The Python Hangman Game

Now with Python, we cannot create physical on screen drawings, so this version of Hangman will simply count down how many lives we have left.

In our case we’re also going to let Python randomly select a word from a list of words we will have already coded into the game. For this we need a list of strings, like this:

words = [‘chicken’, ‘dog’, ‘cat’, ‘mouse’, ‘frog’]

What we’re going to do is follow this step by step guide, first making the various parts of the games functions, then sticking them all together to be able to play the game!

Getting a Random Word

To start a game, we will need a random word from our string that the player will have to guess. Create a new Python code window in IDLE 3, and try this code out, be sure to save it!

```
#04_04_hangman_words
import random

words = ['chicken', 'dog', 'cat', 'mouse', 'frog']
lives_remaining = 14
guessed_letters = ‘’

def pick_a_word():
      word_position = random.randint(0, len(words) - 1)
      return words[word_position]

print(pick_a_word())
```

When you run this, you’ll just get a print out of a random word from your String, feel free to change your string with different values if you wish!

Making a Play Function

What we now need is a function; this is where we will make the game physically “play” so to speak.

Go back to your `#04_04_hangman_words` file, and then “Save As” that as “**python_game.py**”. We’re now going to start from our previous code we wrote to make up the rest of the game.

Remove the following line of code, as we needed this only for testing:

```
print(pick_a_word())
```

Now we'll need the following:

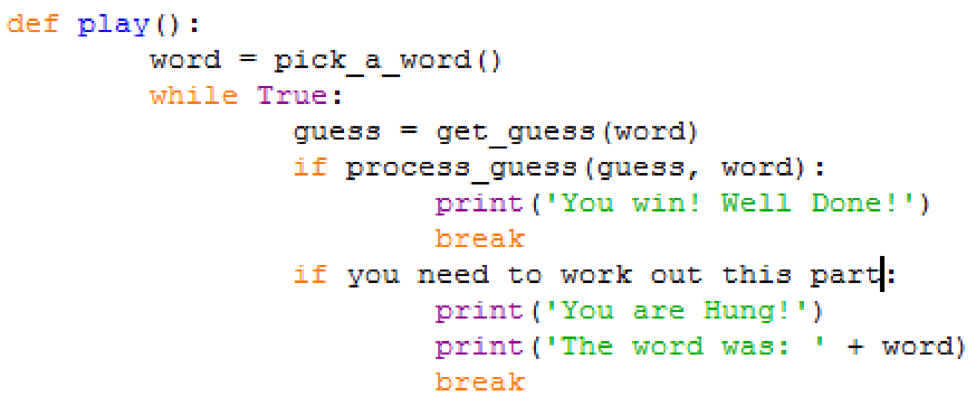
```
def play():
        word = pick_a_word()
        while True:
                guess = get_guess(word)
                if process_guess(guess, word):
                        print('You win! Well Done!')
                        break
                if ==you'll need to work out this bit:
                        print('You are Hung!')
                        print('The word was: ' + word)
                        break
```

Ok, so now you've got this code, there's an If statement incomplete.

Here's a hint, the IF statement is there to tell the user they have no lives left, you'll need to use the **`lives_remaining`** variable and one of the following operators:

- == Equals
- != not equals
- > greater than
- < less than
- >= greater than or equal to
- <= less than or equal to

Note the indentation you use is important; it should look like this when in IDLE:



```
def play():
        word = pick_a_word()
        while True:
                guess = get_guess(word)
                if process_guess(guess, word):
                        print('You win! Well Done!')
                        break
                if you need to work out this part:
                        print('You are Hung!')
                        print('The word was: ' + word)
                        break
```

We cannot run this code we have at the moment because the functions **`get_guess`** and **`process_guess`** don't exist yet.

Firstly, add this code to your game:

```
def pick_a_word():
	word_position = random.randint(0, len(words) - 1)
	return words[word_position]
```

This picks a word from your list.

Getting a Guess

We need a way to tell the player how they're doing, and what the word looks like. For this we'll create a function, as follows:

```
def get_guess(word):
	print_word_with_blanks(word)
	print('Lives Remaining: ' + str(lives_remaining))
	guess = input(' Guess a letter or whole word?')
	return guess
```

What we are doing here is printing the status of the players guessing efforts, from our "`print_word`" function we'll be adding later.

Then we print the amount of lives remaining to the player using print(). Finally we are returning guess, which is what the user entered.

Printing the Word

We’re going to keep the player updated after each guess of how their guessed word is looking, and as to what letters they’ve guessed. So this is where we need our “`print_word`” function. To display and update this, we’ve created a String variable at the beginning of the guide:

```
guessed_letters = ‘’
```

By using this String, we can store the word in its current state every time the player makes a guess.

You’ll need all the code:

```
def print_word_with_blanks(word):
	display_word = ''
	for letter in word:
		if guessed_letters.find(letter) > -1:
			# letter found
			display_word = display_word + letter
		else:
			# letter not found
			display_word = display_word + '-'
	print(display_word)
```

This is a loop that compares the letter the player entered with every letter of the randomly selected word.

It adds a hyphen (-) if the letter is guessed wrong, otherwise it will provide the position of the letter in our String.

Processing the Guess

When a player guesses, they will either be entering a single letter, or the whole word. Therefore we need to use an IF in the `process_guess` function, to tell the game which results to return after the user enters their guess:

```
def process_guess(guess, word):
	if len(guess) > 1 and len(guess) == len(word):
		return whole_word_guess(guess, word)
	else:
		return single_letter_guess(guess, word)
```

Here, if the guess length is more than 1, then we’ll return “`whole_word_guess`”, otherwise it’ll return the “`single_word_guess`”, these are explained on the next page.

Adding the Single Letter and Whole Word functions

These three functions are important, as they decide what the game will do after a player makes a guess. We return true if the guess was correct, and false if it was incorrect.

Copy this code snippet to your program:

```
def whole_word_guess(guess, word):
      global lives_remaining
      if guess.lower() == word.lower():
            return
            #the word guessed is right, what should be returned?
      else:
            lives_remaining - 1
            #the above is also wrong, can you see why and fix it?
            #this is to do with the syntax of adding to a variable
            return False

def single_letter_guess(guess, word):
      global guessed_letters
      global lives_remaining
      if word.find(guess) == -1:
            # letter guess was incorrect
            lives_remaining = lives_remaining - 0
            #the above line of code is wrong, can you see where?
      guessed_letters = guessed_letters + guess.lower()
      if all_letters_guessed(word):
            return True
      return False

def all_letters_guessed(word):
      for letter in word:
            if guessed_letters.find(letter.lower()) == -1:
                  return False
      return True

play()
```

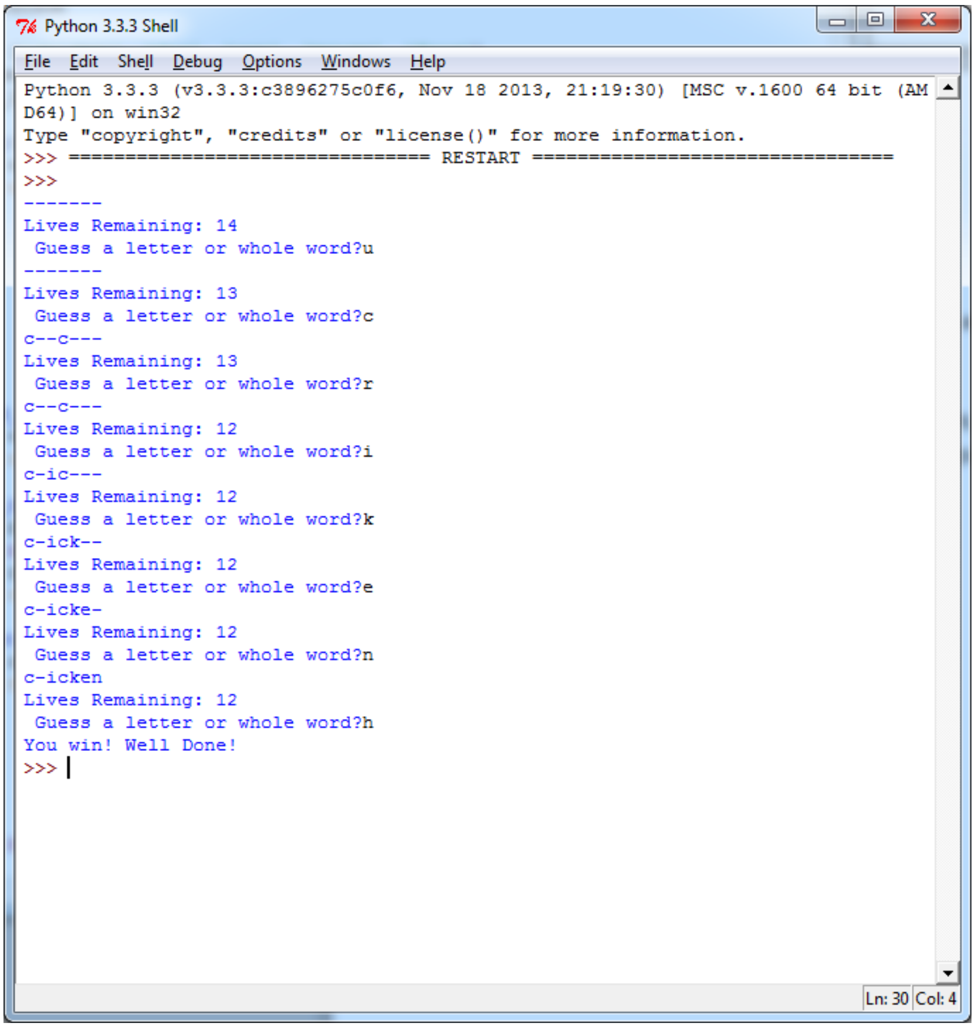
Here's a test, I've broken this snippet, as I've commented above where you need to try and work out and fix them.

If you're stuck, give me a shout.

Playing the Game

You should now be able to run this without any errors. If there are errors, check your syntax, and then ask for the solution print out!

There are some limitations during game play, but you can type either a letter in lower or upper case and it will still guess them and process the guesses correctly.



```
Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 18 2013, 21:19:30) [MSC v.1600 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>> 
-------
Lives Remaining: 14
 Guess a letter or whole word?u
-------
Lives Remaining: 13
 Guess a letter or whole word?c
c--c---
Lives Remaining: 13
 Guess a letter or whole word?r
c--c---
Lives Remaining: 12
 Guess a letter or whole word?i
c-ic---
Lives Remaining: 12
 Guess a letter or whole word?k
c-ick--
Lives Remaining: 12
 Guess a letter or whole word?e
c-icke-
Lives Remaining: 12
 Guess a letter or whole word?n
c-icken
Lives Remaining: 12
 Guess a letter or whole word?h
You win! Well Done!
>>> 
```