

CMPUT 275 Wi18 - INTRO TO TANGIBLE COMPUT II

Combined LBL Wi18

Exercise 1: Word frequency counter

The goal of this exercise is to

- introduce command line argument processing
- show typical file processing
- get experience with dictionaries

Submit a single plain text file called `freq.py` that contains your solution. As with last exercise, you do not have to submit a README nor do you have to zip the file. Just submit the plain text `.py` file.

Unix man pages

Note: this is a motivating example only. The actual task (described below) does not require any knowledge of unix commands beyond running a python script with `python3`

The typical unix command has a description that begins with something like:

```
sort [OPTION]... [FILE]...
```

The `[OPTION]` part is where command line options are placed, with the convention that options begin with `-` or `--`. For example

```
-c, --check
    check whether input is sorted; do not sort

-k, --key=POS1[,POS2]
    start a key at POS1, end it at POS2 (origin 1)
```

Some options are just flags, like `-c`, others are (key,value) pairs like `--key=-1,2`

After the options, the remaining tokens are typically file names.

Try

```
man sort
```

at the terminal to see the typical glory details of a man page.

Processing command line arguments

When you enter a command into the unix shell, the tokens on the line are passed as a list, `sys.argv`, to your program. You then have to interpret them as options and file names.

For example this program (`cl1.py`) shows all the options passed to it. Note that blanks are used to

```
import sys
for t in sys.argv:
    print("|{}|".format(t))
```

This command

```
$ python3 cl1.py arg "hello" ' ' "" "two tokens" last
```

outputs

```
|cl1.py|
|arg|
|hello|
| |
||
|two tokens|
|last|
```

Note how the program name is the first argument in the list.

There is a python module called argparse that makes command line parsing relatively simple. The progam cl2.py shows how you could do it directly, and more aptly illustrates why you want to use something like argparse. The program worder.py illustrates how to use argparse.

Your Task

Write a program **freq.py** that does text analysis on an input text. The main idea is to read in a text and do frequency analysis of the words in the text. After reading in the file and breaking it into words (a word is any sequence of non-blank characters), the program prints a frequency table of the words in the input file. Lines in a frequency table look like:

```
word count freq
```

where

- word is the word in the text
- count is the number of times the word mentioned occurs
- freq is a number in the range [0,1] that is the ratio of the count for the word to the total number of words found in the text.

The behaviour of the program is controlled by the following options:

--ignore-case

an option that ignores the case (upper/lower) when doing all actions.

--sort=[{byfreq,byword}]

byfreq - print a frequency table sorted by frequency in decreasing frequency order.

byword - print a frequency table sorted by word in increasing lexicographical order.

--remove-punct - throws away any punctuation in a word. Punctuation characters are those for which curses.ascii.ispunct() from the curses.ascii module returns True.

For an example, see the file sample-output.txt included with this description.

You may assume the user always uses this script correctly (i.e. will only call with valid command line arguments and a valid text file).

You are not required to use argparse, but you may if you want.

Examples of Help Output

Your program, on this command,

```
$ python3 freq.py --help
```

should produce something **like** this (it does not need to match exactly, as long as the intended message is made clear)

```
usage: freq.py [-h] [--help] [--sort=[{byfreq,byword}]] [--ignore-case]
[--remove-punct] [infile]
```

Text frequency analysis.

positional arguments:

```
infile                file to be sorted, stdin if omitted.
```

optional arguments:

```
--help or -h          show this help message and exit
```

```
--sort=[{byfreq,byword}]
```

Frequency table sort options:

```
byfreq - (default) sort by decreasing frequency.
```

```
byword - sort by word in increasing lexicographical order
```

```
--ignore-case          ignore upper/lower case when doing all actions.
```

```
--remove-punct         remove all punctuation characters in a word,
                        preserving only the alphanumeric characters
```

cl1.py

cl2.py



sample-input.txt

sample-output.txt



worder.py

Submission status

Attempt number	This is attempt 1.	
Submission status	Submitted for grading	2018-04-14, 1:00 p.m.

Assignment	Grading status	Graded	https://eclass.srv.ualberta.ca/mod/assign/view.php...
	Due date	Monday, 22 January 2018, 11:55 PM	
	Time remaining	Assignment was submitted 1 day 3 hours early	
	Last modified	Sunday, 21 January 2018, 8:31 PM	
	File submissions	<div>  freq.py  </div> <div>Export to portfolio</div>	
	Submission comments	► Comments (0)	

Feedback

Grade	9.50 / 10.00
Graded on	Thursday, 25 January 2018, 4:58 PM
Graded by	 Jesse Farebrother
Feedback comments	 <p>Sorting by word should sort lexicographically. You convert every word to lowercase independent of the `--ignore-case` flag. Other than that good job :)</p>