Lecture 11: Bash Shell, Command Line

LING 1340/2340: Data Science for Linguists

Na-Rae Han

Objectives

- Finally, Bash shell
- Running things in command line
- Interacting with text files in command line

Resources

- Learning resources section:
  - http://www.pitt.edu/~naraehan/ling1340/resources.html#bash
  - Software Carpentry, The Unix Shell:
    - http://swcarpentry.github.io/shell-novice/
  - Thirty Useful Unix Commands:
    - http://www.maths.manchester.ac.uk/~pjohnson/resources/unixShort/examples-commands.pdf
    - Ones you do not need:
      - compress, finger, lpr, talk
      - (Windows) "more" is not supported. Use "less" instead.

Shell introduction, navigating

- Introducing the shell
  - http://swcarpentry.github.io/shell-novice/01-intro/
- Navigating & working with files and directories
  - http://swcarpentry.github.io/shell-novice/02-filedir/
  - http://swcarpentry.github.io/shell-novice/03-create/

- We've been doing some of these already, as part of our git routine. You should know:
  - `.`   `..`   `~`
  - `pwd`
  - `cd`
  - `ls`
  - Command-line history with ↑ and ↓
  - Using TAB for file name completion
  - Using Control+C to quit

Settling in, customizing

- You can customize your shell.
  - `.bashrc`
  - `.bash_profile`

  ← These files store your customization.

Mac users: color option is not supported. Remove the color part.

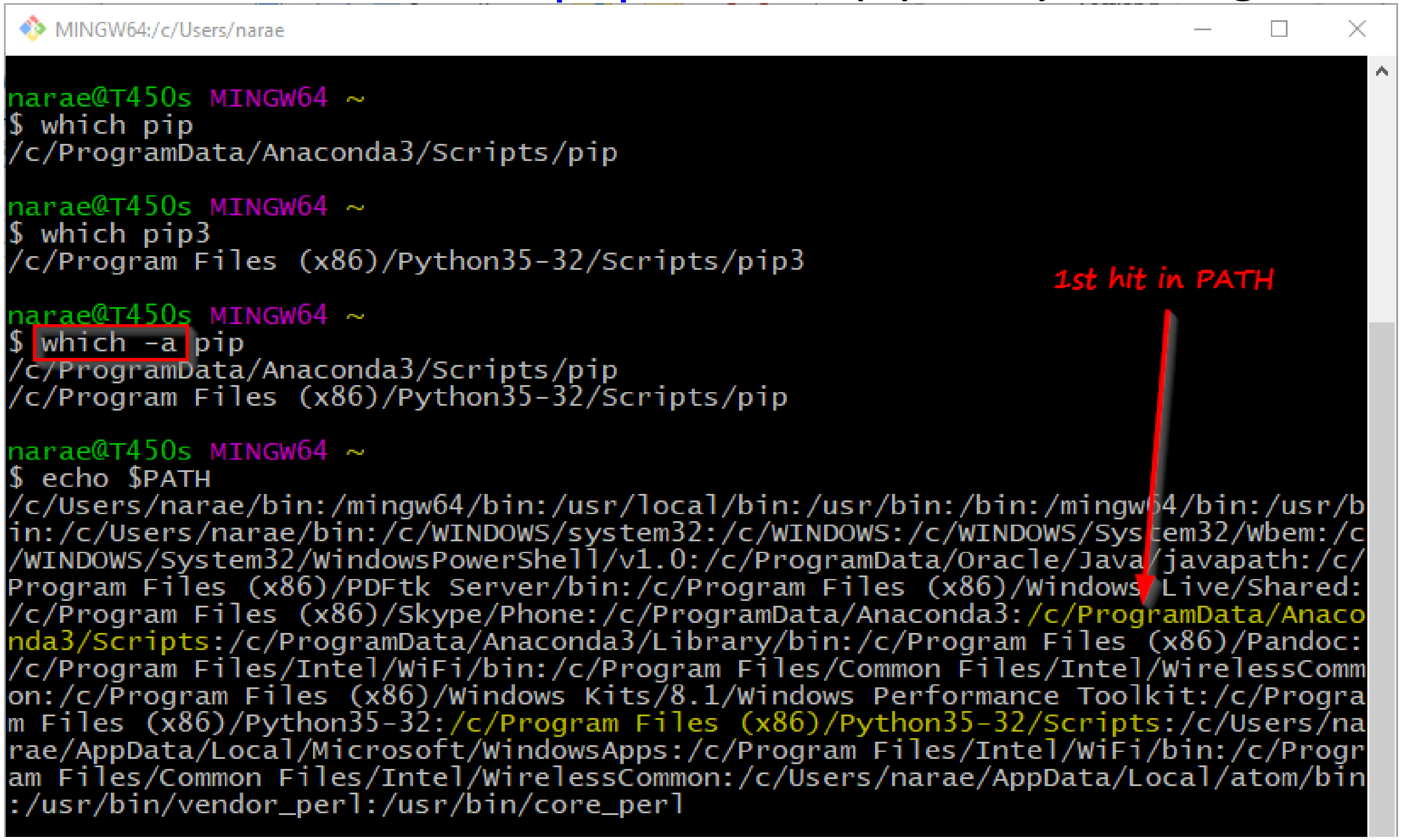
- In your home directory:
  - *your_editor* `.bash_profile &`
  - After adding entries or editing, you should either log back in, or execute `source .bash_profile`.
- Aliasing is the most common customization method:

```
alias calc='/c/windows/system32/calc.exe'
alias ls='ls -hF --color=tty'
```

  ← Your favorite shortcuts and command-line options

PATH, which, where

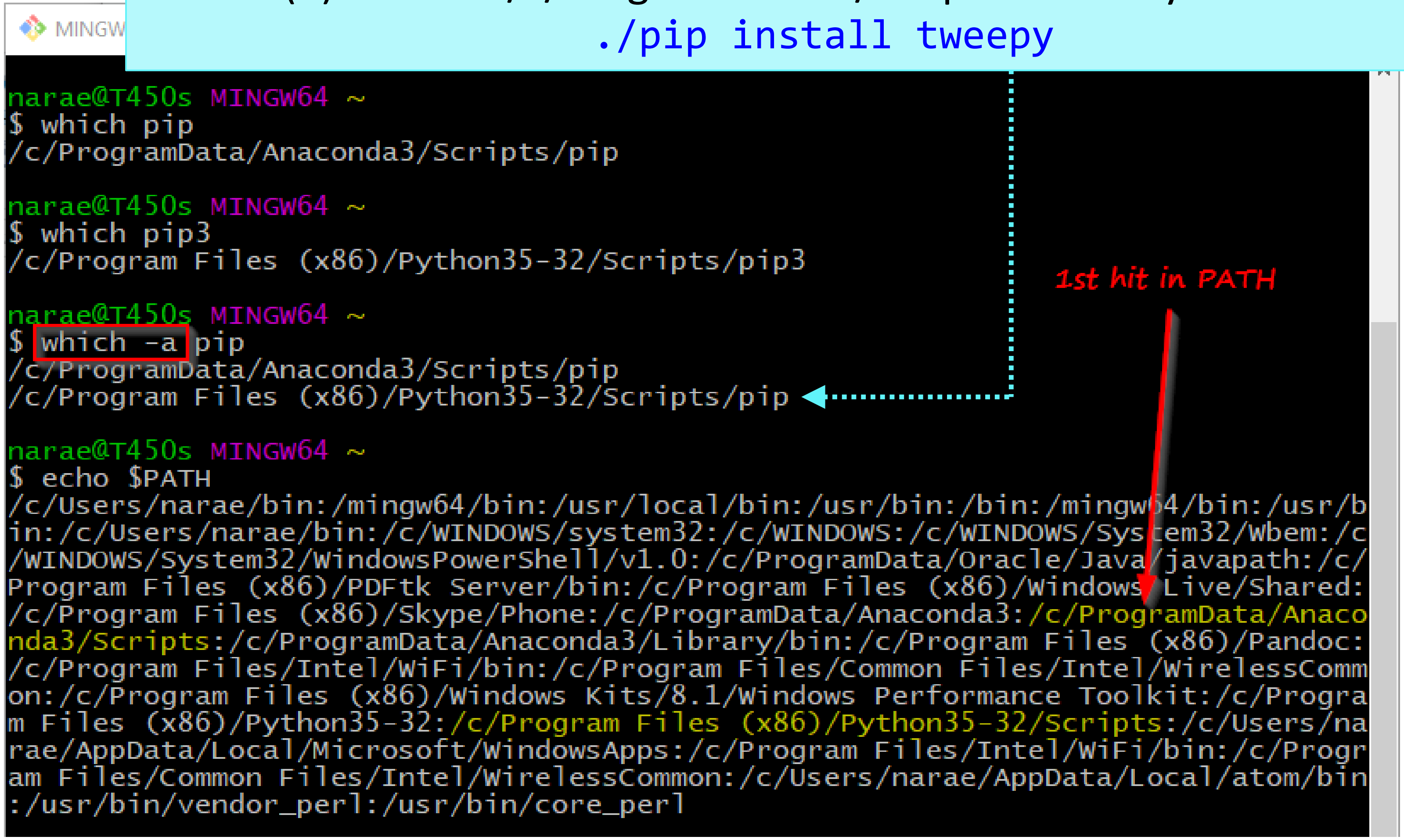
- We have been occasionally using `pip` to install Python libraries. Where is this `pip`? Which pip are you using?



MINGW64:/c/Users/narae

```
narae@T450s MINGW64 ~
$ which pip
/c/ProgramData/Anaconda3/Scripts/pip

narae@T450s MINGW64 ~
$ which pip3
/c/Program Files (x86)/Python35-32/Scripts/pip3

narae@T450s MINGW64 ~
$ which -a pip
/c/ProgramData/Anaconda3/Scripts/pip
/c/Program Files (x86)/Python35-32/Scripts/pip

narae@T450s MINGW64 ~
$ echo $PATH
/c/Users/narae/bin:/mingw64/bin:/usr/local/bin:/usr/bin:/bin:/mingw64/bin:/usr/b
in:/c/Users/narae/bin:/c/WINDOWS/system32:/c/WINDOWS:/c/WINDOWS/System32/Wbem:/c
/WINDOWS/System32/WindowsPowerShell/v1.0:/c/ProgramData/Oracle/Java/javapath:/c/
Program Files (x86)/PDFtk Server/bin:/c/Program Files (x86)/Windows Live/Shared:
/c/Program Files (x86)/Skype/Phone:/c/ProgramData/Anaconda3:/c/ProgramData/Anaco
nda3/Scripts:/c/ProgramData/Anaconda3/Library/bin:/c/Program Files (x86)/Pandoc:
/c/Program Files/Intel/WiFi/bin:/c/Program Files/Common Files/Intel/WirelessComm
on:/c/Program Files (x86)/Windows Kits/8.1/Windows Performance Toolkit:/c/Progra
m Files (x86)/Python35-32:/c/Program Files (x86)/Python35-32/Scripts:/c/Users/na
rae/AppData/Local/Microsoft/WindowsApps:/c/Program Files/Intel/WiFi/bin:/c/Progr
am Files/Common Files/Intel/WirelessCommon:/c/Users/narae/AppData/Local/atom/bin
:/usr/bin/vendor_perl:/usr/bin/core_perl
```

PATH, which, where



MINGW

```
narae@T450s MINGW64 ~
$ which pip
/c/ProgramData/Anaconda3/Scripts/pip

narae@T450s MINGW64 ~
$ which pip3
/c/Program Files (x86)/Python35-32/Scripts/pip3

narae@T450s MINGW64 ~
$ which -a pip
/c/ProgramData/Anaconda3/Scripts/pip
/c/Program Files (x86)/Python35-32/Scripts/pip

narae@T450s MINGW64 ~
$ echo $PATH
/c/Users/narae/bin:/mingw64/bin:/usr/local/bin:/usr/bin:/bin:/mingw64/bin:/usr/b
in:/c/Users/narae/bin:/c/WINDOWS/system32:/c/WINDOWS:/c/WINDOWS/System32/Wbem:/c
/WINDOWS/System32/WindowsPowerShell/v1.0:/c/ProgramData/Oracle/Java/javapath:/c/
Program Files (x86)/PDFtk Server/bin:/c/Program Files (x86)/Windows Live/Shared:
/c/Program Files (x86)/Skype/Phone:/c/ProgramData/Anaconda3:/c/ProgramData/Anaco
nda3/Scripts:/c/ProgramData/Anaconda3/Library/bin:/c/Program Files (x86)/Pandoc:
/c/Program Files/Intel/WiFi/bin:/c/Program Files/Common Files/Intel/WirelessComm
on:/c/Program Files (x86)/Windows Kits/8.1/Windows Performance Toolkit:/c/Progra
m Files (x86)/Python35-32:/c/Program Files (x86)/Python35-32/Scripts:/c/Users/na
rae/AppData/Local/Microsoft/WindowsApps:/c/Program Files/Intel/WiFi/bin:/c/Progr
am Files/Common Files/Intel/WirelessCommon:/c/Users/narae/AppData/Local/atom/bin
:/usr/bin/vendor_perl:/usr/bin/core_perl
```

1st hit in PATH

If you want to install tweepy for this version of python, you can do:

(1) `pip3 install tweepy`

(2) `/c/Program\ Files.../Scripts/pip install tweepy`

(3) cd into /c/Program Files.../Scripts directory and then `./pip install tweepy`

Windows users

- Because git-bash is not a native command-line shell for Windows (cmd is), there are a few additional wrinkles.
- Certain programs are designed to run within a console window. Those need to be prefixed with *winpty* if you want Python shell.
  - `winpty python`
- Pay attention to your directory path.
  - In git-bash, full path starts with `/c/`.
  - In cmd (Windows native), it is `C:\...`
  - In Python, full path can be written as `'C:/...'` or `'C:\\...'` or `r'C:\...'`.
- Not found:
  - `nano` (need to download and do some setting up. See next page.)
  - `more` (use `less`)
  - `man` (Google stuff up)

Setting up nano 2.7.5 to use in Git Bash on Windows

1. Download nano-git-0d9a7347243.exe from:
   - https://www.nano-editor.org/dist/win32-support/
2. Rename it `nano-git.exe`
3. Place it under /usr/bin/.
   - This is likely C:\Program Files\Git\usr\bin if you installed git with default setting.
4. Create a script file called nano (without file extension!) in the same /usr/bin/ directory. Put two lines:

```
#!/bin/sh
START //WAIT nano-git.exe "$@"
```

5. You can now launch nano editor by simply calling 'nano' or 'nano foo.txt'.

Mac users

- Practice nano.
- Add some aliases.
- Like in Windows, you should be able to launch any app that is found in your PATH.
- Surprise! You also have a handy command for launching *any* GUI application from command-line.
  - `open -a Application-Name`
  - http://osxdaily.com/2007/02/01/how-to-launch-gui-applications-from-the-terminal/

Running python script from command-line

1. `python hello.py`
   - Assuming python is in your $PATH, and hello.py is in your current working directory
2. `hello.py`
   - Assuming your current working directory is in your $PATH. If not, you should execute `./hello.py`
   - Assuming your script begins with a line (called 'shebang' line):

     `#!/system_path/to/python`

     - In my case, it's `#!/c/ProgramData/Anaconda3/python`
     - If your path contains a SPACE... tough luck!

Piping and I/O redirections

- Piping is what makes command-line ever so powerful.
- For people working mainly with text data (us!), piping enables us to manipulate data on the fly.
  - `hello.py > out.txt` redirect output to file
  - `hello.py | wc` redirect output to another application
  - `hello.py | wc > out.txt` daisy chain!

  - Also:
  - `<` read in from a file input
  - `>>` append to existing file rather than overwriting

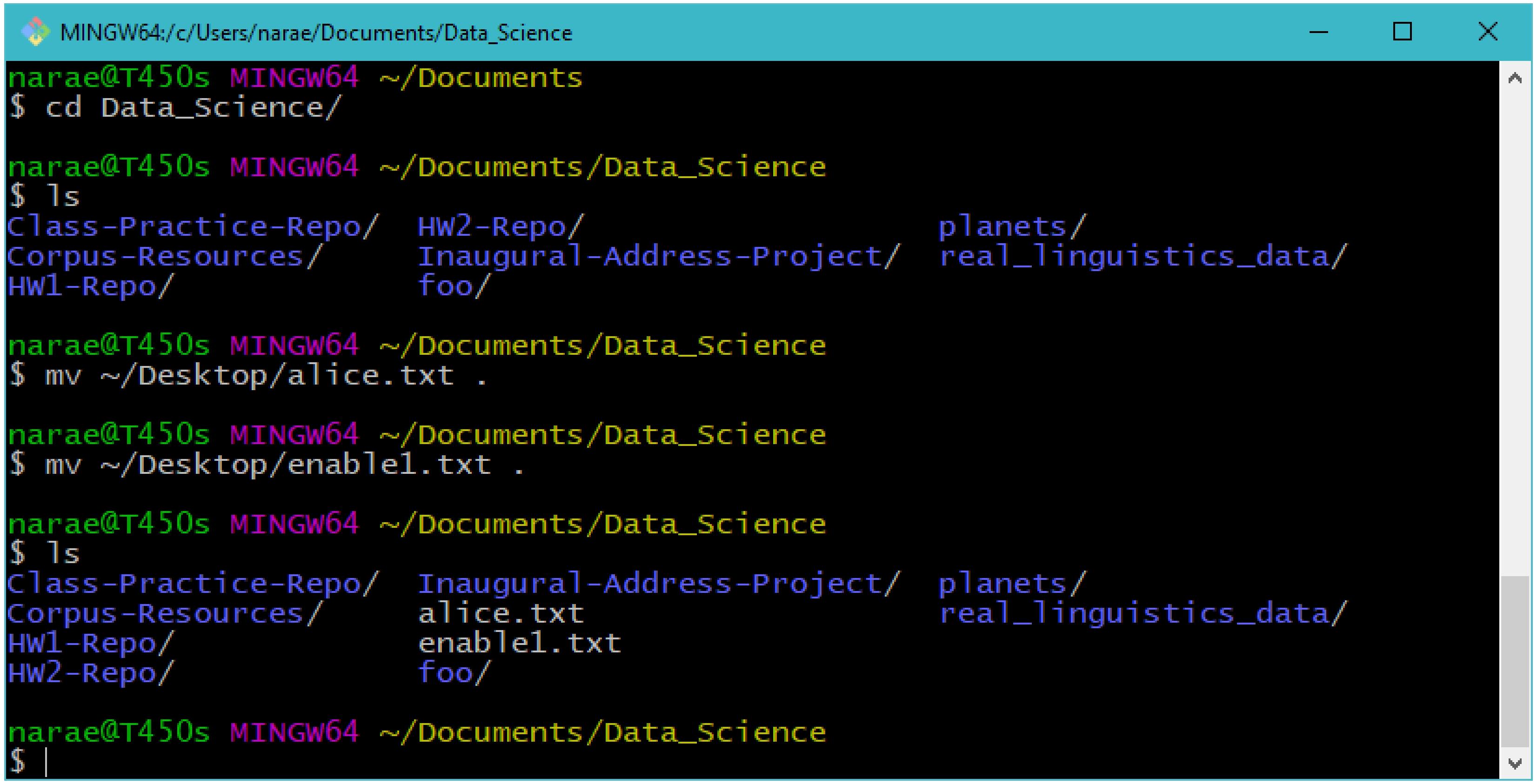
Download two files

- Alice's Adventures in Wonderland
  - http://www.gutenberg.org/ebooks/11
  - Download the Plain Text UTF-8 version.
  - Rename the file to alice.txt
- ENABLE word list from Peter Norvig's site:
  - http://norvig.com/ngrams/
  - Download enable1.txt

🡐 Save them onto your Desktop.

🡐 Then, within bash shell, move the files into your Data_Science directory. (Wait if you are not sure how this is done.)

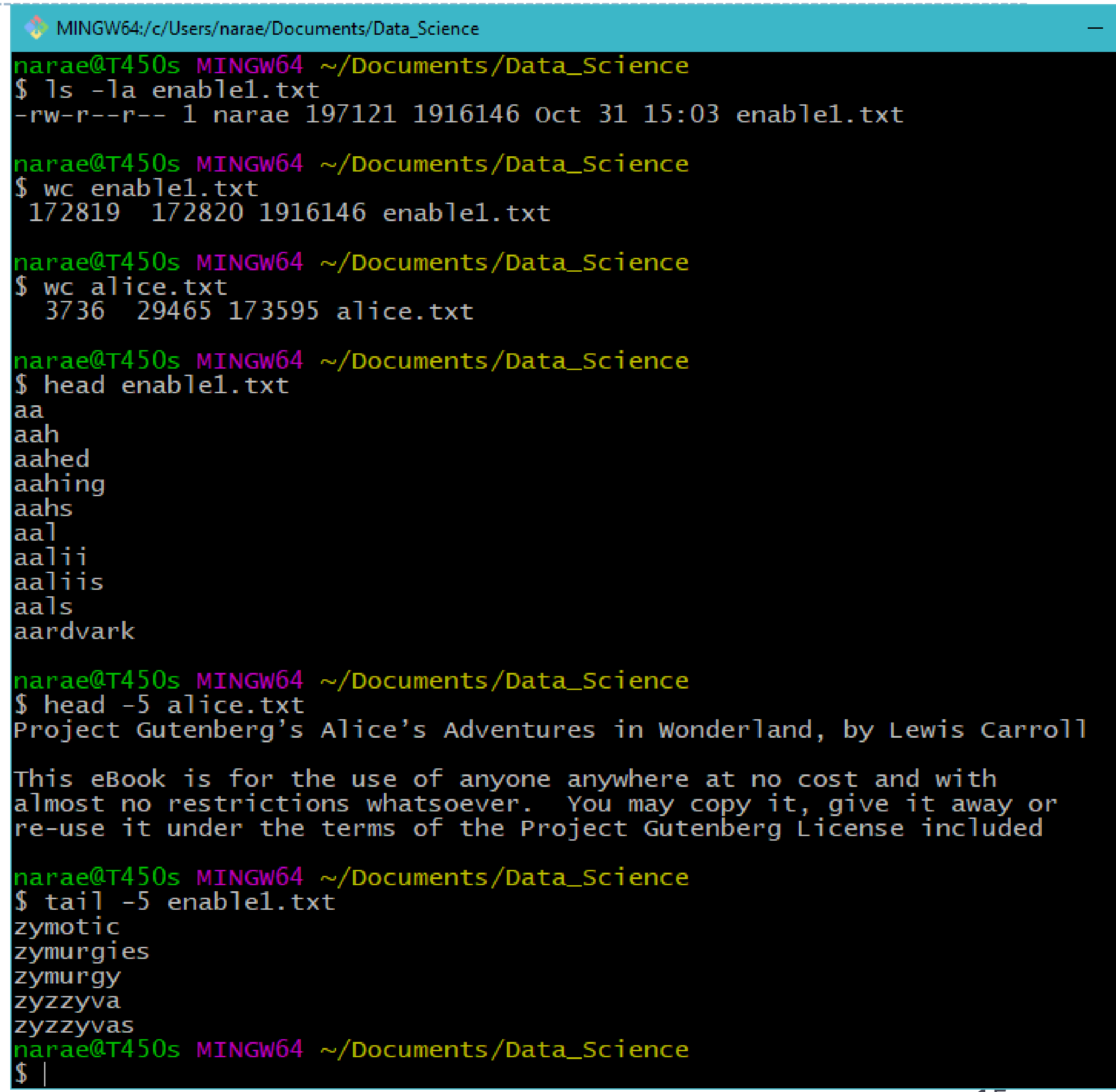
Files in your Data_Science directory



```
narae@T450s MINGW64 ~/Documents
$ cd Data_Science/

narae@T450s MINGW64 ~/Documents/Data_Science
$ ls
Class-Practice-Repo/  HW2-Repo/                     planets/
Corpus-Resources/     Inaugural-Address-Project/  real_linguistics_data/
HW1-Repo/             foo/

narae@T450s MINGW64 ~/Documents/Data_Science
$ mv ~/Desktop/alice.txt .

narae@T450s MINGW64 ~/Documents/Data_Science
$ mv ~/Desktop/enable1.txt .

narae@T450s MINGW64 ~/Documents/Data_Science
$ ls
Class-Practice-Repo/  Inaugural-Address-Project/  planets/
Corpus-Resources/     alice.txt                   real_linguistics_data/
HW1-Repo/             enable1.txt
HW2-Repo/             foo/

narae@T450s MINGW64 ~/Documents/Data_Science
$ |
```

Examining a text file

- `ls -la`
  - Displays file info
- `wc`
  - Displays line count, word count, and character count
- `head -n`
  - Displays initial n lines
- `tail -n`
  - Displays last n lines

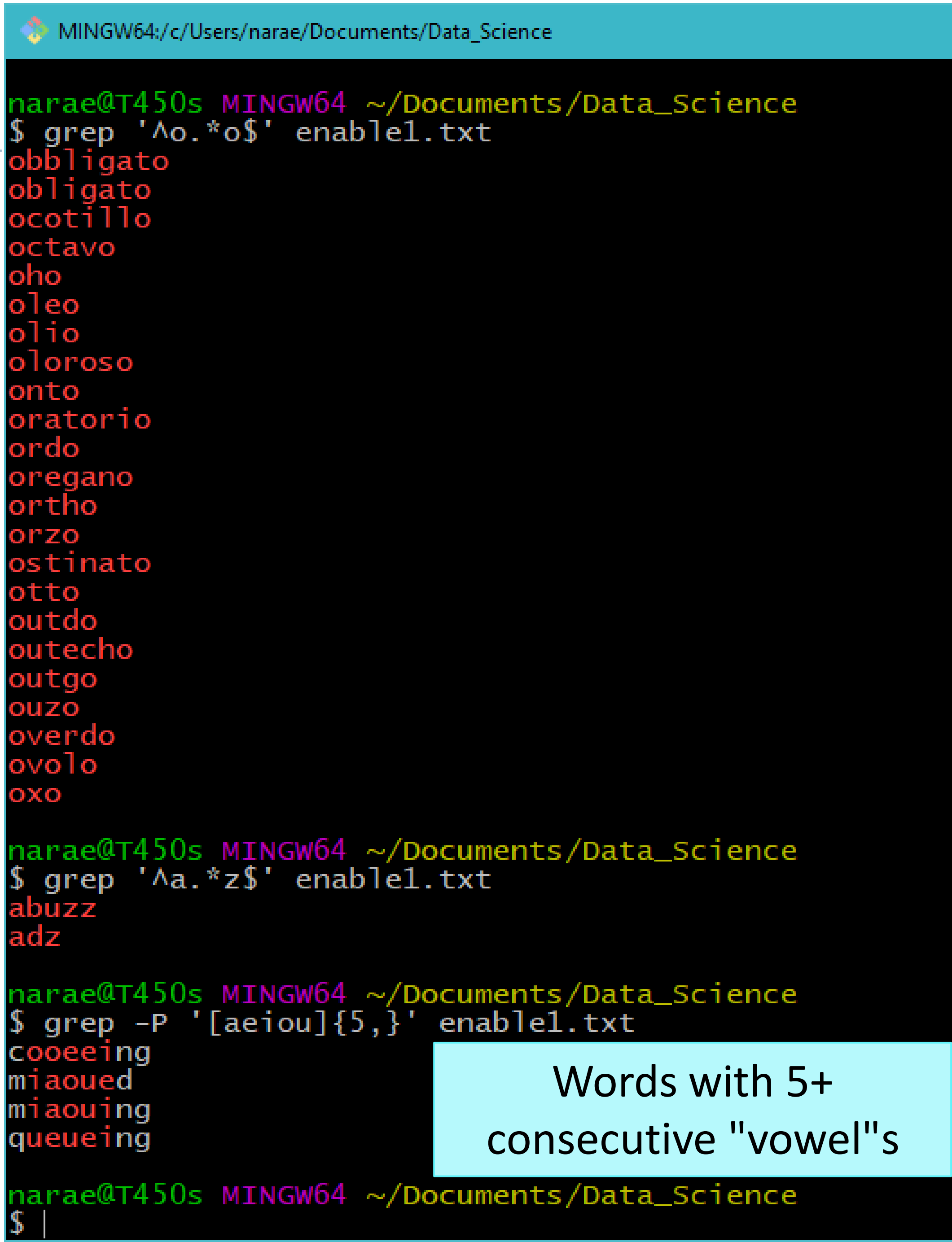


MINGW64:/c/Users/narae/Documents/Data_Science

```
narae@T450s MINGW64 ~/Documents/Data_Science
$ ls -la enable1.txt
-rw-r--r-- 1 narae 197121 1916146 Oct 31 15:03 enable1.txt

narae@T450s MINGW64 ~/Documents/Data_Science
$ wc enable1.txt
 172819  172820 1916146 enable1.txt

narae@T450s MINGW64 ~/Documents/Data_Science
$ wc alice.txt
  3736  29465 173595 alice.txt

narae@T450s MINGW64 ~/Documents/Data_Science
$ head enable1.txt
aa
aah
aahed
aahing
aahs
aal
aalii
aaliis
aals
aardvark

narae@T450s MINGW64 ~/Documents/Data_Science
$ head -5 alice.txt
Project Gutenberg's Alice's Adventures in Wonderland, by Lewis Carroll

This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Project Gutenberg License included

narae@T450s MINGW64 ~/Documents/Data_Science
$ tail -5 enable1.txt
zymotic
zymurgies
zymurgy
zyzzyva
zyzzyvas
narae@T450s MINGW64 ~/Documents/Data_Science
$ |
```

grep

- `grep`
  - Searches each line in text for regular expression match
- `grep -P`
  - Accepts perl-style regular expressions
  - Perl-style == Python-style



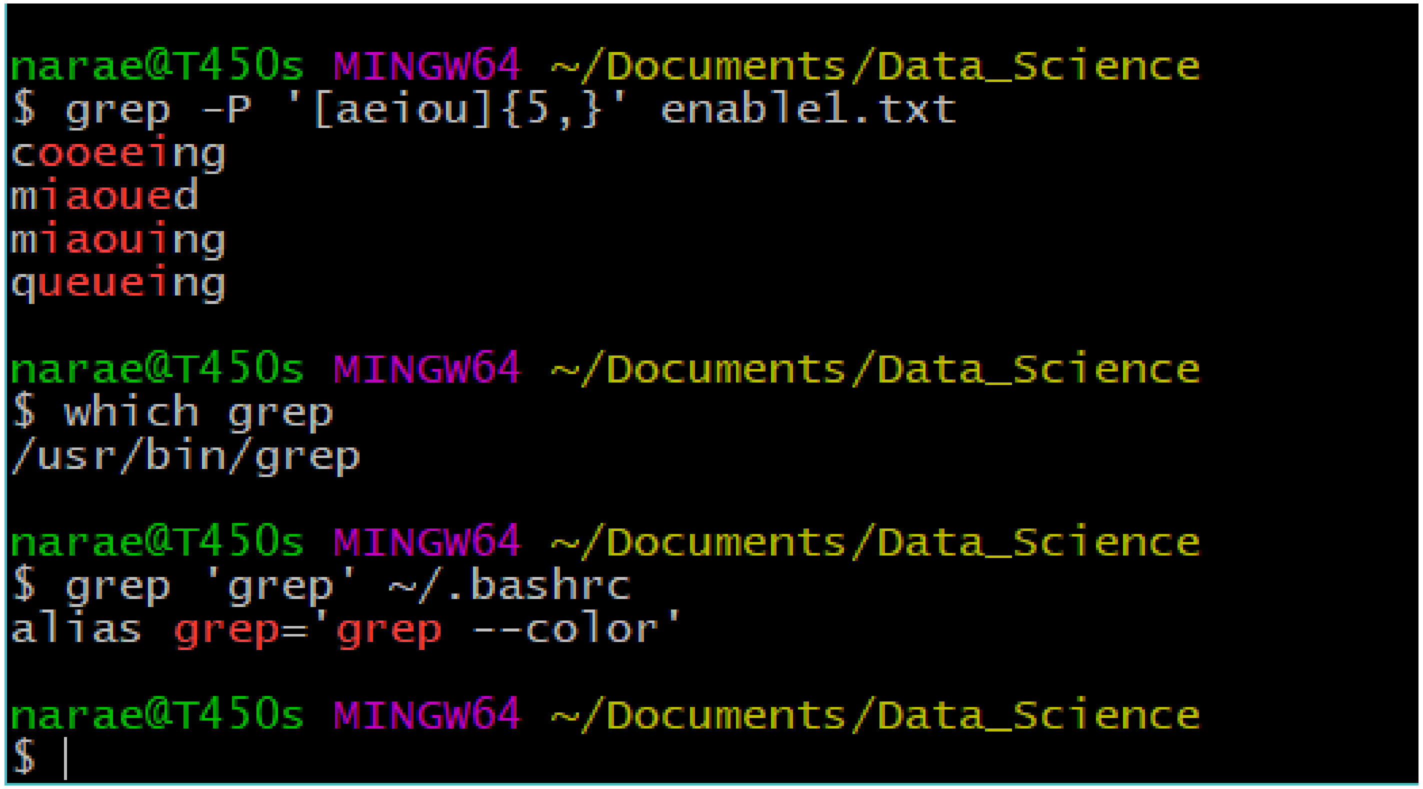
MINGW64:/c/Users/narae/Documents/Data_Science

```
narae@T450s MINGW64 ~/Documents/Data_Science
$ grep '^o.*o$' enable1.txt
obbligato
obligato
ocotillo
octavo
oho
oleo
olio
oloroso
onto
oratorio
ordo
oregano
ortho
orzo
ostinato
otto
outdo
outecho
outgo
ouzo
overdo
ovolo
oxo

narae@T450s MINGW64 ~/Documents/Data_Science
$ grep '^a.*z$' enable1.txt
abuzz
adz

narae@T450s MINGW64 ~/Documents/Data_Science
$ grep -P '[aeiou]{5,}' enable1.txt
cooeeing
miaoued
miaouing
queueing

narae@T450s MINGW64 ~/Documents/Data_Science
$ |
```

Words with 5+ consecutive "vowel"s

Colorizing grep output

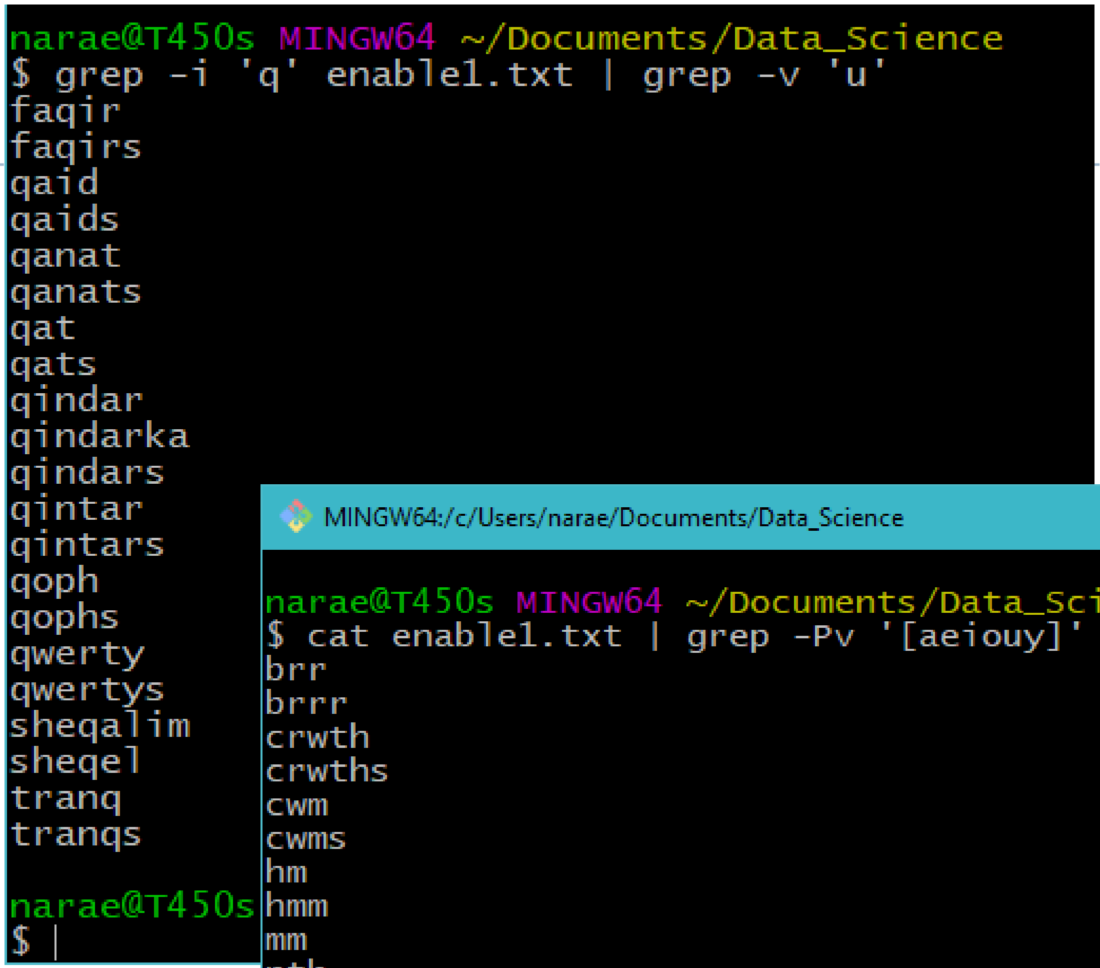
- You might want to colorize your grep output.
- I have grep aliased to use color in my .bashrc file. (Put your alias in .bash_profile)



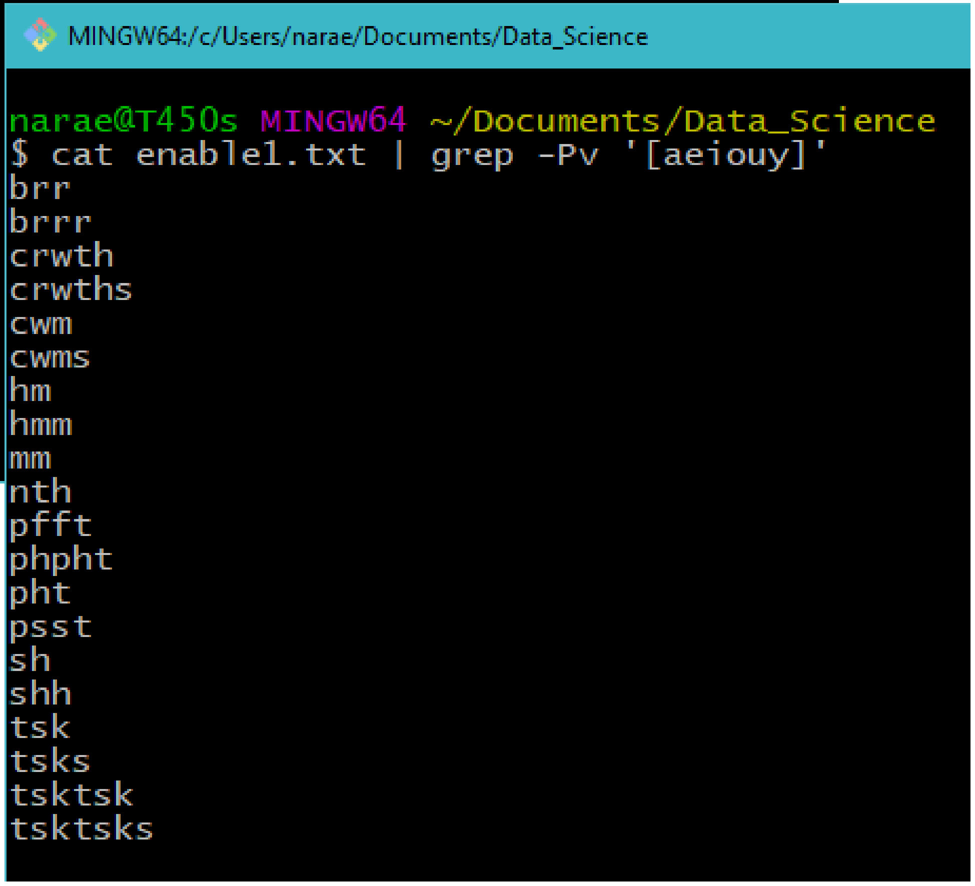
```
narae@T450s MINGW64 ~/Documents/Data_Science
$ grep -P '[aeiou]{5,}' enable1.txt
cooeeing
miaoued
miaouing
queueing

narae@T450s MINGW64 ~/Documents/Data_Science
$ which grep
/usr/bin/grep

narae@T450s MINGW64 ~/Documents/Data_Science
$ grep 'grep' ~/.bashrc
alias grep='grep --color'

narae@T450s MINGW64 ~/Documents/Data_Science
$ |
```

grep -i, -v

- `grep -i`
  - ignores case
- `grep -v`
  - prints lines that DO NOT match



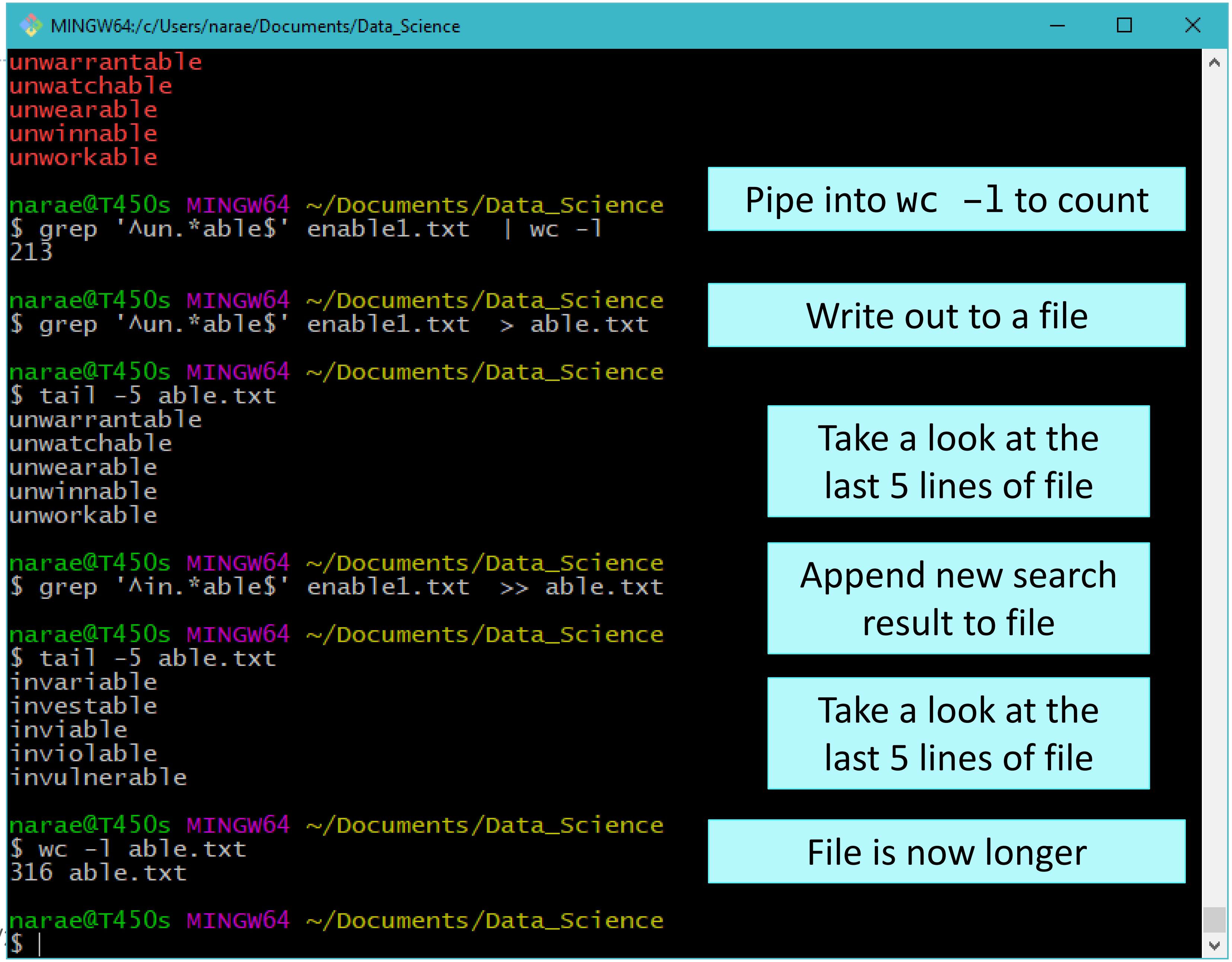
```
narae@T450s MINGW64 ~/Documents/Data_Science
$ grep -i 'q' enable1.txt | grep -v 'u'
faqir
faqirs
qaid
qaids
qanat
qanats
qat
qats
qindar
qindarka
qindars
qintar
qintars
qoph
qophs
qwerty
qwertys
sheqalim
sheqel
tranq
tranqs

narae@T450s
$ |
```



MINGW64:/c/Users/narae/Documents/Data_Science

```
narae@T450s MINGW64 ~/Documents/Data_Science
$ cat enable1.txt | grep -Pv '[aeiouy]'
brr
brrr
crwth
crwths
cwm
cwms
hm
hmm
mm
nth
pffft
phpht
pht
psst
sh
shh
tsk
tsks
tsktsk
tsktsks
```

grep and piping together



MINGW64:/c/Users/narae/Documents/Data_Science

```
unwarrantable
unwatchable
unwearable
unwinnable
unworkable

narae@T450s MINGW64 ~/Documents/Data_Science
$ grep '^un.*able$' enable1.txt  | wc -l
213

narae@T450s MINGW64 ~/Documents/Data_Science
$ grep '^un.*able$' enable1.txt  > able.txt

narae@T450s MINGW64 ~/Documents/Data_Science
$ tail -5 able.txt
unwarrantable
unwatchable
unwearable
unwinnable
unworkable

narae@T450s MINGW64 ~/Documents/Data_Science
$ grep '^in.*able$' enable1.txt  >> able.txt

narae@T450s MINGW64 ~/Documents/Data_Science
$ tail -5 able.txt
invariable
investable
inviable
inviolable
invulnerable

narae@T450s MINGW64 ~/Documents/Data_Science
$ wc -l able.txt
316 able.txt

narae@T450s MINGW64 ~/Documents/Data_Science
$ |
```

Pipe into wc –l to count

Write out to a file

Take a look at the last 5 lines of file

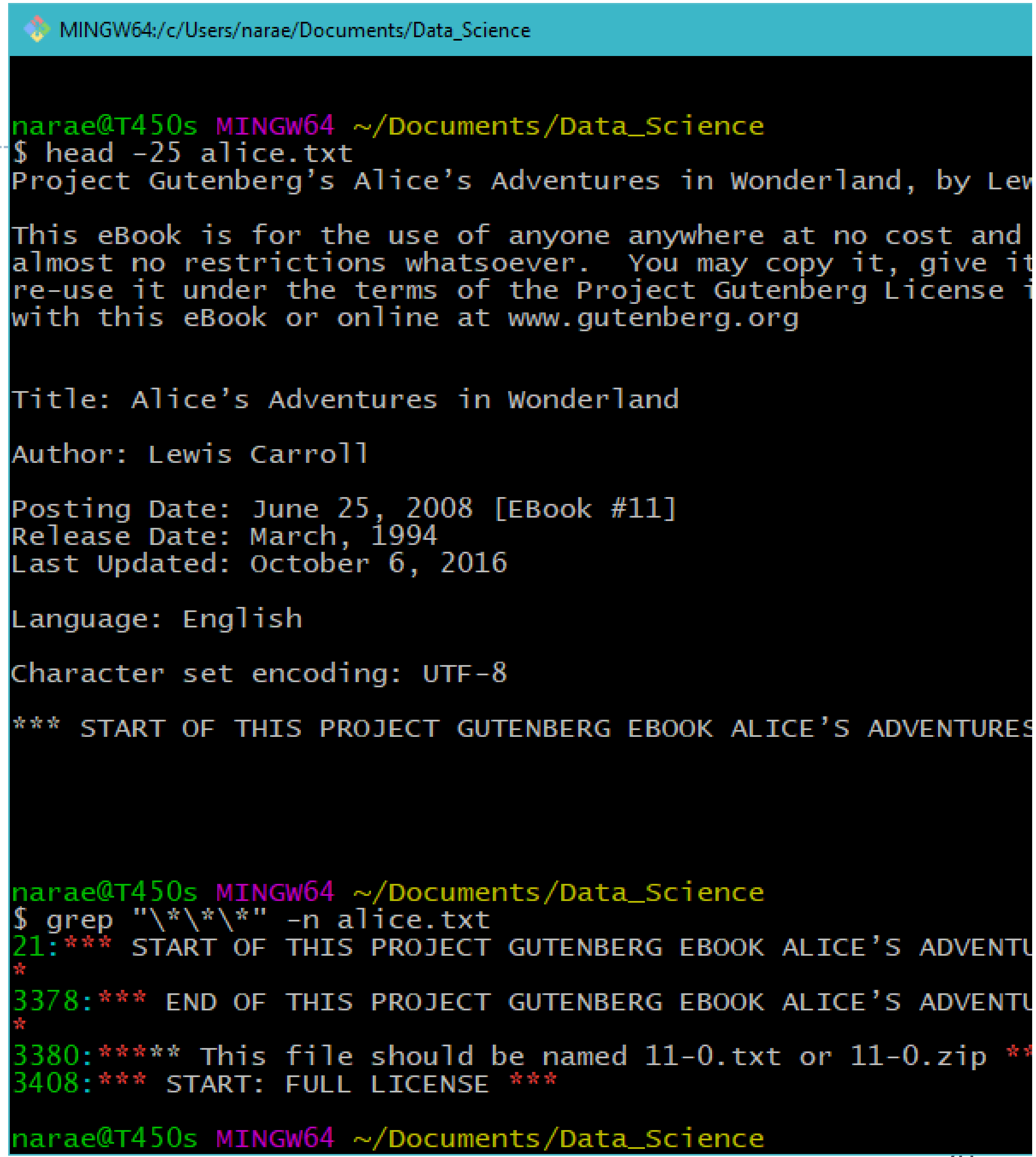
Append new search result to file

Take a look at the last 5 lines of file

File is now longer

grep -n, -l

- `grep -n`
  - prints out line number
- `grep -l`
  - prints file names only if match is found

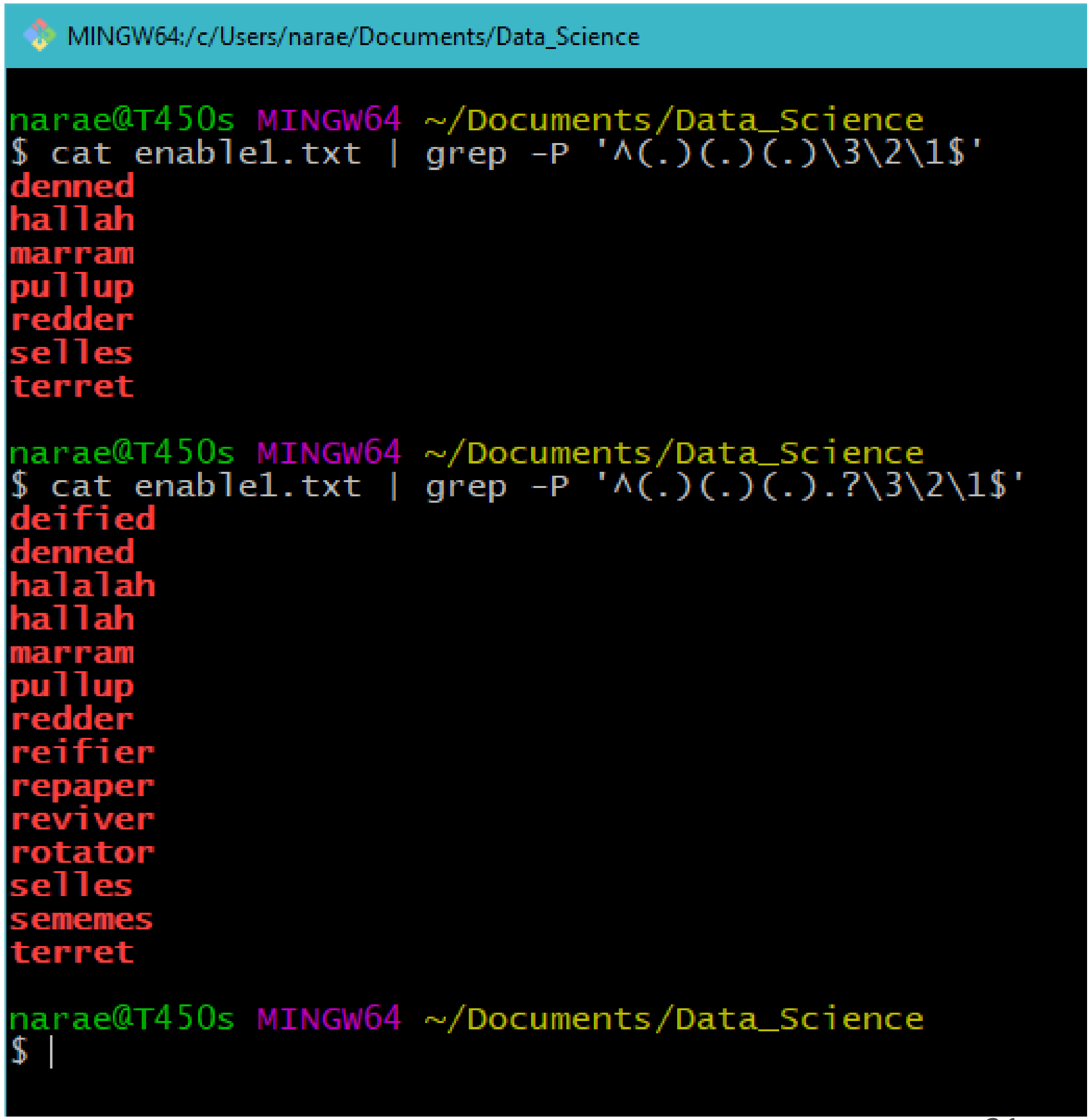


MINGW64:/c/Users/narae/Documents/Data_Science

```
narae@T450s MINGW64 ~/Documents/Data_Science
$ head -25 alice.txt
Project Gutenberg's Alice's Adventures in Wonderland, by Lew

This eBook is for the use of anyone anywhere at no cost and
almost no restrictions whatsoever.  You may copy it, give it
re-use it under the terms of the Project Gutenberg License i
with this eBook or online at www.gutenberg.org


Title: Alice's Adventures in Wonderland

Author: Lewis Carroll

Posting Date: June 25, 2008 [EBook #11]
Release Date: March, 1994
Last Updated: October 6, 2016

Language: English

Character set encoding: UTF-8

*** START OF THIS PROJECT GUTENBERG EBOOK ALICE'S ADVENTURES




narae@T450s MINGW64 ~/Documents/Data_Science
$ grep "\*\*\*" -n alice.txt
21:*** START OF THIS PROJECT GUTENBERG EBOOK ALICE'S ADVENTU
*
3378:*** END OF THIS PROJECT GUTENBERG EBOOK ALICE'S ADVENTU
*
3380:***** This file should be named 11-0.txt or 11-0.zip **
3408:*** START: FULL LICENSE ***

narae@T450s MINGW64 ~/Documents/Data_Science
```

Regular expressions are powerful

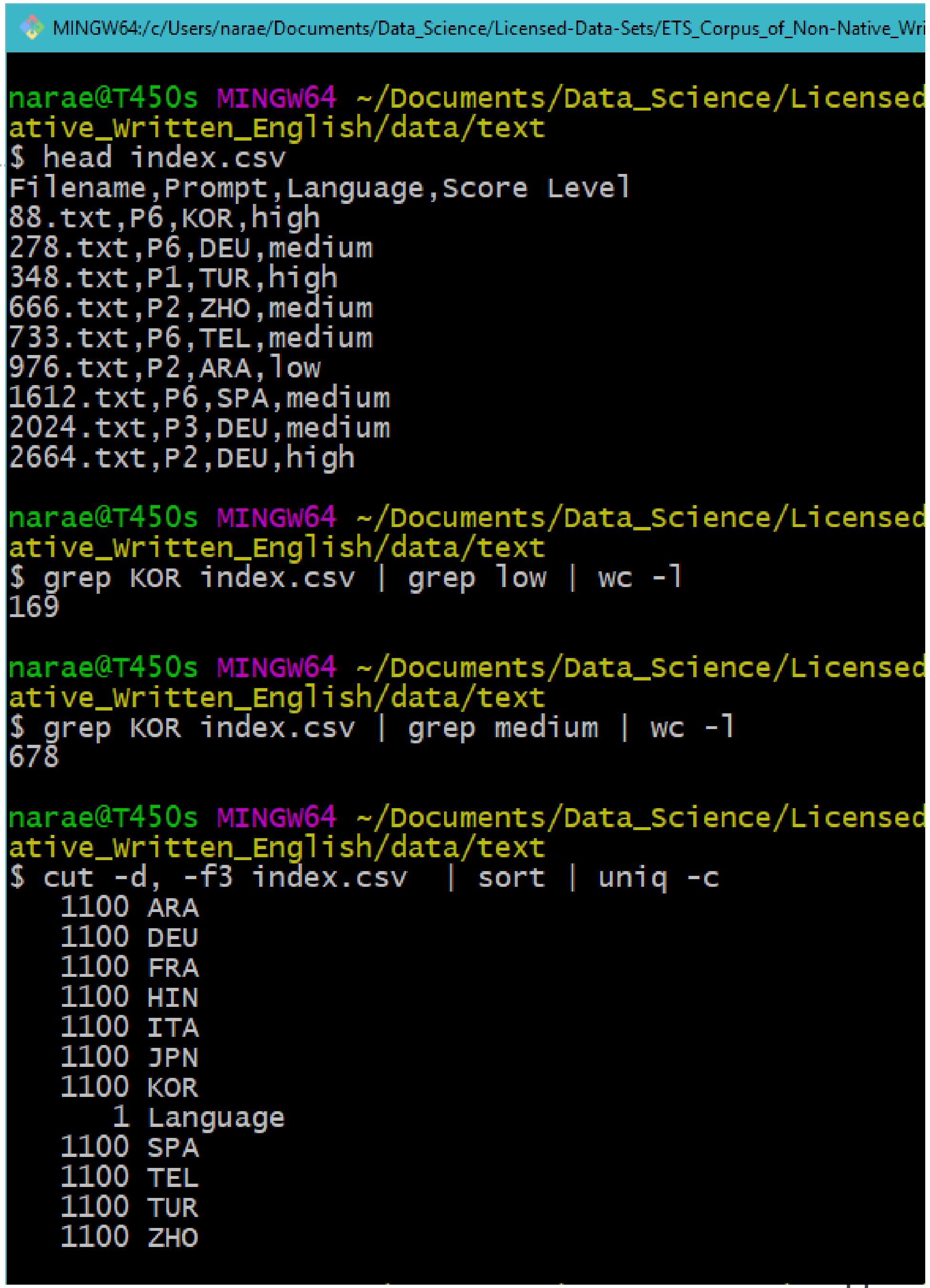
- Learn regex:
  - https://www.regular-expressions.info/tutorial.html
- Palindromes with limited lengths can be captured with regex.
  - \1, \2, \3 are backreferences and match portions in ()



```
narae@T450s MINGW64 ~/Documents/Data_Science
$ cat enable1.txt | grep -P '^(.)(.)(.)\3\2\1$'
denned
hallah
marram
pullup
redder
selles
terret

narae@T450s MINGW64 ~/Documents/Data_Science
$ cat enable1.txt | grep -P '^(.)(.)(.).?\3\2\1$'
deified
denned
halalah
hallah
marram
pullup
redder
reifier
repaper
reviver
rotator
selles
sememes
terret

narae@T450s MINGW64 ~/Documents/Data_Science
$ |
```

sort + uniq + count

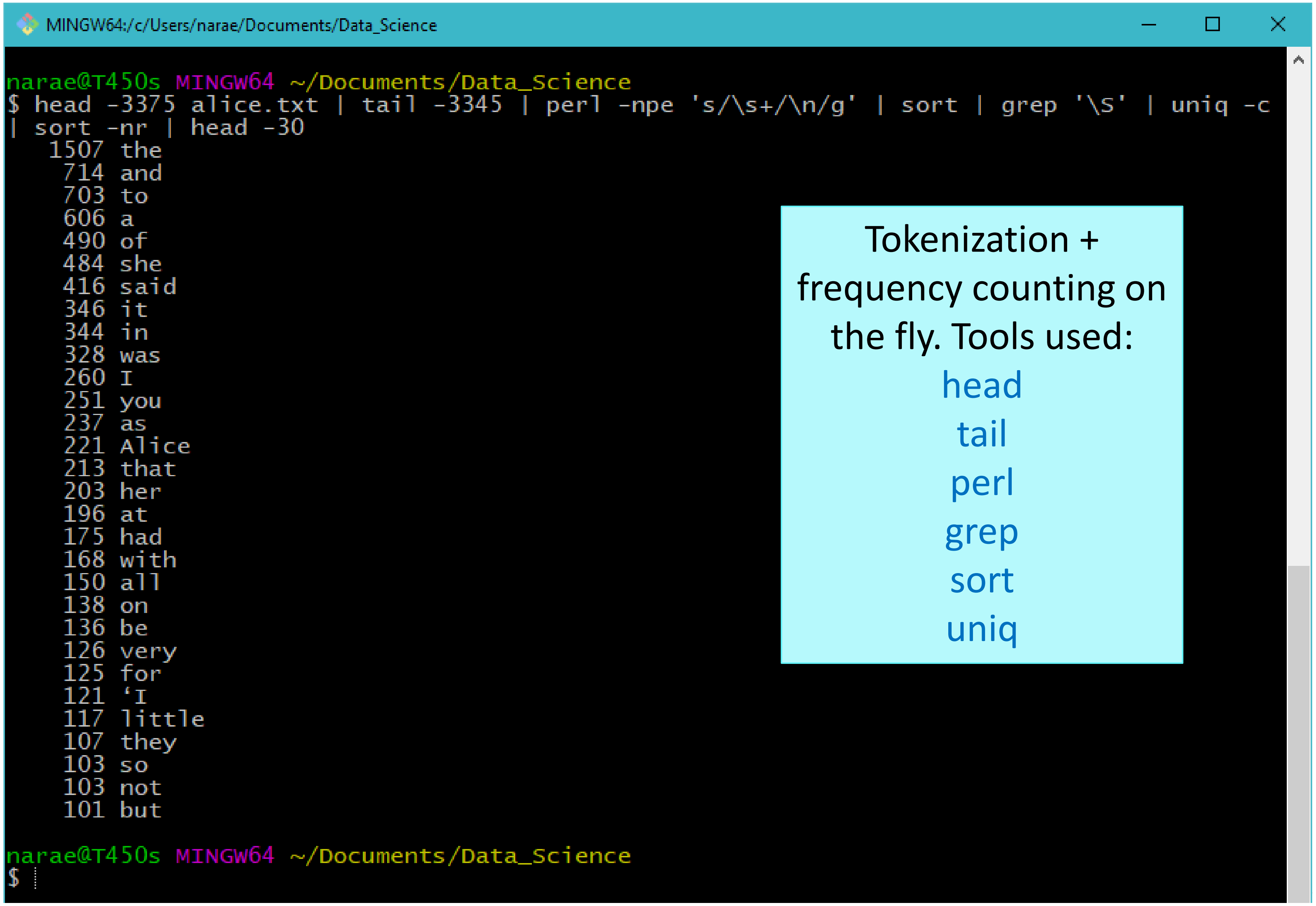
- `sort`
  - sorts lines alphabetically
- `uniq -c`
  - folds identical lines, adding count in front
- `wc -l`
  - counts lines
- `cut -d, -f3`
  - cuts 3rd column with ',' as delimiter



MINGW64:/c/Users/narae/Documents/Data_Science/Licensed-Data-Sets/ETS_Corpus_of_Non-Native_Wri

```
narae@T450s MINGW64 ~/Documents/Data_Science/Licensed
ative_Written_English/data/text
$ head index.csv
Filename,Prompt,Language,Score Level
88.txt,P6,KOR,high
278.txt,P6,DEU,medium
348.txt,P1,TUR,high
666.txt,P2,ZHO,medium
733.txt,P6,TEL,medium
976.txt,P2,ARA,low
1612.txt,P6,SPA,medium
2024.txt,P3,DEU,medium
2664.txt,P2,DEU,high

narae@T450s MINGW64 ~/Documents/Data_Science/Licensed
ative_Written_English/data/text
$ grep KOR index.csv | grep low | wc -l
169

narae@T450s MINGW64 ~/Documents/Data_Science/Licensed
ative_Written_English/data/text
$ grep KOR index.csv | grep medium | wc -l
678

narae@T450s MINGW64 ~/Documents/Data_Science/Licensed
ative_Written_English/data/text
$ cut -d, -f3 index.csv  | sort | uniq -c
   1100 ARA
   1100 DEU
   1100 FRA
   1100 HIN
   1100 ITA
   1100 JPN
   1100 KOR
      1 Language
   1100 SPA
   1100 TEL
   1100 TUR
   1100 ZHO
```

Piping gone mad



MINGW64:/c/Users/narae/Documents/Data_Science

```
narae@T450s MINGW64 ~/Documents/Data_Science
$ head -3375 alice.txt | tail -3345 | perl -npe 's/\s+/\n/g' | sort | grep '\S' | uniq -c
| sort -nr | head -30
   1507 the
    714 and
    703 to
    606 a
    490 of
    484 she
    416 said
    346 it
    344 in
    328 was
    260 I
    251 you
    237 as
    221 Alice
    213 that
    203 her
    196 at
    175 had
    168 with
    150 all
    138 on
    136 be
    126 very
    125 for
    121 'I
    117 little
    107 they
    103 so
    103 not
    101 but

narae@T450s MINGW64 ~/Documents/Data_Science
$
```

Tokenization + frequency counting on the fly. Tools used:

head
tail
perl
grep
sort
uniq

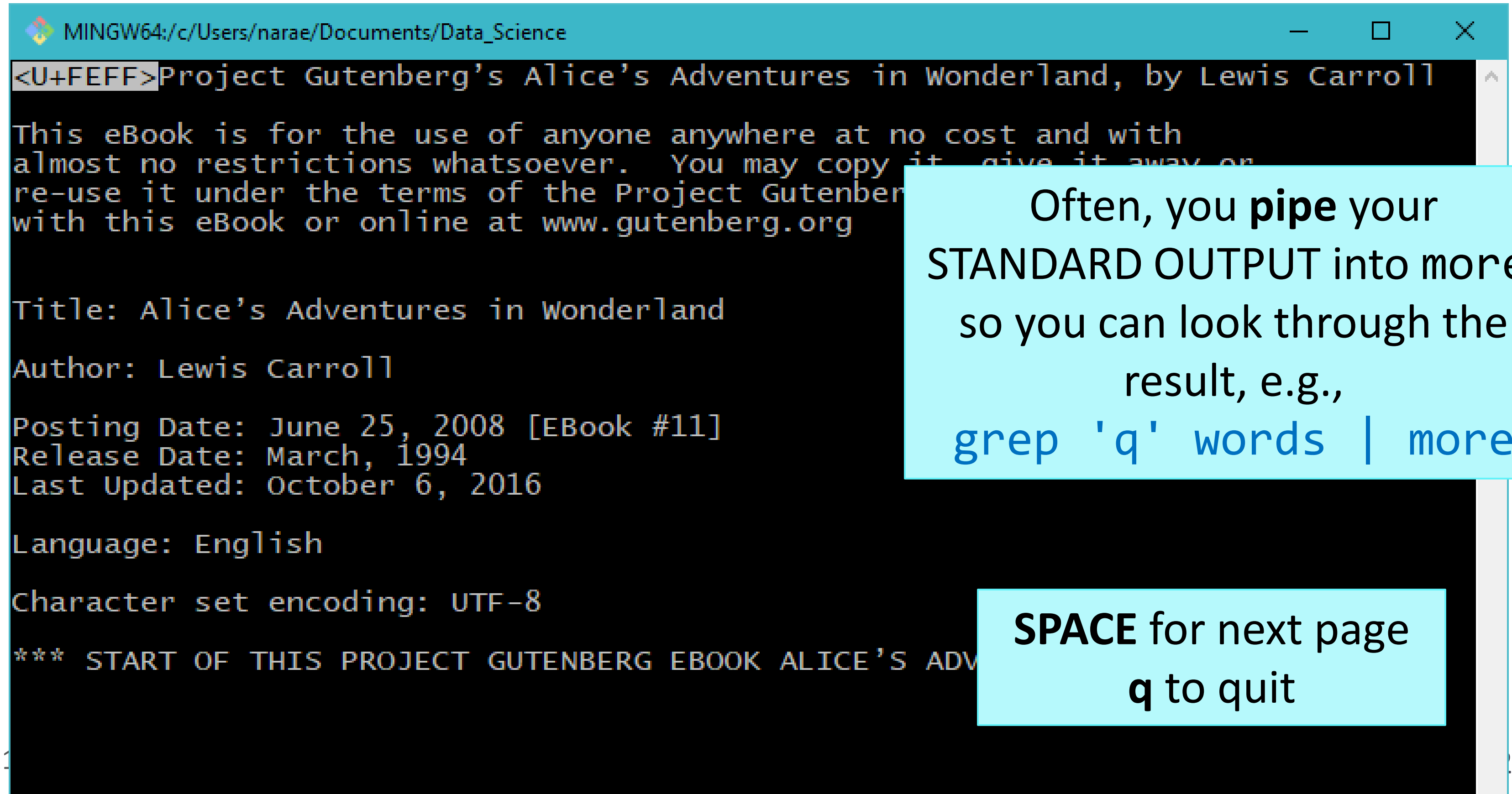
Text processing on the command line

- What you just witnessed is command-line text processing wizardry.
- Computational linguists have long been doing this.
  - Text documents these days tend to have more structure (XML, HTML, etc.) so these tools are less useful, but nevertheless still very handy to know.

- Ken Church, *Unix for Poets*, original document:
  - https://www.cs.upc.edu/~padro/Unixforpoets.pdf
- Christopher Manning (Stanford) brings it up-to-date:
  - https://web.stanford.edu/class/linguist278/notes/278-UnixForPoets.pdf

more or less

- `more` (and `less`) through a text file content, one screen-full at a time. Press **SPACE** for next page, **q** to quit.
  - Windows users: only `less` is available on git bash.



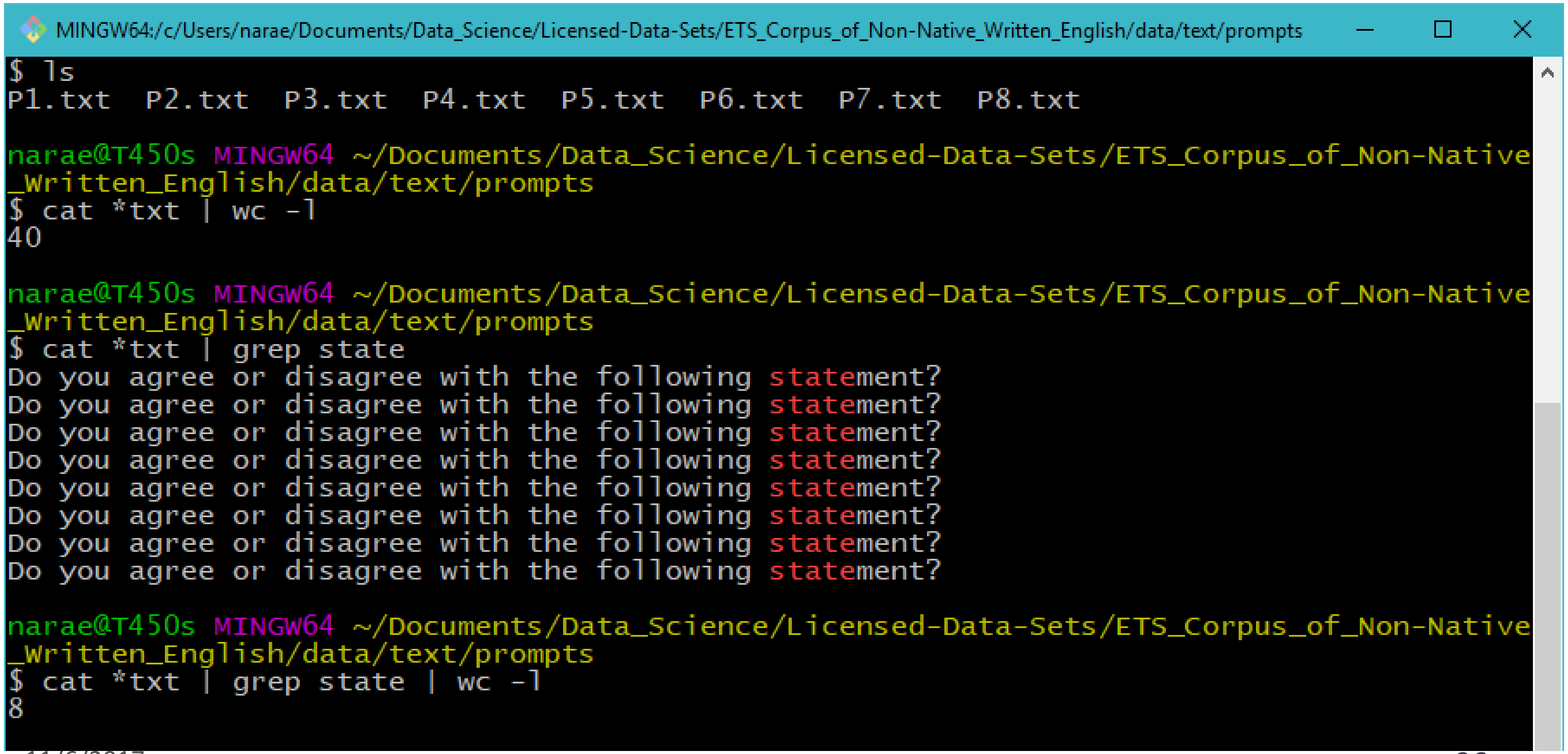
```
<U+FEFF>Project Gutenberg's Alice's Adventures in Wonderland, by Lewis Carroll

This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Project Gutenber
with this eBook or online at www.gutenberg.org


Title: Alice's Adventures in Wonderland

Author: Lewis Carroll

Posting Date: June 25, 2008 [EBook #11]
Release Date: March, 1994
Last Updated: October 6, 2016

Language: English

Character set encoding: UTF-8

*** START OF THIS PROJECT GUTENBERG EBOOK ALICE'S ADV
```

Often, you **pipe** your STANDARD OUTPUT into `more`, so you can look through the result, e.g.,
`grep 'q' words | more`

SPACE for next page
q to quit

cat

- cat concatenates text file content and prints on the standard output.
  - Often used as the first step of piping.



MINGW64:/c/Users/narae/Documents/Data_Science/Licensed-Data-Sets/ETS_Corpus_of_Non-Native_Written_English/data/text/prompts

```
$ ls
P1.txt  P2.txt  P3.txt  P4.txt  P5.txt  P6.txt  P7.txt  P8.txt

narae@T450s MINGW64 ~/Documents/Data_Science/Licensed-Data-Sets/ETS_Corpus_of_Non-Native
_Written_English/data/text/prompts
$ cat *txt | wc -l
40

narae@T450s MINGW64 ~/Documents/Data_Science/Licensed-Data-Sets/ETS_Corpus_of_Non-Native
_Written_English/data/text/prompts
$ cat *txt | grep state
Do you agree or disagree with the following statement?
Do you agree or disagree with the following statement?
Do you agree or disagree with the following statement?
Do you agree or disagree with the following statement?
Do you agree or disagree with the following statement?
Do you agree or disagree with the following statement?
Do you agree or disagree with the following statement?
Do you agree or disagree with the following statement?

narae@T450s MINGW64 ~/Documents/Data_Science/Licensed-Data-Sets/ETS_Corpus_of_Non-Native
_Written_English/data/text/prompts
$ cat *txt | grep state | wc -l
8
```

Wrapping up

- To-Do 10
  - Visit 2 classmates' projects and add your feedback and impression to their Visitor's Log.
- Practice Unix tools and bash shell!
- Work on your project!