Beginning Python: Lesson 5

In this lesson you will learn about **Strings**. In Python, a string is a chunk of text between two single quotes `'` or double quotes `"`. For example:
- 'There was a dog called Eddie'
- "abracadabra"
There are two parts to this lessons: a joke program and a word jumble program.

Jokes

In this part of the lesson you will create a joke telling program. If you know any good jokes then add them to the program.

1. Create a new program, `jokes.py`

- Open Wing IDE on your computer. Click on the *File* menu and select *New*. A new blank window will appear for us to type our program in.
- Enter the following comments in the editor window and save the file as `jokes.py` in the "python-programs" folder you created in the last lesson.

```
# Jokes
# YOUR_NAME 7/3/2015
```

2. Add the jokes

Add the following print statements to your file:

```
print('What do you get when you cross a snowman with a vampire?')
input()
print('Frostbite!')
print()
print('What do you call an alligator in a vest?')
input()
print('An investigator!')
print('')
print("Why did the cow cross the road?")
input()
print('To get to the \'udder\' side!')
input()
```

3. Run the program

Now run the program. The program will stop at every `input()` and wait for you read the joke, then press Enter, and read the punch line. Update the program with your own jokes and let your friends play the game!

Escape Characters

The last print() above has a backslash (\ is a backslash) right before the single quote. This tells the computer that the letter right after it is an escape character. An escape character helps us print out letters that are hard to enter into the source code. T

Word Jumble

In this part of the lesson we'll create a word jumble game in Python. The game goes like this: the computer chooses a random word and mixes all the letters up. The player then tries to guess what the word is. Each time the player enters a guess, the computer tells the player whether the guess is correct or incorrect.

Planning the game

It's a good idea to plan your programs before you write them. Here is a plan for the word jumble game:

```
welcome the player to the game and explain it
create a list of words
choose a random word from the list of words
create an empty jumble word
while the chosen word has letters in it
    take a random letter from the chosen word
    add the random letter to the jumble word
show the jumble word to the player
while the player has not guessed the word
    ask the player to guess the word
    if the player if incorrect
        Tell the player to try again
Congratulate the player on guessing correctly
```

Now we will code each step in this algorithm

1. Create a new program, `word-jumble.py`

All good programs begin with a block of comments. The comment block says what the program does and who wrote it. The `#` at the start of each line tells Python that this is a comment.
- Open Wing IDE on your computer. Click on the *File* menu and select *New*. A new blank window will appear for us to type our program in.
- Enter the following comments in the editor window and save the file as `word-jumble.py` in the "python-programs" folder you created in the last lesson.

```
# Word Jumble game
# YOUR_NAME 7/3/2015
```

2. Import the Random module

The program needs to generate a random number. Import the `random` module like you did

last week. Add the following code to the program

```
import random
```

3. Welcome the player and explain the game

We want to tell the player how the game works. Add the following highlighted lines of code to your program to do this.

```
print("Welcome to Word Jumble! Unscramble the letters to make a
word...")
```

4. Tuples

Tuples are a sequence of things. You can have a tuple that stores a bunch of high scores for a game, or one that stores a group of player names. Create a tuple that contains the list of words we will use in the game.

```
WORDS = ("python", "jumble", "easy", "difficult", "answer",
"xylophone")
```

Next, use a new function, random.choice(), to grab a random word from WORDS:

```
word = random.choice(WORDS)
```

The `choice(WORDS)` function looks at the sequence you give and picks a random word. Once the computer has chosen a random word, it assigns it to the variable `word`. This is the word the player will have to guess. Lastly, assign word to correct, which we'll use later to see if the player makes a correct guess:

```
correct = word
```

5. Create an empty jumble string

The program creates the empty string and assigns it to jumble, which will refer to the final, jumbled word.

```
jumble =""
```

6. Create the jumbled word

This is the hard bit! The jumble creation process is controlled by a while loop. Enter the following code.

```
while word:
    position = random.randrange(len(word))
    jumble = jumble + word[position]
    word = word[:position] + word[(position + 1):]
```

The loop will continue until word is equal to the empty string. Each time the loop executes, the computer creates a new version of word with one letter “extracted” and assigns it back to word. Eventually, word will become the empty string and the jumbling will be done.

7. Get the players guess

Next, the computer gets the player’s guess. The computer keep asking the player for a guess as long as the player doesn’t enter the correct word or presses the ‘Enter’ key at the prompt:

```
guess = raw_input("\nYour guess: ")
while (guess != correct) and (guess != ""):
    print "Sorry, that's not it."
    guess = input("Your guess: ")
```

8. Congratulate the Player

At this stage the player has guessed correctly or decided to leave the game. Enter the following code:

```
if guess == correct:
    print "That's it! You guessed it!\n"
```

9. Ending the Game

Finally, the program thanks the player for playing the game and ends:

```
print "Thanks for playing."
```

10. Challenge

If you've finished this program, you can try the following challenges:

- Modify the Jumble game so that the player has 2 guesses. If the player fails to guess, the program should display the message "Sorry, you lose!".
- Modify the program to give the player a hint if they don't guess the word the first time.

> Written with StackEdit.