Chapter12_Exercises

June 3, 2017

1 Chapter 12 Exercises

1.1 Exercise 12.1

Write a function called most_frequent that takes a string and prints the letters in decreasing order of frequency. Find text samples from several different languages and see how letter frequency varies between languages. Compare your results with the tables at http: // en. wikipedia. org/ wiki/ Letter_ frequencies

```
In [27]: def most_frequent(stringy):
             l = list(stringy.lower())
             d = dict()
             #print(l)
             for c in l:
                 if c in d:
                     d[c] += 1
                 else:
                     d[c] = 1
             print(list(reversed(sorted(d, key=d.get))))
             #print(d)

         most_frequent('hiasdfasdf asdf jjjjjj')
```

```
['j', 'f', 'd', 's', 'a', ' ', 'i', 'h']
```

1.2 Exercise 12.2

More anagrams! 1. Write a program that reads a word list from a file (see Section 9.1) and prints all the sets of words that are anagrams. Here is an example of what the output might look like: [‘deltas’, ‘desalt’, ‘lasted’, ‘salted’, ‘slated’, ‘staled’][‘retainers’, ‘ternaries’] [‘generating’, ‘greatening’][‘resmelts’, ‘smelters’, ‘termless’] Hint: you might want to build a dictionary that maps from a collection of letters to a list of words that can be spelled with those letters. The question is, how can you represent the collection of letters in a way that can be used as a key? 2. Modify the previous program so that it prints the longest list of anagrams first, followed by the second longest, and so on. 3. In Scrabble a “bingo” is when you play all seven tiles in your rack, along with a letter on the board, to form an eight-letter word. What collection of 8 letters forms the most possible bingos? Hint: there are seven.

```
In [19]: fin = open('words.txt')

         d = dict()

         for line in fin:
             l = list(line.strip())
             l.sort()
             l = ''.join(l)

             if l not in d:
                 d[l] = [line.strip()]
             else:
                 d[l].append(line.strip())

         # print(d) uncomment me to see output
         t = []
         for key, v in d.items():
             t.append((len(v), v, key))

         t.sort(reverse=True)

         bingo = []

         for x in t:
             # print(x) uncomment me to see output *** Warning long loading time
             if len(x[2]) == 8:
                 bingo.append(x)

         print(bingo[0])

         fin.close()

         print('Finished')
```

```
(7, ['angriest', 'astringe', 'ganister', 'gantries', 'granites', 'ingrates', 'rangi
Finished
```

1.3 Exercise 12.3

Two words form a "metathesis pair" if you can transform one into the other by swapping two letters; for example, "converse" and "conserve". Write a program that finds all of the metathesis pairs in the dictionary. Hint: don't test all pairs of words, and don't test all possible swaps |

```
In [38]: def swappable(str1,str2):
             for i in range(0, len(str1)-1):
                 teststr = str1[0:i] + str1[i+1:i+2] + str1[i:i+1] + str1[i+2:]
                 if teststr == str2:
```

```
                return True
        return False

    for x in t:
        for i in range(len(x[1])-1):
            if swappable(x[1][i], x[1][i+1]):
                # print(x[1][i], x[1][i+1]) uncomment me to see output
                None
```

1.4 Exercise 12.4

Here's another Car Talk Puzzler (http: // www. cartalk. com/ content/ puzzlers): What is the longest English word, that remains a valid English word, as you remove its letters one at a time? Now, letters can be removed from either end, or the middle, but you can't rearrange any of the letters. Every time you drop a letter, you wind up with another English word. If you do that, you're eventually going to wind up with one letter and that too is going to be an English word—one that's found in the dictionary. I want to know what's the longest word and how many letters does it have? I'm going to give you a little modest example: Sprite. Ok? You start off with sprite, you take a letter off, one from the interior of the word, take the r away, and we're left with the word spite, then we take the e off the end, we're left with spit, we take the s off, we're left with pit, it, and I. Write a program to find all words that can be reduced in this way, and then find the longest one. This exercise is a little more challenging than most, so here are some suggestions: 1. You might want to write a function that takes a word and computes a list of all the words that can be formed by removing one letter. These are the "children" of the word. 2. Recursively, a word is reducible if any of its children are reducible. As a base case, you can consider the empty string reducible. 3. The wordlist I provided, words.txt, doesn't contain single letter words. So you might want to add "I", "a", and the empty string. 4. To improve the performance of your program, you might want to memoize the words that are known to be reducible.

```
In [14]: fininja = open('words.txt')
         dic = dict()

         for line in fininja:
             dic[line.strip()] = False

         dic['i'] = 'i'
         dic['a'] = 'a'

         def get_children(word):
             t = list(word)

             children = []

             for i in range(len(t)):
                 tmp = t[:]
                 tmp.pop(i)

                 if ''.join(tmp) in dic:
```

```
                    children.append(''.join(tmp))

            return children

        def istrue(child):
            #print(child)

            if len(child) == 1:
                return True

            answer = get_children(child)
            #print(answer)

            if answer:
                for subchild in answer:
                    #print(subchild)
                    return istrue(subchild)
            else:
                return False

        def find_all():
            for key in dic:
                childs = get_children(key)
                if childs:
                    for child in childs:
                        if istrue(child):
                            dic[key] = True
                            break
                    # now dic has {'word': True or False} for each word in words.t

        find_all()

        #print(istrue('sprite'))

        winners = []

        for key, val in dic.items():
            if val == True:
                #print(key)
                winners.append([len(key), key])

        winners.sort()

        print(winners[len(winners)-1])
```

```
[11, 'complecting']
```

In [ ]: