# Chapter14_Exercises

June 5, 2017

## 1 Chapter 14 Exercises

### 1.1 Exercise 14.1

Write a function called sed that takes as arguments a pattern string, a replacement string, and two filenames; it should read the first file and write the contents into the second file (creating it if necessary). If the pattern string appears anywhere in the file, it should be replaced with the replacement string. If an error occurs while opening, reading, writing or closing files, your program should catch the exception, print an error message, and exit.

```
In [1]: def sed(pttrn, replace, file1, file2):
            try:
                fin = open(file1, 'r')
            except:
                print('Error opening file one')
                return -1

            try:
                fout = open(file2, 'w')
            except:
                print('Error opening file two')
                return -1

            try:
                for line in fin:
                    if pttrn in line:
                        line = line.replace(pttrn, replace, len(line))

                    fout.write(line)
            except:
                print('Error writing to file two')
                return -1

            try:
                fin.close()
                fout.close()
            except:
```

```
            print('Error closing files')
            return -1

        return None
```

## 1.2 Exercise 14.2

If you download my solution to Exercise 12.2 from http: // thinkpython2. com/ code/ anagram_ sets. py , you'll see that it creates a dictionary that maps from a sorted string of letters to the list of words that can be spelled with those letters. For example, 'opst' maps to the list ['opts', 'post', 'pots', 'spot', 'stop', 'tops']. Write a module that imports anagram_sets and provides two new functions: store_anagrams should store the anagram dictionary in a "shelf"; read_anagrams should look up a word and return a list of its anagrams.

```
In [8]: import anagram_sets as anaset
        import shelve

        def store_anagrams(filename, d):
            """d is an anagram dictionary"""

            with shelve.open(filename, 'c') as db:
                for key, val in d.items():
                    db[key] = val

        def read_anagrams(filename, word):
            """db is a database file that maps from words to a list of anagrams"""
            try:
                with shelve.open(filename) as db:
                    return db[anaset.signature(word)]
            except:
                print('Error')
                return -1

        if __name__ == '__main__':
            wordict = anaset.all_anagrams('words.txt')
            # store_anagrams('anagrams', wordict)
            # ^^^ only run once, takes forever to actually finish so just use words
            print(read_anagrams('anagrams', 'apple'))
            print('Finished')
```

```
['appel', 'apple', 'pepla']
Finished
```

## 1.3 Exercise 14.3

In a large collection of MP3 files, there may be more than one copy of the same song, stored in different directories or with different file names. The goal of this exercise is to search for duplicates. 1. Write a program that searches a directory and all of its subdirectories, recursively, and returns a list of complete paths for all files with a given suffix (like .mp3). Hint: os.path provides several useful functions for manipulating file and path names. 2. To recognize duplicates, you can use md5sum to compute a "checksum" for each files. If two files have the same checksum, they probably have the same contents. 3. To double-check, you can use the Unix command diff.

```
In [23]: import os

         def find_suffix_files(dirname, suffix):
             checksums = dict()
             paths = []
             dups = []

             for root, dirs, files in os.walk(dirname):
                 for filename in files:
                     file = os.path.join(root, filename)
                     if file.endswith(suffix):
                         cmd = 'md5sum ' + file
                         fp = os.popen(cmd)
                         res = fp.read()
                         stat = fp.close()
                         #print(file)
                         #print(res)
                         #print(stat)
                         checksums[str(res)] = checksums.get(file, 0) + 1
                         if checksums[str(res)] == 1:
                             paths.append(file)
                         else:
                             dups.append(file)

             return dups

         find_suffix_files('C:\\Users\\Jim\\Music', '.mp3')

         # I'm not on unix when runninng this so it does not output properly
```

```
Out[23]: ['C:\\Users\\Jim\\Music\\test\\song1.mp3',
          'C:\\Users\\Jim\\Music\\test\\song2.mp3',
          'C:\\Users\\Jim\\Music\\test\\song3.mp3',
          'C:\\Users\\Jim\\Music\\test\\song4.mp3',
          'C:\\Users\\Jim\\Music\\test\\song5.mp3',
          'C:\\Users\\Jim\\Music\\test\\song6.mp3',
          'C:\\Users\\Jim\\Music\\test\\song7.mp3',
          'C:\\Users\\Jim\\Music\\test\\test2\\asdf.mp3',
          'C:\\Users\\Jim\\Music\\test\\test2\\asdf2.mp3']
```