CSCI 141, Fall 2018
Hangman final project

Due by 2359 on Wednesday, December 12, 2018

1 Rules of the game

Your hangman board should look something like this:

```
-----
|   |
|   O
|  /|\
|  / \
|
|

l o a t h s o m
- - - - - - - - -

Guess a letter: / You have already guessed 'c'. / Another game? (y/n): ]
```

The object of the game is to guess all the letters in a hidden word. The player guesses at the missing letters.

1. If they guess a letter that has already been guess, they are prompted to try again.
2. If then enter something other than a single letter, they are prompted to try again.
3. If they guess a letter that appears in the word, then all instances of the letter in the word are revealed. This will require updating and redrawing the board.
4. If they guess a letter that does not appear in the word, then another part of the dangling figure is drawn. This will require updating and redrawing the board.

When all six parts of the figure are drawn, the game ends in a loss for the player and the entire word is exposed for their edification. If the entire word has been guessed, the game ends in a win for the player.

2 Implementation details

Implement hangman as a `Hangman` class. The class constructor `__init__()` should include arguments that allow the user to specify

1. `level`, the difficulty of the game, in terms of the minimum number of letters in the words,
2. `non_ascii`, which is `True` if non-ASCII characters are allowed, and

3. `dictionary`, which specifies a default dictionary file to read.

All of these should have default values assigned to them.

For instance, in my implementation, `__init__()` starts like this:

```
class Hangman(object):
    def __init__(self, level=5, non_ascii=False,
                 dictionary='/usr/share/dict/american'):
```

This sets the default difficulty to words of length five and longer, causes words with non-ASCII characters to be ignored, and sets the default dictionary to a value appropriate for the machines in the CS lab. You are free to add other options for your game (e.g., you might allow proper nouns or contractions). **Be sure to comment what options your game has in your code.**

The presence of the default values makes it easy to create a default instance of hangman:

```
hangman = Hangman()  # Use the default values.
hangman.play()  # Loop until the player no longer wants to play.
```

Besides `__init__()`, the only other required method is `play()`. This is the method the player will call to play the game. Once a game is completed, `play()` should include a prompt asking whether the player wants to play the game again. If so, a new game is started; otherwise `play()` exits.

You are otherwise free to include any methods you need. For this assignment you may ignore information hiding and make all your methods and class variables publicly accessible.

3 Exemplar

I've placed a working (?) version of my code on the CS system. This version will only run on the CS systems. You can see what it looks like with the following incantations:

```
goofy% cd /home/scratch/rmlteach/python
goofy% ../anaconda3/bin/python
Python 3.7.0 (default, Jun 28 2018, 13:15:42)
[GCC 7.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from hangman import *
>>> h = Hangman()
>>> h.play()

-----
|   |
|
|
|
|
|
|
```

```
- - - - -

Guess a letter: a
Correct!
-----
|   |
|
|
|
|
|
|

    a
- - - - -

Guess a letter:
```

4 What to submit

Turn in your code as `hangman.py` to the Blackboard site.