CS1 Lecture 20 Mar. 3, 2017

- HW 5, Q1 available. Q2 and 3 available Monday. Q1 is more than half the work. Start now.
- Complete this week's discussion section surveys!

Today

- Exercises from last time, plus a few variants
- HW 5 discussion/tips

Comment/question about simple password scheme from ppldata.py dictionary example from last time

```
>>> hashPassword("letsgo")
'147e81309435d1f60319f4775d96a6e272e29b2c'
>>> hashPassword("letsgo1")
'cc4e7a7d3b697ca492e35eb06a94f111da0074ba'
>>> hashPassword("letsga")
'8e190c4de1b45eca0543c8fcd2151233a52d7af7'
>>> hashPassword("password")
'5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8'
>>> hashPassword("1234")
'7110eda4d09e062aa5e4a390b0a572ac0d2c0220'
```

I said it's hard to go from stored hash value to plain text password. I.e. We don't know the math function that yields 'letsgo' from '147e81309435d1f60319f4775d96a6e272e29b2c'. But have we learned anything so far in this class that might help you 'crack' the passwords if you have access to hashPassword and the list of stored hashed passwords? If interested Google: "dictionary attack" (and partial remedy "password salt")

A few little exercises

1. Given a list of numbers, find the pair with greatest difference ✔
2. Given a list of numbers find the pair with smallest difference ✔
3. Given a list of numbers and a target number (call it k), find two numbers (if they exist) in the list that sum to k

lec20exercises.py

Small variants of/questions about third problem

4. Suppose:
   - No dictionaries allowed/available
   - Numbers in lists are know to have limited magnitude. E.g. all numbers between 0 and 10000

   Fast solution?
5. Modify findKPairFast to provide indices/location of found pair in list
6. Question: if we generate a list of, say, 10,000 random numbers between -1,000,000 and 1,000,000 how likely is it that the list contains a pair that sums to k for any k in, say, 0…999?

lec20exercises.py

HW5

It is not hard if you do a little bit at a time. Get it working bit by bit.

1. Read the file, storing all the messages and their labels (spam/ham). E.g.
   - [[‘ham’, ‘text me later!’], [‘ham’, ‘…’], …] and
   - [[‘spam’, ‘call 1412 to win’], [‘spam’, [‘…’]], …]
   - two separate lists, one for spam, one for ham
2. For each message, extract its words, and update spam or ham dictionary of word counts accordingly
   - for ‘text me later!’ increment ‘text’, ‘me’, ‘later’ entries in ham dictionary
3. Use the two dictionaries to compute and print some statistics
   - get total spam/ham word counts and unique word counts
   - extra most common words from dictionaries
   - print some stats (maybe eliminating tiny words, numbers, …? Experiment to see what produces interesting output)

HW5

1. File has some non-Ascii characters.

   use: open(fileName, encoding = 'utf-8')

2. To break line into tokens – individual elements of a line, learn how to use string **split**

   ```
   for line in fileStream:
       lineAsList = line.split()
   ```

   lec20split.py

3. get rid of extra stuff "…cool!?" learn how to use string **strip** (and/or lstrip, rstrip)

Next Time

- Chapter 12: tuples (and zip)
- A bit of Ch 19: conditional expressions and list comprehensions (will be topic of HW5 Q2 and Q3)