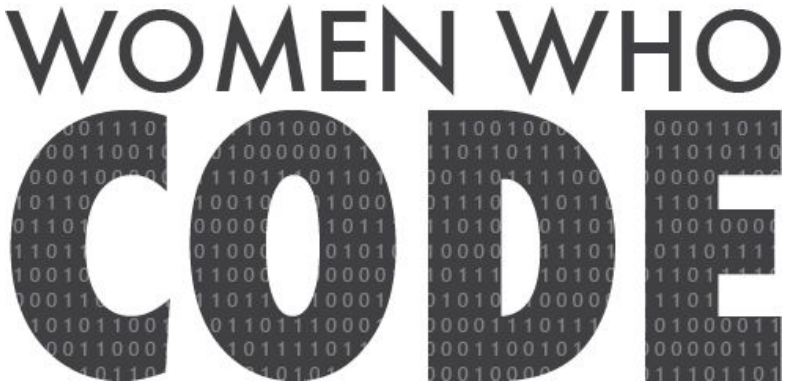

INSPIRING women to EXCEL in technology careers

@WomenWhoCodeNYC

THE MISSION

WWCode is a **global** nonprofit dedicated to inspiring women to **excel** in **technology** careers.

> *"We connect amazing women with like-minded women leaders around the globe who unite under one simple notion – the world of technology is much better with women in it."*
>
> -Alaina Percival, WWCode CEO

20 Countries, 60 Cities + Expanding

Over 10,000 members in NYC alone!!

Our Community is a network of

100,000 technical women

New York: womenwhocode.com/nyc

CODE OF CONDUCT

Women Who Code (WWCode) is dedicated to providing an **empowering experience for everyone** who participates in or supports our community, regardless of gender, gender identity and expression, sexual orientation, ability, physical appearance, body size, race, ethnicity, age, religion, socioeconomic status, caste, or creed. **Our events are intended to inspire women to excel in technology careers, and anyone who is there for this purpose is welcome**. Because we value the safety and security of our members and strive to have an inclusive community, **we do not tolerate harassment of members or event participants in any form**. Our Code of Conduct applies to all events run by Women Who Code, Inc. If you would like to report an incident or contact our leadership team.

https://www.womenwhocode.com/codeofconduct

WOMEN WHO CODE

“Flatiron Health is a healthcare technology and services company focused on accelerating cancer research and improving patient care.”

We’re hiring (lots of different roles!)

flatiron.com/careers

Follow us! twitter.com/flatironhealth

THANK YOU!

Meetup: WomenWhoCodeNYC

Twitter: @WomenWhoCodeNYC

Website: womenwhocode.com/nyc

Learn to bash!

Sam Bail (Twitter: @spbail)
Women Who Code workshop, June 2019

Intro!

- I’m a Data Insights Engineer at **Flatiron Health**
  - We’re hiring :)
- **Plan for today:**
  - Some background and terminology
  - Basic navigation, file manipulation, searching
  - Environment variables and .bash_profile
  - Your first bash script!
- Follow along by typing the **bold commands**
- The DIY blocks are mini-exercises!

Background & Terminology

What are all these words?

- **Shell** = user interface (command-line or graphical)
  - Usually shell = command-line user interface
- **Bash** = a type of shell (“**B**ourne-**A**gain **SH**ell”)*
  - There are others: zsh, ksh…
  - macOS is switching from bash to zsh in 10.15
  - Don’t worry, the basics are all the same :)
- **Terminal** = the program than runs a shell

* Read the Wikipedia pages on “Unix shell” and “Bash (Unix shell)” for some history

Why use the terminal/shell/bash?

- **It's often faster or more flexible**
  - Bulk operations on files
  - Searching for files/text in files
  - ...
- **Sometimes you have to**
  - Connecting to a server via ssh
  - Setting aliases or environment variables
  - ...

Navigating the file system

Let's get started!

- Some basics for orientation:
  - **pwd =** path of current working directory
  - **ls =** list files in the directory
  - **ls -l =** add a "long format" flag
  - **ls -la** = add long format + "all" flag
- How do we know what flags are available?
  - **man ls =** man(ual) pages
  - **<space bar>** = navigate down
  - **q** = get out of man!
- DIY: check **man** and try out different flags for **ls**

Some helpful shortcuts

- **clear** = refresh your terminal window
- **<Up arrow>** = cycle through previous commands
- **<Tab>** = bash will tab-complete file/dir paths
- **<CTRL+c>** = exit pretty much any process
- **<CTRL+a>/<CTRL+e>** = go to start/end of line
- **<CTRL+w>** = delete word before the cursor
- **<CTRL+r>** = incremental reverse search through previous commands, keep hitting **<CTRL+r>** to cycle
- DIY: use **<CTRL+r>** to find and run your first **pwd**

Navigating the file system

- **cd (directory)** = go to (directory)
  - Use **pwd**, **tab-complete** and **ls** for orientation!
- Example:
  - **pwd**
  - **ls Documents**
  - **cd Documents**
  - **pwd**
- ~ = your home directory (/Users/<name>)
- . = the current directory
- DIY: go back to your home directory using ~

Basic file operations

File operations (1)

- **touch (filename)** = create an empty file
  - **touch hello.txt**
- **mkdir (directory)** = make a directory
  - **mkdir some_dir**
- **open (filename)** = open file in default text editor
  - **open hello.txt**
  - Type some text and save the file
- DIY: create a directory “wwc_workshop”, cd into it, and create a file hello.txt
  - Open hello.txt, write some random text into it, save

File operations (2)

- **cp (file1) (file2)** = copy file1 to file2
  - **cp hello.txt hello2.txt**
  - **ls**
- **mv (file1) (file2)** = rename file1 to file2
  - **mv hello2.txt hello3.txt**
  - **ls**
- **rm (file)** = remove file (irreversibly!)
- Use wildcard * in filenames for any operation
  - **ls hello*.txt**

Looking at file content (1)

- Download the **permits.csv** file from here:
  - https://github.com/spbail/bash-workshop
  - Hit download, then “save as” into your **wwc_workshop** directory
  - This is an NYC OpenData file with locations and types of film permits in the city (pretty cool!)
- Make sure you’re in the right directory
  - **cd ~/wwc_workshop**
  - **ls** (should show permits.csv)

Looking at file content (2)

- **cat** = print content of file
  - **cat permits.csv**
- **head** = cat, but only show the top n rows of a file
  - **head permits.csv**
  - **head -n 2 permits.csv**
- **tail** = like head, but shows bottom n rows
  - **tail permits.csv**
  - **tail -n 2 permits.csv**

Looking at file content (3)

- **less** = interactive text reader
  - **less permits.txt**
  - Navigate through pages with **<space bar>**
  - Exit with **q**
  - (What does this remind you of?)

Searching Text in files

Finding text in files (1)

- Option 1: **less** has a built-in search
  - **less permits.csv**
  - **/** (this gets you the search prompt)
  - **film (enter)**
    - This will say “Pattern not found”
  - Type **-I** to make the search case insensitive
  - Try searching again!
  - **p/n** = jump to previous/next occurrence of “television”

Finding text in files (2)

- Option 2: use **grep**
  - **grep (search word) (file)**
    - **grep film permits.csv**
      - No results! grep is case sensitive, too!
    - **grep -i film permits.csv**
- **grep** returns the line with the search word
  - Works best with line-based files, e.g. CSV
- DIY: grep for the word “theater”

Chaining commands using a Pipe

- **wc -l (file)** = count lines in file
  - **wc -l permits.csv**
- The pipe **|** chains commands, e.g
  - **grep -i film permits.csv | wc -l**
  - This pipes the output of **grep** as input to **wc**
- Pipes can be chained: cmd_a | cmd_b | cmd_c…
  - Output of cmd_a = input to cmd_b
  - Output of cmd_b = input to cmd_c
  - etc.

DIY: Mini-Project 1

- Go to the directory with permits.csv
- Look at the file using **head, tail,** and **less**
- Get some counts using **grep** and **wc**:
  - How many lines are in the file?
  - How many lines contain the word “Television”?
  - How many lines contain the word “Television” and your own ZIP code? (Hint: remember you can chain pipes!)
- **Take a break :)**

Finding files

Finding Files

- **find (directory) -name “(search word)”**
- I mostly use it to find filenames, e.g.
  - **cd ~/wwc_workshop**
  - **find . -name “per*”**
  - **find . -iname “Per*”** (case insensitive)
  - **find . -name “*.csv”** (search for file type)
- You can also specify a directory to search in:
  - **find ~/wwc_workshop -name "*.csv"**

Environment variables and .bash_profile

Environment variables

- Environment variables are system-wide global variables that your shell knows about
- **env** = show all environment variables
- You can access any variable using the $ sign
  - **echo $USER** (your username)
  - **echo $HOME** (your home directory)
- We also just learned about **echo** (print statement)
  - **echo “hello”**
  - **echo “hello $USER”**

SETTING AN ENVIRONMENT VARIABLE

- **export (varname)=(value)** = sets an environment var
  - **export myvar=42**
  - Spaces and quote types matter in bash!
  - **echo $myvar**
- Setting environment vars is often done for software to know where to find stuff, e.g. $PYTHONPATH
- DIY: open a new terminal tab with **<CMD+t>** and check if it knows about myvar

Your .bash_profile (1)

- Every terminal window/tab only knows about its own environment variables
- Variables that are needed everywhere need to be exported at startup
- Your terminal reads a “profile” file at startup
  - ~/.bash_profile
  - ~/.profile
  - ~/.bashrc
- We’ll be using ~/.bash_profile, feel free to read up on differences :)

Your .bash_profile (2)

- Let's export a new environment variable
  - **open ~/.bash_profile**
  - Add this line: **export anothervar="hello"**
  - Save the file
  - **echo $anothervar**
- Your open shell doesn't know about changes to .bash_profile automatically!
  - Either: open a new terminal tab
  - Or: **source ~/.bash_profile**

Your first bash script!

Bash scripting 101

- Instead of typing commands, you can also add them to a file and execute the file
- There are several options to execute a script:
  - **./samscript.sh**
  - **bash samscript.sh**
  - **sh samscript.sh**
- DIY: create a new file **myscript.sh** and add a line to echo “Hello $USER!”, save the file, and execute
  - You’ll get a “permission denied” error - **stop here!**

FILE PERMISSIONS 101

- Files in your file system have permissions
  - **ls -l myscript.sh**
- The permissions are as follows:
    - R=read, W=write, X=execute, -=no permission
    - 3-letter-blocks each for owner/user, group, other
- In order to execute a script, it needs to be set to “executable” (duh.) at least for yourself (user)
- **chmod** changes the “mode” (permissions) of a file
  - **chmod u+x myscript.sh**
  - **ls -l myscript.sh**

DIY: Mini Project 2 (optional)

- Create a new bash script
- Add at least 2 commands you've learned
- Save the file
- Set the right permissions
- Execute the script
- **Congratulations, you're done for today!**

Wrap-up

- We've covered:
  - Directory navigation and file manipulation
  - Searching for text and finding files
  - Environment variables and .bash_profile
  - Your first bash script
- Other fun bash stuff to look up (e.g. man):
  - The **sudo** command
  - Setting an **alias** (shortcut) for bash commands
  - Using **vim** as text editor
  - Other ways to exit and kill processes (**CTRL+z**, **CTRL+d**…)
  - Utilities like **cal**, **date**, disk space checking, **cowsay**...

UPCOMING EVENTS!

Python and PuLP workshop @ Dataminr 6/13

Algorithms @ Betterment 6/18

React Workshop @ Grace Hopper Academy 6/19