Assignment 1

Due January 29, 2017

Submissions are due by 11:59PM on the specified due date. Submissions may be made on the Blackboard course site under the Assignments tab. Late submissions will be accepted up to two days late with a 10% penalty for each day (24 hours).

Make sure your name and FSUID are in a module docstring at the top of the file.

You may use the standard library modules *random* and *time* as well as the PyEnchant library. No other modules or libraries are allowed and significant penalties will be applied if any other modules or libraries are used. Please keep in mind that for this and all assignments, the *__future__* module is always allowed.

1 Boggle (100 points)

Peggy Hill needs your help in preparing for the big Boggle tournament coming up! Your task is to create a Boggle simulator. If you don't know the rules to Boggle, start by reading up on them here: https://en.wikipedia.org/wiki/Boggle.

The Boggle simulator will serve as a tool to help players practice their Boggle skills – that is, it is intended to be used by a single player. The game has the following phases:

1. The 16 dice (described on the next page) are "rolled" and randomly positioned on a 4x4 grid.

2. The player begins to type the words that appear in the grid. The player must search for words that can be constructed from the letters of sequentially adjacent cubes, where "adjacent" cubes are those horizontally, vertically, and diagonally neighboring. For a word to be scored, it must meet the following requirements:
   - The word must not have already been scored.
   - It must be at least three letters long.
   - It must be a word in the English language.
   - It must be present in the 4x4 grid of dice.
   - It may not use the same letter cube more than once per word.

These conditions should be checked in this order. For example, the word "OG" should be rejected for being too short, not for being absent from the dictionary. The user entries should not be case-sensitive. The player must be prompted to end this phase by entering the letter 'X'.

3. When the player elects to quit, the player will be presented with their final score. The player may only receive the points for the word if it meets the qualifications above. The program should print out how many points the player received for each word and if the word was not included, the program must print the reason. Finally, the program should print the total score. Note that we do not enforce a timer as is done in traditional Boggle.

Words are awarded points according to the table below (as long as they meet the conditions listed above):

Word Length	Points
3, 4	1
5	2
6	3
7	5
8+	11

The 16 dice have the following letters on each of their six sides:

1. A E A N E G
2. A H S P C O
3. A S P F F K
4. O B J O A B
5. I O T M U C
6. R Y V D E L
7. L R E I X D
8. E I U N E S
9. W N G E E H
10. L N H N R Z
11. T S T I Y D
12. O W T O A T
13. E R T T Y L
14. T O E S S I
15. T E R W H V
16. N U I H M Qu

Note that Qu is the only two-letter sequence that appears on a single face of a die.

To verify that a word is actually present in the English language, you are encouraged to use the PyEnchant module. The PyEnchant module may be installed on Ubuntu 14.04 with the following command:

```
$ sudo apt-get install python-enchant
```

and an example usage of the dictionary lookup functionality is shown below:

```
>>> import enchant
>>> d = enchant.Dict("en_US")
>>> d.check("Hello")
True
>>> d.check("sdjh")
False
```

A few more suggestions:

- Consider a recursive approach to verifying that the word can be found within the dice grid. Start with finding all of the possible positions where the word may start, then recursively check each valid position for the next letter, backtracking when you determine that the word cannot be found from that point.
- The majority of your implementation will likely rely on lists and strings. Make use of list comprehensions for brevity and use built-in string methods to make it easier on yourself.
- Come see me if you are struggling!

Some sample runs follow:

```
$ python boggle.py
[O] [I] [S] [E]

[L] [R] [O] [N]

[T] [K] [N] [I]

[Y] [N] [J] [I]

Start typing your words! (press enter after each word and enter
'X' when done):
> NORSE
> RISE
> SON
> IN
> SORT
> NINE
> KIN
> NOR
> EONS
> KON
> X
The word NORSE is worth 2 points.
The word RISE is worth 1 point.
The word SON is worth 1 point.
The word IN is too short.
The word SORT is worth 1 point.
The word NINE is worth 1 point.
The word KIN is not present.
The word NOR is worth 1 point.
The word EONS is worth 1 point.
The word KON is ... not a word.
Your total score is 8 points!
```

```
$ python boggle.py
[S] [E] [T] [T]

[O] [A] [N] [T]

[M] [O] [V] [E]

[T] [H] [E] [E]

Start typing your words! (press enter after each word and enter
'X' when done):
> TEN
> OVEN
> MOVE
> MOVED
> TEAM
> TEA
> AT
> MATE
> NAOS
> TEA
> X
The word TEN is worth 1 point.
The word OVEN is worth 1 point.
The word MOVE is worth 1 point.
The word MOVED is not present.
The word TEAM is worth 1 point.
The word TEA is worth 1 point.
The word AT is too short.
The word MATE is worth 1 point.
The word NAOS is ... not a word.
The word TEA has already been used.
Your total score is 6 points!

$ python boggle.py
[I] [N] [D] [E]

[E] [C] [A] [E]

[I] [S] [O] [T]

[D] [T] [M] [A]
```

```
Start typing your words! (press enter after each word and enter
'X' when done):
> NICEST
> ATOM
> TOAST
> X
The word NICEST is worth 3 points.
The word ATOM is worth 1 point.
The word TOAST is worth 2 points.
Your total score is 6 points!
```