CS1210 Lecture 16 Sep. 29, 2021

- Quiz 1 scores available
  - Option to replace Quiz 1 score with Quiz 2 score (up to 18 points), essentially counting Quiz 2 twice
- Quiz 2, Wednesday, Oct. 6, in class
- HW4 and DS5 are available:
  - DS 5 due tonight, 8pm.
  - HW 4 due Tuesday, Oct. 5, 8pm

Last time

- Dictionaries - Ch 12
- Background example for DS 5 assignment

Today

- One more small dictionary example
- A few little exercises
- Overview of HW 4

(last time) Chapter 12: Dictionaries

- Python supports the extremely useful **dictionary** 'dict' type in Python
- Dictionaries are:
  - collections of key – value pairs
- Similar to but importantly different from lists
  - could think of lists as *ordered* collection of key-value pairs, where the keys are integers 0, 1, 2, …
  - with dictionaries, the collection is *unordered* but the interesting thing is that the *keys can be any immutable values*
  - *E.g.* create dictionary numlegs

    >>> numlegs = { 'frog': 4, 'human': 2, 'ant':6, 'dog':4}

(last time) Dictionaries

- create with { k1:v1, k2:v2, …}
- empty dictionary: {}
- retrieve value: dict[key]
- modify (or insert) value for key: dict[key]=value

- one important feature of dictionaries is that they provide *very fast* access (we might discuss *how* later in term) to values associated with keys despite being more flexible (not restricted to integer keys, etc.) than lists (demo: dicttest.py for speed comparison with lists)

Background examples for this discussion section assignment

- How would you implement printLetterCounts(inputString, letters) that prints the number of occurrences in inputString of each letter in letters? E.g

```
>>> printLetterCounts("This is a sentence containing a variety of letters",
                      "aeiouy")
'This is a sentence containing a variety of letters' has:
        4 'a's
        6 'e's
        5 'i's
        2 'o's
        0 'u's
        1 'y's
        and 32 other letter
```

list version: letterCountsWLists.py
In discussion section assignment
you will redo this with dictionaries

- birdDict.py example

A few little exercises

- Given a list of numbers, find the pair with greatest difference
- Given a list of numbers find the pair with smallest difference
- 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

lec16exercises.py has solutions for first two. Has slow (and not completely correct) solution for third one. Can you think of a much faster solution using dictionaries?

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?

lec16exercisesB.py

HW4

http://www.cs.uiowa.edu/~cremer/courses/hw/hw4/hw4.html

It is interesting, and 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.
   - Two separate lists: ham list [[‘text’, ‘me’, ‘later!’], [‘…’, …], ...] and spam list [[‘call’, ‘1412’, ‘to’, ‘win’], …] *(I recommend this option)*
   - Or one list [[‘spam’, [‘call’, ‘1412’, ‘to’, ‘win’]], [‘ham’, [‘text’, ‘me’, ‘later!’]], […], …]
   - Note: don’t keep ham/spam label/tag as part of message. I’ve seen people do this and then write special case code to ignore ‘ham’/’spam’ when processing message words in step 2 below – this can yield errors.
2. Create a ham and a spam dictionary. 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 dict
3. Use the two dictionaries to compute and print some statistics
   - get total spam/ham word counts and unique word counts
   - extract most common words from dictionaries
   - print stats

HW4

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()

   lec16split.py

3. get rid of extra stuff "…cool!?" learn how to use string **strip** (and/or lstrip, rstrip)
   - I *strongly recommend against* using replace() method
   - *Don't put "" (empty string) as a word in your dictionaries*

Next Time

- Tuples and tuple assignment
- default/optional and keyword arguments to function
- Zip and sorting for HW4