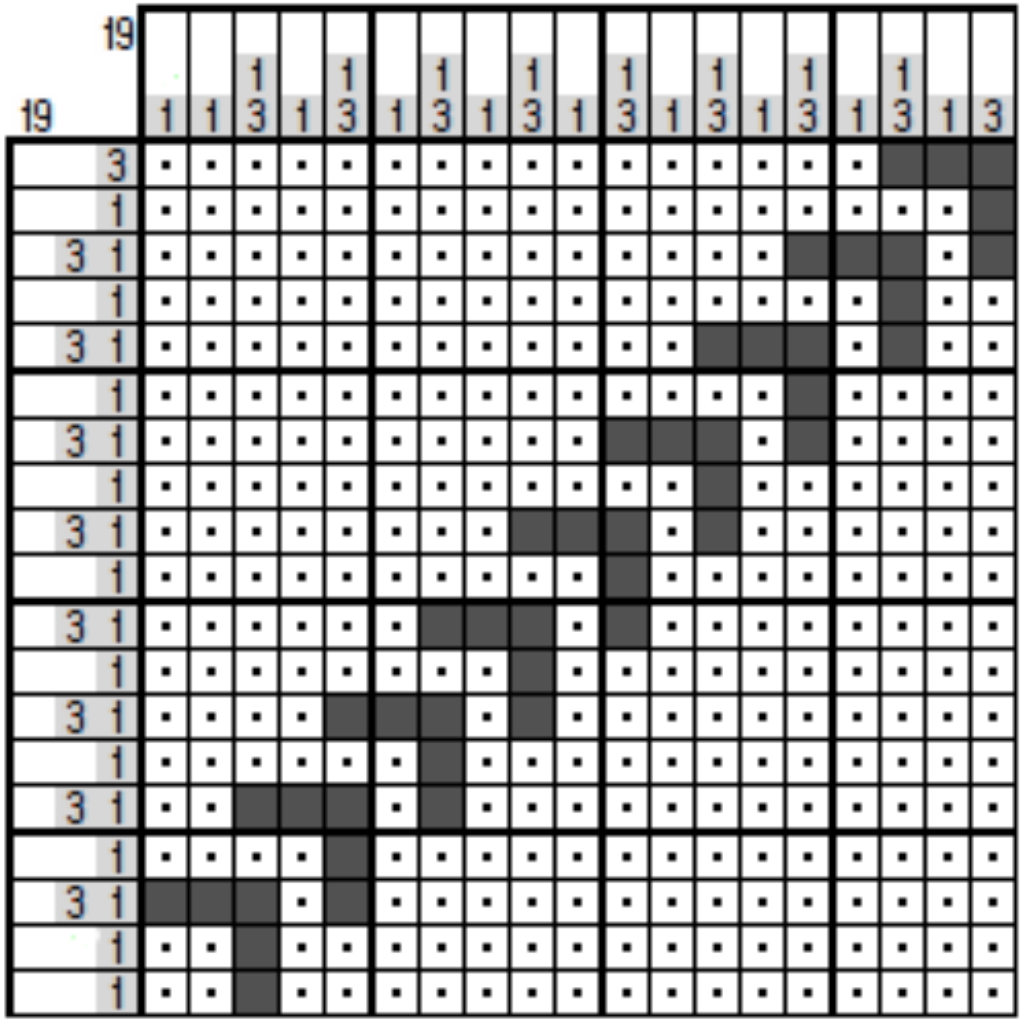
while vs. goto

```
n = 0
while n < 100:
    sum = sum + n
    n = n + 1
print sum
```

```
1. n = 0
2. if n >= 100 goto 6
3. sum = sum + n
4. n = n + 1
5. goto  2
6. print sum
```

More CS theory

- P vs. NP - the biggest unsolved problem in computer science
  - there are many problems that we don't know *efficient* algorithms for, and don't even know whether or not efficient ones even exist
  - Hundreds of real-world have been shown to be *NP-complete*
    - this means that if you solve any one of them, we can efficiently convert that solution into a solution for the rest of them. *Solve one efficiently* → *solve all efficiently!*
    - Longest path (but not shortest path), graph coloring, subset sum, many puzzles (Sudoku (!, ?, ?, solvers, hardest??, solver???, solver!, thoughts), minesweeper, etc.), satisfiability of logic formulas, nonograms (demo: nopuzzle.py)Traveling Salesperson (movie!?)
  - Solve it for extra practice this week == $1 million?
  - Demo: sudoku solver using "dancing links" algorithm sudoku.py



Human solvable via “global” thinking.
But not solvable by simple “local” thinking (single row/col constraints analysis) algorithm demonstrated In nopuzzle.py

https://webpbn.com/survey/dom.html

https://webpbn.com/survey/index.html

http://www.ijcsonline.com/IJCS/IJCS_2016_0303005.pdf

- Regarding puzzles:
  - Hundreds/thousands of logic puzzle types
    - Some cs (and other) research on puzzle design - lots of mobile games are puzzles – how are good puzzle levels made – via algorithm? by hand? Are hand-made ones better?
    - A recommended puzzle site (there are *many* good ones but this one provides an interesting variety of carefully hand-crafted ones, once per month, by CS PhD Pavel Curtis, who works at Microsoft)

      http://www.pavelspuzzles.com/aenigmas

The Halting Problem (later this week)

- it's important to know what we can and can't compute
- It turns out that we cannot create program that can check *all* other programs for infinite loops
- see, e.g., http://en.wikipedia.org/wiki/Halting_problem
- First: demonstrate that we can write programs that create and execute new programs/functions. testProgramOnInput.py
- Informal proof that we can't write doesItHalt
  - why can't we create fully correct doesItHalt function? (doesItHalt.py)
  - To see why, consider function test in doesItHaltTest.py

Next time

- Additional HW11 help
- Intro to some advanced CS topics
  - “baby” machine learning - classification