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 an external file.
- It must implement a **pop** method that takes an option **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), 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 that 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**.
- A word dictionary will be provided as an external text file. The dictionary must remain an external file to allow using different dictionaries, so the word list cannot be hard-coded into your program.
- The class must provide a method named **list_words**, which will return a list 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). This list should be sorted by score in descending order (highest scoring words first). Each list item should be a tuple of (word, score). *Note: The score will be part of the tile pool's external csv file.*

Program Framework

Your module will run against the following code, so pay attention to the naming of your module, classes, and methods.

Use this code for testing, but be aware that the final code used to test your module will be slightly different.

```
import sys
import scrabble_challenge

cross_letters = [chr(x) for x in xrange(97, 123)]
cross_letters.insert(0, None) # the first word will not have a cross letter

def main():
    tile_pool = scrabble_challenge.TilePool()
    word_finder = scrabble_challenge.WordFinder()
    while len(tile_pool):
        letters = tile_pool.pop()
        for cross_letter in cross_letters:
            for word, score in word_finder.list_words(letters, cross_letter):
                print word, score

if __name__=='__main__':
    main()
    sys.exit(0)
```

Scoring

We're looking for efficiency, so we will be measuring running time and memory usage. We're also looking for pythonic code style, so keep it fairly readable.

Restrictions

The only restriction besides the naming conventions listed above is that the external files must remain external, meaning that you must read and structure their contents at runtime. The structuring of the data and module is completely up to you.