Scrabble Challenge

Part 1: Tile Pool

Create a class that will maintain a pool of tiles.

- It must be named **TilePool**.
- It must read in a tile set from the file **tiles.csv** when initialized[1].
- It must implement a **pop** method that takes an optional **tile_count** argument. If no argument is passed, the **pop** function should return 7 random tiles. Otherwise it should only accept numeric values between 1 and 7 inclusive[2], and return that many tiles. When a tile is popped it must be removed from the pool.
- It must implement the **len** operator which will be used to determine when the pool is empty.

Part 2: Word Finder

Create a class that when given 7 tile letters and a cross letter, will return a list of possible words.

- It must be named **WordFinder**.
- The **word_dictionary.txt** file must be read at runtime.
- The class must provide a method named **list_words**, which will return or yield a sequence of all words matching up to 7 tiles and a cross letter (watch out, this can be **None** to represent an empty board, or it could be invalid). The results should be sorted by score[3] in descending order (highest scoring words first). Each item should be a **tuple** of (*word*, *score*)[4].

1 All of the external files will be in the working directory.
2 What should happen if an invalid value for **tile_count** is passed to **pop**?
3 The score for each tile is in **tiles.csv**.
4 In this exercise cross tiles do not count towards the word's score.

Program Framework

Your module should run against the included **scrabble_test.py**, so pay attention to the naming of your module, classes, and methods. Use this code for testing, but be aware that the actual code used to test your module will be different.

Scoring

We will be timing your code and measuring its memory consumption so pay attention to efficiency, but the most important thing is readability. Here are some things we look for:

- PEP-8 code style
- self-documenting code
- Python 2/3 compatibility
- platform independence