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

Introduction to Python
Homework Discussion, Session 8

8.1 Directory Grep. This shouldn't be harder than looping through the directory and listing out the files, and then inside the directory loop, for each file in that loop, opening each file and looping through it line-by-line (thus, a for loop (for lines of each file) inside a for loop (for each file)). As you loop through each line of each file, use the in operator to check the search string against the line -- i.e., to see if the search string appears in the line.

Since each printed entry includes the filename and the line number where the term was found, you'll set a line count integer to 0 just before the loop through the file begins, and then increment the line counter inside the file loop.

If your program can't seem to find the file with the filename, don't forget that you have to os.path.join() the directory name to the filename so that Python can find it.

8.2 A watched directory. The purpose of the assignment is to reinforce file lists and use sets to notice differences. Since set.difference() can tell us what's in one set that isn't in the other, it would be pretty simple to be able to say whether a file was added or removed by comparing the contents of the directory, listed out at different times.

The overall approach contemplated here is to list the files in the directory and store them in a set; wait 5 seconds; then list the files into a set again, and compare the sets. You'll do oldset.difference(newset) to detect file deletions and newset.difference(oldset) to detect file additions. Putting this process into a while(True) loop allows this process to go on indefinitely.

Since the while(True) loop has no natural way to break out, the program requires the use of Ctrl-C to exit.

The logic of the while(True) loop can be done in different order, perhaps, but here is the logic outline that worked for me:

```
# test sys.argv to affirm that dir argument has been passed

# list files into set ("old" set)

# start while True: loop

    # sleep 5 seconds
    # list files into set ("new" set)
    # compare sets to see what files have been added
    # compare sets to see what files have been removed
    # report any added or removed files

    # assign "new" set to "old" set variable for next loop comparison
```

8.3 Largest files in a directory using os.walk. This script will be pretty close to the os.walk example in the slides. You simply need to join the directory to the filename to get the filepath, and then use os.path.getsize to get the file's size. Initialize an empty dictionary before the loop begins, store the filepath as the key and size as the value in the dictionary, and after the loop ends, sort the dictionary by value using key=dictname.get (where dictname is the name of your dictionary). Limit the results by using a slice subscript which can be placed directly after sorted().