Secure Scripting (Python)

**The Basics**

# lab: Looking for strings

In this lab, you will build a simple tool to look up words containing a sequence of characters (a *string*). You will build this script in three stages. First, assume that both the string and the filename are given. Next, assume that the string is to be read from the command line. Finally, create a script that will read both the string and the filename from the command line. Check for errors throughout the entire process.

For this lab, you will need the following files:

* dict.txt

For reference, please use the following file:

* mytype.py

### Lab Exercise 1

Find the words in the word list “dict.txt” that contain the string “gry”.

1. What is the format of the file “dict.txt”? How many words per line does it contain?

*The file “dict.txt” contains a list of words, one word per line.*

1. Look up the program *grep*. What arguments would you give it to look for the string “gry” in dict.txt?

*The command is “grep gry dict.txt”, which prints each line containing the string “gry”.*

1. Write a script called “lookfor1.py” that executes this command to look for words containing “gry” in dict.txt. The first line should specify that the program “usr/bin/python” is to be used. Try it out. If you did it right, you will see eight words, the first being “agrypnia” and the last being “pouggry”.

*Key… lookfor1.py*

### Lab Exercise 2

Find the words in the word list “dict.txt” that contain a string supplied by the user. Hint: Begin with your script from Lab Exercise 1 and modify it as indicated.

1. How do you represent the first argument from the command line in the command you put into your script?

*sys.argv[0] this is the file of the Python script*

*sys.argv[1] this is the first actual argument to process from file argv[0]*

1. Modify the script you wrote for Lab Exercise 1 to take the string to be searched for from the command line. Test your script by searching for the strings “hello” and “world”. Call this script “lookfor2.py”.

*Key… lookfor2.py*

1. What happens if no arguments are given? Two arguments?

*If no arguments are given, you get an error. If multiple arguments are given, the first is treated as the string and the rest as filenames.*

1. How would you embed a space in a single argument?

You would surround the argument by single quotes (double quotes also work), so to look for the string “hello world” you would type:

lookfor2aa.py “hello world”

1. What happens if an argument contains a blank?

If the argument contains a blank, it will normally be read as two separate arguments. The problem is that the full argument, with the embedded blank, is passed to the script as a single argument. When the substitution for $1 occurs, it looks like two words, not one.

### Lab Exercise 3

Modify the script you wrote for Lab Exercise 2 by adding an “if” statement that checks whether there is exactly one argument. Call this script “lookfor3.py”.

If there is *not* exactly one argument (**in python considers not exactly 2, since the first argument is the actual file name**), your script should print the error message “Usage: give exactly 1 argument, the string to be looked for” and exit immediately.

*Key… lookfor3.py*

### Lab Exercise 4

Find the words in the word list named by the user (like “dict.txt”) that contain a string supplied by the user. Your shell script is to be called “lookfor4.py”.

Hint: Begin with your script from Lab Exercise 2 and modify it as indicated.

1. Modify your script in Exercise 2 so that the word list is the second argument on the command line. Call this script “lookfor4.py”. Remember to change your “if” statement so it balks if there are not exactly two arguments!

*Key… lookfor4.py*

1. Change the error message to “Usage: lookfor4 string file”. Call this script “lookfor4a.py”.

*Key… lookfor4a.py*

1. Change the error message so it prints the exact name of the script. That is, if you call the script “find4” and not “lookfor4.py”, the error message should print as “Usage: find4 string file”. A similar error message should occur if you call the script “catdog” *without changing the script!* Call this script “lookfor4b.py”.

*Key… lookfor4b.py*

### Puzzler

In the script you wrote for Lab Exercise 4, if the file does not exist, *grep* prints an error message. This will be confusing to beginners. Add a test at the start of that script that prints the error message

*script\_name*: file *file\_name* cannot be read

where *script\_name* is the name of the script and *file\_name* is the name of the file that the user gives. Call this script “lookforp1.py”.

*Key… lookforp1.py*