Master of Engineering in Internetworking

Lab # 5

Data Structure Selections and Files,

INWK 6312

SECTION A

deterministic: Pertaining to a program that does the same thing each time it runs, given the same inputs.

pseudorandom: Pertaining to a sequence of numbers that appear to be random, but are generated by a deterministic program.

default value: The value given to an optional parameter if no argument is provided.
override: To replace a default value with an argument.

benchmarking: The process of choosing between data structures by implementing alternatives and testing them on a sample of the possible inputs.

Go to Project Gutenberg (`http://gutenberg.org`) and download your favorite out-of-copyright book in plain text format.
Download these books or any three books:
The New England Country by Clifton Johnson,
Woodbarrow Farm by Jerome K. Jerome,
A Book for a Rainy Day by John Thomas Smith

Question 1

Write a program that reads a file, breaks each line into words, strips whitespace and punctuation from the words, and converts them to lowercase.

Hint: The `string` *module provides strings named* `whitespace`*, which contains space, tab, newline, etc.,* `punctuation` *which contains the punctuation characters and* `printable` *which contains printable characters.*

```
>>> import string
>>> print string.punctuation
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

Also, you might consider using the string methods `strip` *and* `replace`

Question 2
Modify your program from the previous exercise to read the book you downloaded, , and process the rest of the words as before.

Then modify the program to count the total number of words in the book, and the number of times each word is used.

Print the number of different words used in the book. Compare different books by different authors, written in different eras. Which author uses the most extensive vocabulary?

Bonus: write a function that takes in any number of books and produce the book with the most number of words

Question 3
Modify the program from the previous exercise to print the 20 most frequently-used words in the book.

Question 4

- *Modify the previous program to read a word list and then print all the words in the book that are not in the word list. How many of them are typos? How many of them are common words that* should *be in the word list, and how many of them are really obscure?*

- *Python provides a data structure called* `set` *that provides many common set operations. Read the documentation and write a program that uses set subtraction to find words in the book that are not in the word list.*

Question 5

The “rank” of a word is its position in a list of words sorted by frequency: the most common word has rank 1, the second most common has rank 2, etc.

Zipf’s law describes a relationship between the ranks and frequencies of words in natural languages (`http://en.wikipedia.org/wiki/Zipf's_law`). Specifically, it predicts that the frequency, f, of the word with rank r is:

$f = c\, r^{-s}$

where s and c are parameters that depend on the language and the text. If you take the logarithm of both sides of this equation, you get:

$\log f = \log c - s \log r$

So if you plot log f versus log r, you should get a straight line with slope −s and intercept log c.

Write a program that reads a text from a file, counts word frequencies, and prints one line for each word, in descending order of frequency, with log f and log r. Use the graphing program of your choice to plot the results and check whether they form a straight line. Can you estimate the value of s?

To make the plots, you might have to install matplotlib (see`http://matplotlib.sourceforge.net/`).

To plot using matplotlib:

```
import matplotlib.pyplot as pyplot
    pyplot.clf()
    pyplot.xscale(scale)
    pyplot.yscale(scale)
    pyplot.title(“title string)
    pyplot.xlabel(“label string”)
    pyplot.ylabel(“label, string”)
    pyplot.plot(x-value, y-value)
    pyplot.show()
```

SECTION B

persistent: Pertaining to a program that runs indefinitely and keeps at least some of its data in permanent storage.

format operator: An operator, `%`, that takes a format string and a tuple and generates a string that includes the elements of the tuple formatted as specified by the format string.
format string: A string, used with the format operator, that contains format sequences.

format sequence: A sequence of characters in a format string, like `%d`, that specifies how a value should be formatted.

text file: A sequence of characters stored in permanent storage like a hard drive.

directory: A named collection of files, also called a folder.

path: A string that identifies a file.

relative path: A path that starts from the current directory.

absolute path: A path that starts from the topmost directory in the file system.

catch: To prevent an exception from terminating a program using the `try` and `except` statements.

database: A file whose contents are organized like a dictionary with keys that correspond to values.

`os.`**`listdir`**(*path*)

Return a list containing the names of the entries in the directory given by *path*. The list is in arbitrary order. It does not include the special entries `'.'` and `'..'` even if they are present in the directory.

`os.path.`**`join`**(*path*, **paths*)

Join one or more path components intelligently. The return value is the concatenation of *path* and any members of **paths* with exactly one directory separator (`os.sep`) following each non-empty part except the last, meaning that the result will only end in a separator if the last part is empty. If a component is an absolute path, all previous components are thrown away and joining continues from the absolute path component.

`os.path.`**`isfile`**(*path*)

Return `True` if *path* is an existing regular file. This follows symbolic links, so both **`islink()`** and **`isfile()`** can be true for the same path.

`os.path.`**`abspath`**(*path*)

Return a normalized absolutized version of the pathname *path*. On most platforms, this is equivalent to calling the function **`normpath()`** as follows: `normpath(join(os.getcwd(), path))`.

`os.path.`**`exists`**(*path*)

Return `True` if *path* refers to an existing path. Returns `False` for broken symbolic links. On some platforms, this function may return `False` if permission is not granted to execute **`os.stat()`** on the requested file, even if the *path* physically exists.

`os.path.`**`isdir`**(*path*)

Return `True` if *path* is an existing directory. This follows symbolic links, so both **`islink()`** and **`isdir()`** can be true for the same path.

Question 1

The `os` *module provides a function called* `walk` *that is similar to one below but more versatile. Read the documentation and use it to print the names of the files in a given directory and its subdirectories.*

```
def walk(dirname):
    for name in os.listdir(dirname):
        path = os.path.join(dirname, name)

        if os.path.isfile(path):
            print path
        else:
            walk(path)
```

Question 2

Write a function called `sed` *that takes as arguments a pattern string, a replacement string, and two filenames; it should read the first file and write the contents into the second file (creating it if necessary). If the pattern string appears anywhere in the file, it should be replaced with the replacement string.*

If an error occurs while opening, reading, writing or closing files, your program should catch the exception, print an error message, and exit.

Question 3

Type this example into a file named `wc.py` *and run it as a script. Then run the Python interpreter and* `import wc`*. What is the value of* `__name__` *when the module is being imported?*

Question 4

The `urllib` *module provides methods for manipulating URLs and downloading information from the web. The following example downloads and prints a secret message from* `thinkpython.com`*:*

```
import urllib

conn = urllib.urlopen('http://thinkpython.com/secret.html')
for line in conn:
    print line.strip()
```

Run this code and follow the instructions you see there.