Curiouser and Curiouser

With our "pig latin" program we saw that we can use Python to break a sentence into words and mess with the letters and letter-order in words. This time we're going to do some more work with words, but we are going to work with a whole book full of words.

The book I've chosen in Lewis Carroll's "Alice In Wonderland". This book is out of copyright and so legal electronic copies of it are readily available (eg, from www.gutenberg.org).

When you use a word processor to modify something you've written before, the first thing you have to do is to *open* the file or document. We do the same thing in Python. We are not going to modify the file, so we do not have to save it afterwards; when we have finished with it, we have to *close* it.

Let's get started. As usual, start by opening the python shell (IDLE) and use `file->new file` and then, in that new window, `file->save as`. Navigate to the code club drive and set the name of your program. Mine is called neal_alice. Use your own name, so that everyone's program has a different name.

Type in this piece of code. You don't need to type in the comments (lines that start with "#"). Be careful to get the indentation (spaces at the start of lines) correct.

```
# alice

file = open('alice.txt')
lines = file.readlines()
file.close()

count = 0
for line in lines:
    count = count + 1

print("There are " + str(count) + " lines in the file")
```

The program starts by opening a file, which contains the complete text of Alice In Wonderland. It reads the whole text in, line by line, into a *list* named `lines`, then it closes the file again. Now we can simply work with the list `lines`.

The next thing that the program does is to loop through each of the parts of the list (called *elements*, remember?) and count them.

Finally, the program prints out the total number of lines in the book. How long do you think it will take to do this? Run the program and see (click `run->run module` (or press F5)). If you get an error when you try to run it, compare the code above carefully with what you typed. Correct any mistakes and try again.

After you have run the program successfully, that list named `lines` is still there. Go into the Python shell and print some elements of it – don't type the ">>>" that is Python prompting you to type something in:

```
>>> lines[37]
>>> lines[3511]
>>> lines[3512]
```

That last one should have given you an error. You should have seen that there are 3512 lines in the book, so lines[0] is the first and lines[3511] is the last. lines[3512] does not exist (hence the error).

Next, remember how we broke a sentence into separate words (in igpay atinlay). We're going to do the same thing here and then count the number of words. At the same time we will count the number of lines (as we already did) and count the number of times that the name Alice occurs in the book.

Modify your program to look like the one below. You don't need to type it all in; just the parts that have changed. The new code is shown in red.

When you type the code in, your code will not be coloured like this!

```
# alice

file = open('alice.txt')
lines = file.readlines()
file.close()

line_count = 0
word_count = 0
alice_count = 0
for line in lines:
    line_count = line_count + 1
    words = line.split()
    for word in words:
        word_count = word_count + 1
        if word.lower() == "alice":
            alice_count = alice_count + 1

print("There are " + str(line_count) + " lines in the file")
print("There are " + str(word_count) + " words in the file")
print("There are " + str(alice_count) + " occurrences of the word alice in the
file")
```

Save the modified program and run it. Can you see how it works? It is running the same loop as last time, but now it's doing more inside the loop. For every line it's breaking the line into words (using `.split()`) and then running a second loop to count the words. Also, for each word it is doing a check to see if the word is "Alice". The program is doing a lot more than the last one. Did you notice that it took a lot longer to run than the last one? No? – it still seemed to run instantly. We need to make more work for the computer!

Next, instead of just counting how many times the word Alice occurs, we are going to make a count of every different word in the whole book. To do this, we're going to use a new sort of variable, called a *dictionary.*

Remember that the variables we've used so far have had a name (the name on the box) and a value (what's in the box) like this:

```
>>> fred = 6
>>> name = "sarah"
```

This new kind of dictionary variable works slightly differently. We can do this:

```
>>> fred["bread"] = 2
>>> fred["cheese"] = 5
>>> fred["butter"] = 99
```

Now we say that the *name* of the variable is fred, and that it holds *keys* ("bread", "cheese", "butter" in the example above) and each *key* has a *value* (2, 5, 99 in the example above). Because each key has a value, we sometimes call them *key/value pairs*. Let's look at how these can be useful.

Modify your program to look like the one below. You don't need to type it all in; just the parts that have changed. The new code is shown in red.

```
# alice

file = open('alice.txt')
lines = file.readlines()
file.close()

# Put EVERY word in this book into its own "bucket" and count the number of
# times that it occurred.

# For this we use a thing that Python calls a "dictionary"
wordlist = {}

for line in lines:
    words = line.split()
    for word in words:
        if word in wordlist:
            wordlist[word] = wordlist[word] + 1
        else:
            wordlist[word] = 1

keys = wordlist.keys()
for key in keys:
    print(key + " occurs " + str(wordlist[key]) + " times")
```

As before, click `run->run module` (or press F5). If it runs correctly, you will see lots of text scroll off the screen! Hold down the CTRL key (bottom left) and press C. This interrupts the program (stops it) and gets us back to the >>> prompt (Python will probably print something in red to tell you that it has been interrupted).

In this version of the program we are creating a *dictionary* named `wordlist`. We are going to use the words in the book as keys, and use the value to hold a count of the number of times that the word occurs. We loop over every word in every line, just as we did before.

For each word, we do this test: do we already have the word in our wordlist? If the answer is no, we add it to our dictionary and set its count to 1. If the answer is yes, we

add 1 to the count that we already have for the word. If you are unclear about any of this, ask for help.

Even though we interrupted our program, our `wordlist` still exists. Go to the command prompt and think of some words that might exist in the book:

```
>>> wordlist["alice"]
>>> wordlist["Alice"]
>>> wordlist["Queen"]
>>> wordlist["The"]
>>> wordlist["the"]
```

If the key (for example, the word "Alice") exists, Python will print the value (the number of times it occurred in the book). If the key does not exist, Python will print an error. Python has all of the words of the whole book stored in that variable named `wordlist`.

There are a few problems, though. For example, when I looked at the list of words I saw output like this:

```
Duchess, occurs 8 times
Duchess. occurs 3 times
Duchess) occurs 1 times
Duchess' occurs 1 times
Duchess! occurs 3 times
```

The program has treated all of these words as different because of the different punctuation at the end of the word. There's another problem, too: we have separate counts for `The` (starting a sentence) and `the` (part-way through a sentence). The final problem is that the list of words is being printed in any old order. It would be neater to print them alphabetically. Let's fix those three problems.

Modify your program to look like the one below. You don't need to type it all in; just the parts that have changed. The new code is shown in red.

```
# alice

import string

file = open('alice.txt')
lines = file.readlines()
file.close()

# Put EVERY word in this book into its own "bucket" and count the number of
# times that it occurred.

# For this we use a thing that Python calls a "dictionary"
wordlist = {}

for line in lines:
    # fix problem 1: change everything to lower case
    line = line.lower()

    # fix problem 2: get rid of all the punctuation
    for c in string.punctuation:
        line=line.replace(c," ")

    words = line.split()
    for word in words:
        if word in wordlist:
            wordlist[word] = wordlist[word] + 1
        else:
            wordlist[word] = 1

keys = wordlist.keys()

# fix problem 3: sort the keys before printing them.
for key in sorted(keys):
    print(key + " occurs " + str(wordlist[key]) + " times")
```

As before, click `run->run module` (or press F5). If it runs correctly, you will see lots of text printed out. Again you will need to press CTRL and C to stop the output. Now try these at the prompt:

```
>>> wordlist["alice"]
>>> wordlist["Alice"]
>>> wordlist["Queen"]
>>> wordlist["The"]
>>> wordlist["the"]
>>> wordlist["duchess"]
```

Now there should be no `Alice` or `Queen` or `The` (but there are `alice` and `queen` and `the`).

Look at the program and see how the three problems were fixed. We've seen the fix for problem 1 before.

If you are unclear about any of this, ask for help. Otherwise, read on for some extras.

Extras

For each of these, write a few new lines of code after the original program. If you get stuck on any of them, ask for help.

1. Print the 10 words that occur most often in the book.
2. What word is used most often; and how many times is it used
3. How many different words are used in the book?
4. Print a list of the words that are only used once in the book
5. What is the name of Alice's cat (hint: loop on each line and print any line that contains the word "cat")
6. What is the longest word in the book (hint: if `foo = "hello"` then `len(foo)` is 5. Loop over all the words and keep track of the longest so far. When you get to the end, the longest so far will be the longest.)
7. Change the program to read in a different book "les_miserables.txt". This is an (English) translation of a very long book by French writer Victor Hugo. You may have heard of the London stage show of the same name.
8. What are the 10 words that occur most often in *this* book. Can you tell which is a book for adults and which for children, based on this top 10?
9. Super bonus point: list the 10 longest words in each of these two books. Can you tell which is a book for adults and which for children, based on *this* top 10?