Back to the Word Jumble Game

The Word Jumble game combines several new ideas you learned about in this chapter. You can easily modify the program to contain your own list of words to guess.

Setting Up the Program

After my initial comments, I import the `random` module:

```
# Word Jumble
#
# The computer picks a random word and then "jumbles" it
# The player has to guess the original word
#
# Michael Dawson - 1/28/03

import random
```

Next, I used a tuple to create a sequence of words. Notice that the variable name `WORD` is in all caps, implying that I'll treat it as a constant.

```
# create a sequence of words to choose from
WORDS = ("python", "jumble", "easy", "difficult", "answer", "xylophone")
```

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

```
# pick one word randomly from the sequence
word = random.choice(WORDS)
```

This function is new to you, but it's pretty simple. The computer looks at whatever sequence you give and picks a random element.

Once the computer has chosen a random word, it assigns it to `word`. This is the word the player will have to guess. Lastly, I assign `word` to `correct`, which I'll use later to see if the player makes a correct guess:

```
# create a variable to use later to see if the guess is correct
correct = word
```

Planning the Jumble Creation Section

The next section of code uses the new concepts in the chapter and is the most interesting part of the program. It's the section that actually creates the jumbled word from the original, randomly chosen word.

But, before I wrote any code, I planned out this part of the program in pseudocode (yes, I actually use all that stuff I write about). Here's my first pass at the algorithm to create a jumbled word from the chosen word:

create an empty jumble word
while the chosen word has letters in it
  extract a random letter from the chosen word
  add the random letter to the jumble word

Conceptually, this is pretty good, but I have to watch my semantics. Because strings are immutable, I can't actually "extract a random letter" from the string the user entered. But, I can create a new string that doesn't contain the randomly chosen letter. And while I can't "add the random letter" to the jumble word string either, I can create a new string by concatenating the current jumble word with the "extracted" letter.

Creating an Empty Jumble String

The very first part of the algorithm is easy:

```
# create a jumbled version of the word
jumble =""
```

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

Setting Up the Loop

The jumble creation process is controlled by a `while` loop. The loop condition is pretty simple, as you can see:

```
while word:
```

I set the loop up this way so that it will continue until `word` is equal to the empty string. This is perfect, because 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.

Generating a Random Position in `word`

The first line in the loop body generates a random position in `word`, based on its length:

```
    position = random.randrange(len(word))
```

So, the letter `word[position]` is the letter that is going to be "extracted" from `word` and "added to" `jumble`.

Creating a New Version of `jumble`

The next line in the loop creates a new version of the string `jumble`. It becomes equal to its old self, plus the letter `word[position]`.

```
    jumble += word[position]
```

Creating a New Version of `word`

The next line in the loop,

```
    word = word[:position] + word[(position + 1):]
```

creates a new version of `word` minus the one letter at position `position`. Using slicing, the computer creates two new strings from `word`. The first slice, `word[:position]`, is every letter up to, but not including, `word[position]`. The next slice, `word[(position + 1):]`, is every letter after `word[position]`. These two string are joined together and assigned to `word`, which is now equal to its old self, minus the one letter `word[position]`.

Welcoming the Player

After the jumbled word has been created, the next section of the program welcomes the player to the game and displays the jumbled word to be rearranged:

```
# start the game
print \
"""
            Welcome to Word Jumble!

   Unscramble the letters to make a word.
(Press the enter key at the prompt to quit.)
"""
print "The jumble is:", jumble
```

Getting the Player's Guess

Next, the computer gets the player's guess. The computer keeps 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: ")
guess = guess.lower()
while (guess != correct) and (guess != ""):
    print "Sorry, that's not it."
    guess = raw_input("Your guess: ")
    guess = guess.lower()
```

I made sure to convert `guess` to lowercase since the word the player is trying to guess is in lowercase.

Congratulating the Player

At this point in the program, the player has either correctly guessed the word or quit the game. If the player has guessed the word, then the computer offers its hearty congratulations:

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

Ending the Game

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

```
print "Thanks for playing."

raw_input("\n\nPress the enter key to exit.")
```