Hangman

Hangman is an old game that originated in the late 17th to early 18th centuries (lookup "Rite of words and life"). It was the basis for the creation of the `Wheel of Fortune` game show in 1973. The rules are simple:

- The player tries to guess a word one letter at a time
- If the guess is correct, the letter is revealed in it's position(s) in the word
- If the guess is incorrect, the number of allowed gueses is reduced
- The game ends with five incorrect guesses (a loss) or correctly revealing all of the letters in the word (a win)

We're going to write Hangman as a computer program.

Decomposition

With any problem, a good approach is to break it down into the essential steps of the solution. This is called `decomposition`. By identifying the steps in the solution, the most important steps can then be focused on, with less important steps saved for last. For really big problems each of these steps may be decomposed, again, and again.

What are the steps required for playing Hangman?

First pass, our game engine

The essentials of our game are to:

- pick a secret word
- let the user guess a letter
- determine if the guess is correct or not
- repeat until the game is over

To simplify this, let's use `orange` for our secret word and focus on the other steps. Also, we will only support lower-case.

Don't worry about revealing the position of the letters yet, just print `Correct` or `Incorrect` after each guess, and `Win!` or `Lost :(` when the game is over.

Some helpful tips:

You can get a list of letters from a string using the `list()` cast:

```
letters = list('word')
print(letters)
```

You can check to see if a letter (or string) is in an array by using the `in` operator:

```
letters = list('barage')
if 'a' in letters:
    print('there is an a')
```

Second pass, fixing some test cases

One important test case is if the secret word has any letters duplicated. Try changing your secret word from `orange` to `apple`. Can you still finish the game (both win and lose) ?

If not, how would you fix this?

Here's another useful hint. You can find out how many times a letter (or string) occurs in a list by using the `count()` function:

```
apple = list('apple')
how_many_p = apple.count('p')
print(how_many_p)
```

Another thing that can be useful is when you want to loop through a definite loop a particular number of times. You can use a `for-loop` for this, but in Python you need a collection of some kind to loop over. The `range()` function can give you a list of consecutive numbers for this purpose:

```
# Print numbers from 0 to 9
for i in range(10):
    print(i)

# Print numbers from 5 to 9
for i in range(5, 10):
    print(i)
```

Third pass, making the game more usable

We have a functional game, but one of the primary features of Hangman is knowing where the correct letters go.

Instead of just printing `Correct` on a good guess, print the word with the correct letters and `_` for the unknown letters. For example, for `apple`, after guessing `e` you should print:

`_ _ _ _ e`

While we're making things helpful, on an incorrect guess it would be nice to tell the user how many guesses they have left.

If you haven't been using functions yet, your program is now getting big enough that they would be very useful. Think about using a function to calculate the display string, for example.

Last pass, fit and finish

Finally, using a single secret word isn't very exciting. Think of how you can write a function to pick a new word each game.

You can get random numbers using the `random` module, but you have to import it first:

```python
import random

# Picks a number between 0 and 10
small_number = random.randrange(10)

# Picks a number between 200 and 300
medium_number = random.randrange(200, 300)
```