Exercise Sheet 2

Exercise 1 - Word list

For the exercises in this chapter we need a list of English words. We will use one of the word lists collected and contributed to the public domain by Grady Ward as part of the Moby lexicon project (see `http://wikipedia.org/wiki/Moby_Project`). It is a list of 113,809 official crosswords; that is, words that are considered valid in crossword puzzles and other word games. In the Moby collection, the filename is 113809of.fic; you can obtain a copy in this directory with the simpler name words.txt.

a) Write a program that reads words.txt and prints only the words with more than 20 characters (not counting whitespace).

b) In 1939 Ernest Vincent Wright published a 50,000 word novel called Gadsby that does not contain the letter "e". Since "e" is the most common letter in English, that's not easy to do. Write a function called has_no_e that returns True if the given word doesn't have the letter "e" in it. Write a program that reads words.txt and prints only the words that have no "e". Compute the percentage of words in the list that have no "e".

c) Write a function that reads the file words.txt and builds a list with one element per word. Write two versions of this function, one using the append method and the other using the idiom t = t + [x]. Which one takes longer to run? Why?

```python
import time

def make_word_list1():
    t = []
    fin = open('words.txt')
    for line in fin:
        word = line.strip()
        t.append(word)
    return t



def make_word_list1():
    """Reads lines from a file and builds a list using append."""
    t = []
    fin = open('words.txt')
    for line in fin:
        word = line.strip()
        t.append(word)
    return t


def make_word_list2():
    """Reads lines from a file and builds a list using list +."""
    t = []
    fin = open('words.txt')
    for line in fin:
        word = line.strip()
        t = t + [word]
    return t


start_time = time.time()
t = make_word_list1()
elapsed_time = time.time() - start_time

print(len(t))
print(t[:10])
print(elapsed_time, 'seconds')

start_time = time.time()
t = make_word_list2()
elapsed_time = time.time() - start_time

print(len(t))
print(t[:10])
print(elapsed_time, 'seconds')
```

Exercise 2 - Word list cont.

This exercise continues on from Exercise 2 and uses the same words.txt file

a) Write a program that prints all the sets of words that are anagrams (i.e. each word in the set uses all of the same set of letters, just rearranged).

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?

b) Modify the previous program so that it prints the longest list of anagrams first, followed by the second longest, and so on.

```python
"""This module contains a code example related to

Think Python, 2nd Edition
by Allen Downey
http://thinkpython2.com

Copyright 2015 Allen Downey

License: http://creativecommons.org/licenses/by/4.0/
"""

from __future__ import print_function, division


def signature(s):
    """Returns the signature of this string.

    Signature is a string that contains all of the letters in order.

    s: string
    """
    # TODO: rewrite using sorted()
    t = list(s)
    t.sort()
    t = ''.join(t)
    return t


def all_anagrams(filename):
    """Finds all anagrams in a list of words.

    filename: string filename of the word list

    Returns: a map from each word to a list of its anagrams.
    """
    d = {}
    for line in open(filename):
        word = line.strip().lower()
        t = signature(word)

        # TODO: rewrite using defaultdict
        if t not in d:
            d[t] = [word]
        else:
            d[t].append(word)
    return d


def print_anagram_sets(d):
    """Prints the anagram sets in d.

    d: map from words to list of their anagrams
    """
    for v in d.values():
        if len(v) > 1:
            print(len(v), v)


def print_anagram_sets_in_order(d):
    """Prints the anagram sets in d in decreasing order of size.

    d: map from words to list of their anagrams
    """
    # make a list of (length, word pairs)
    t = []
    for v in d.values():
        if len(v) > 1:
            t.append((len(v), v))

    # sort in ascending order of length
    t.sort()
    # print the sorted list
    for x in t:
        print(x)


if __name__ == '__main__':
    anagram_map = all_anagrams('words.txt')
    print_anagram_sets_in_order(anagram_map)

    eight_letters = filter_length(anagram_map, 8)
    print_anagram_sets_in_order(eight_letters)
```

Exercise 3 - Wallis formula continued

Again compute the decimals of Pi using the Wallis formula, this using list comprehensions, lambda functions and reduce

$$\pi = 2\prod_{i=1}^{\infty} \frac{4i^2}{4i^2 - 1}$$

```
"""
The correction for the calculation of pi using the Wallis formula.
"""
###############################################################################
# Solution in a single line using more advanced constructs (reduce, lambda,
# list comprehensions
print(2 * reduce(lambda x, y: x * y,
                 [float((4 * (i ** 2))) / ((4 * (i ** 2)) - 1)
                 for i in range(1, 100000)]))
```

Exercise 4 - Directory Listing

Implement a script that takes a directory name as argument, and returns the list of '.py' files, sorted by name length.

```
"""
Script to list all the '.py' files in a directory, in the order of file
name length.
"""

import os
import sys


def filter_and_sort(file_list):
    """ Out of a list of file names, returns only the ones ending by
        '.py', ordered with increasing file name length.
    """
    file_list = [filename for filename in file_list
                      if filename.endswith('.py')]

    def key(item):
        return len(item)

    file_list.sort(key=key)
    return file_list


if __name__ == '__main__':
    file_list = os.listdir(sys.argv[-1])
    sorted_file_list = filter_and_sort(file_list)
    print(sorted_file_list)
```