

# Lecture 11: Bash Shell & Command Line

LING 1340/2340: Data Science for Linguists

Na-Rae Han

# Objectives

---

- ▶ Finally, shell (bash, zsh)
  - ◆ Running things in command line
  - ◆ Interacting with text files in command line
  - ◆ Regex-based text search using grep

# Bash/Zsh shell

---

## ▶ What is a "shell"?

- ◆ [https://en.wikipedia.org/wiki/Shell\\_\(computing\)](https://en.wikipedia.org/wiki/Shell_(computing))
- ◆ Usually refers to the command-line interface (CLI) as opposed to graphical user interface (GUI).
- ◆ **Bash** is the most common flavor of shell in Unix-like OS.

## ▶ Mac users

- ◆ Mac OS is a Unix-type OS.
- ◆ **Terminal** is a built-in terminal. **Zsh** is the default shell, very similar to bash.

## ▶ Windows users

- ◆ We installed "**git bash**": a bash environment for running command-line git.
- ◆ As a bonus, it came with pretty much all of **popular Unix command-line tools**!

# 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 via editing:

- `.bash_profile`

- `.zprofile`

- ▶ 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

Mac users: color option is not supported by default unless you customize Terminal.

# 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/bin:/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/Anaconda3/Scripts:/c/ProgramData/Anaconda3/Library/bin:/c/Program Files (x86)/Pandoc:/c/Program Files/Intel/WiFi/bin:/c/Program Files/Common Files/Intel/WirelessCommon:/c/Program Files (x86)/Windows Kits/8.1/Windows Performance Toolkit:/c/Program Files (x86)/Python35-32:/c/Program Files (x86)/Python35-32/Scripts:/c/Users/narae/AppData/Local/Microsoft/WindowsApps:/c/Program Files/Intel/WiFi/bin:/c/Program Files/Common Files/Intel/WirelessCommon:/c/Users/narae/AppData/Local/atom/bin:/usr/bin/vendor_perl:/usr/bin/core_perl
```

*1st hit in PATH*

# PATH, which, where

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`

```
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

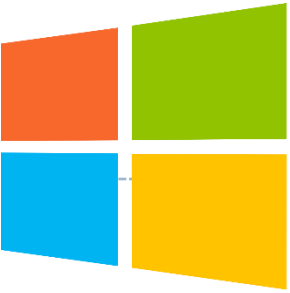
narae@T450s MINGW64 ~
$ which -a pip
/c/ProgramData/Anaconda3/Scripts/pip
/c/Program Files (x86)/Python35-32/Scripts/pip ←

narae@T450s MINGW64 ~
$ echo $PATH
/c/Users/narae/bin:/mingw64/bin:/usr/local/bin:/usr/bin:/bin:/mingw64/bin:/usr/bin:/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/Anaconda3/Scripts:/c/ProgramData/Anaconda3/Library/bin:/c/Program Files (x86)/Pandoc:/c/Program Files/Intel/WiFi/bin:/c/Program Files/Common Files/Intel/WirelessCommon:/c/Program Files (x86)/Windows Kits/8.1/Windows Performance Toolkit:/c/Program Files (x86)/Python35-32:/c/Program Files (x86)/Python35-32/Scripts:/c/Users/narae/AppData/Local/Microsoft/WindowsApps:/c/Program Files/Intel/WiFi/bin:/c/Program Files/Common Files/Intel/WirelessCommon:/c/Users/narae/AppData/Local/atom/bin:/usr/bin/vendor_perl:/usr/bin/core_perl
```

*1st hit in PATH*

# 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*. So if you want Python interactive 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 included:
  - ◆ `more` (use `less` instead)
  - ◆ `man` (you're going to have to Google)



# Mac users

---



- ▶ Add some aliases to your `.zprofile`
- ▶ Like in Windows, you should be able to launch any app that is found in your PATH.
- ▶ Surprise! You also get 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/>

# nano

- ▶ **nano** is a simple command-line based editor. It is found on all Linux distros.
  - ◆ Already present on Macs, and also part of Windows git Bash.

```
MINGW64:/c/Users/narae/Documents/Data_Science
GNU nano 2.9.7 hello.py Modified
#!/c/ProgramData/Anaconda3/python
print("Hello, world!")
print("I'm learning command line.")

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^_ Replace   ^U Uncut Text ^T To Linter ^_ Go To Line
```

# 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):

`#!/systempath/to/python`

- ◆ In my case, it's `#!/c/ProgramData/Anaconda3/python`
- ◆ If your path contains a SPACE... tough luck! (Just kidding, there are ways around it.)

# Wrapping up

---

- ▶ Progress report #2 due on Thu!
- ▶ Next class
  - ◆ More command line, grep, bash shell scripting
  - ◆ Supercomputing at CRC