New York University
School of Continuing and Professional Studies
Division of Programs in Information Technology

Introduction to Python
Homework, Session 8

8.1 Directory Grep. grep is a Unix utility that does a text search of a file or files. It prints each line of a group of files that contains the search text. This program will do the same work as grep. Accept two arguments: the name of an existing directory, and a search string. Loop through each plaintext file in the directory, printing every line of each file that contains the search string, preceded by the filename and the line number where the string was found (see output).

Hint: remember that in can be used to find one string within another string.

Exceptions: trap exceptions for missing arguments and for unreadable directory.

Expected output (this is based on the shakespeare/ directory unzipped from the shakespeare.zip file found in the source data page on the class website. In the example below, the (line 1635) number is the line number of the 1_king_henry_vi.txt file.

```
$ python myprog.py ../python_data/shakespeare/ perforce

1_king_henry_vi.txt (line 1635):  PLANTAGENET   How I am braved and must perforce endure it!

2_king_henry_iv.txt (line 410):     To stormy passion, must perforce decay.

2_king_henry_iv.txt (line 909):     And one against Glendower; perforce a third

2_king_henry_iv.txt (line 2317):    And these unseason'd hours perforce must add

2_king_henry_iv.txt (line 2957):    Was force perforce compell'd to banish him:

... continues for 38 more lines ...
```

8.2 A watched directory. Accept one argument: the pathname of an existing directory. Create a program that stays running and continually watches the directory to see if any files have been added or removed. The program first lists and stores the files found in the directory at program start, and subsequently "polls" (re-checks) the directory files to see if any changes have been made. The checking can be done every 5 seconds or so.

To pause the program so it can wait before checking the directory again, use the following code:

```
import time   # at top of script with other imports
time.sleep(5)
```

To test, keep your program running, and then manually go into the directory and create and remove files to see the program's output reflect the changes. (Note that this can be a bit tricky in timing. Make sure to both add and remove a file to see that both are occurring.)

Hint: your program will run for an indefinite period, looping endlessly and checking the contents of the directory continuously. Use a while(True) loop to do this. The loop will not stop looping until the program is terminated using Ctrl-C.

Exceptions: trap exceptions for missing argument and unreadable directory.

Expected output (will of course vary based on directory and add/remove events):

```
$ python watchdir.py my_files_dir/
# time passes as the program polls (periodically reads) the directory
# at some point, we add two files to the directory (you can do this manually)

file1.txt added
file2.txt added

# time passes as the program continues polling the directory
# at some point, we remove one of the files


file2.txt removed

# time passes... we add two more files


file3.txt added
file4.txt added

# time passes... we remove one of the files

file3.txt removed
```

SUPPLEMENTARY (EXTRA CREDIT) EXERCISES

8.3 Largest files in a directory using os.walk. Accept two arguments: the name of an existing directory, and the number of files to display. Use os.walk to go through an entire directory tree (review slide explaining this function). Store the names of a directory's files (the directory is submitted as an argument on the command line) paired with each file's size. Print the number of files requested, sorted by size, largest to smallest. Print out the filenames along with their paths.

Exceptions: trap exceptions for missing arguments and if the second argument is not an integer. (os.walk does not raise an error if the directory is invalid; choose your own method of noting this error.)

Expected output (will of course vary based on arguments):

```
$ python largestfiles.py '/Users/david/python' 5
9339  homework_4_plaintext
8366  homework_4_outline.txt
6148  .DS_Store
564  ff_highestval.py
513  watched.py
```