# CIS 210, Fall 2016
# Introduction to Computer Science


## Main Menu

- Class home page
- Piazza
- How to Succeed
- Schedule
- Assignments
- References
- Exams


## Project 4, Part 1

This assignment is due Friday, October 21 at 5pm. Save your Python program in a file called jumbler.py and turn that file in using Canvas.

## Purpose

A common programming task is to scan a file, picking out elements that meet some criterion. This assignment introduces the file-scan idiom in Python, along with some basic manipulation of strings (text).

## Word Jumbles (Anagrams)

Anagrams are words whose letters have been scrambled. In newspapers and online, "word jumble" puzzles are based on anagrams, and can be solved using the program you will write. (For example, you should be able to solve the anagrams at the daily jumble.

### Requirements

Your program will read two strings from the command line. The first is a scrambled word (e.g., "hinttree"). The second is the name of a file containing a list of words, which will serve as a dictionary. Your program will check the scrambled word against every word in the dictionary. If the scrambled word can be rearranged to match a dictionary word, the dictionary word will be printed. After scanning the whole word list, the program prints the number of matches and the number of words in the list. For example:

```
$ python3 jumbler.py trsesi dict.txt
resist
sister
2 matches in 41238 words
$ python3 jumbler.py rayin dict.txt
rainy
1 match in 41238 words
$ python3 jumbler.py tororo dict.txt
No matches
$ python3 jumbler.py taroro dict.txt
orator
1 match in 41238 words
$
```

Note that 41238 is the number of words found in that particular dictionary. We could also run the program with a shorter or longer dictionary (by typing a different file name in place of dict.txt), and would see a different word count.

```
$ python3 jumbler.py phaal shortdict.txt
alpha
1 match in 5 words
```

## Notes and hints on the jumble solver

Please name your program file jumbler.py.

I've provided some starter code in jumbler.py. Note there are several parts you need to change. Some are flagged with FIXME, some are not.

### How to check

There is a simple trick for matching anagrams in a word list - we sort the letters in the anagram (e.g., "trscsi" becomes "cirsst"), and we also sort the letters in each dictionary word before checking. Then, we can simply check to see if they match exactly.

Note: when python reads a line of text from a file, the line includes the ending carriage return and/or newline character. That's why we strip "white space" off the ends of each entry in the dictionary file this way:

```
word = word.strip()  # Remove spaces or carriage return at ends
```

Python has a built-in function called sorted which puts any sequence into sorted order. A string in Python is an immutable sequence, so you can use the sorted function to get the letters in a word in alphabetical order. The sorted function always returns a *list*, so for example sorted("resist") will return ["e", "i", "r", "s", "s", "t"]. Since we can check equality between lists, this feature of sorted works fine. For example, if we check sorted("abc") == sorted("cba"), we will be comparing ["a", "b", "c"] to ["a", "b", "c"], and the result will be True.

You'll need a good word list to test your program. dict.txt will work. Put it into the same directory as your program for easy testing. Don't turn it in with your program.

You can find lots of test cases by searching for jumble or word jumble on the web.

## Grading

35 points possible

- 15: Program runs and consistently produces correct output for any scrambled word and any dictionary. 5 point penalty for any of the following:
- Test cases running in addition to program functionality. (You need to comment out the call to run_tests for your final testing and submission.)
- Incorrect or missing counts of matches or items in the word list.
- Other extra output, missing output, or output formatted differently than the examples above.

- 10: Follows CIS 210 coding guidelines, including author identification and header.
- 10: Clarity. The program shouldnot only be consistent with the requirements and approach described here, but it should be very easy to read the program and verify its consistency with the spec.