Jo O’Harrow (they & she) is a mathematician and programmer living in Corvallis Oregon. They can be found programming in various coffee shops, running in the research forests, and gardening on nice days :)

**Email me with questions!**
josieoharrow@hotmail.com

# HACKING FOR SOCIAL JUSTICE

## ... FOR BEGINNERS

### ... (LEGALLY)

**What kind of hacking?** Scripting refers to the automation of tasks using computer code. This is a tutorial-based introduction to obtaining, reformatting, and scrubbing data, all of which is "scripting", using Ruby. Ruby is a programming language that is excellent for scripting, especially when the work you want to automate involves parsing and processing large text-based datasets.

Aside: You may have heard of "shell scripts" before. These are as cool as they sound! Using a shell script, you can have your computer automatically execute command line operations. We won't be writing any for this tutorial, but please reach out if you are interested in learning how to shell script :)

**Who am I?** I have worked as an automation engineer for several years, and in the last six months I have worked as a legislative advocate. In this role, I have applied my automation expertise to use scripting to obtain data on legislator funding, upcoming committee meetings, and representative redistricting. I have the background to make this tutorial, but **YOU DON'T NEED ANY TECHNICAL EXPERIENCE** to follow along. If computers and "code" are scary, that's okay! By doing this tutorial, you will have an invaluable tool to speed up manual data processing.

**This sounds... Illegal.** It isn't! There were two main types of scripting we will cover: data scraping, and data processing. Data scraping involves retrieving information from a webpage, and data processing is a blanket term I will use to describe anything from reformatting to cleaning to analyzing data. Data scraping is protected by the supreme court. **It is completely legal** to scrape data from websites. That said, you may be violating a website's terms and conditions if you create an account to have access to data on the webpage. If you do that, you need to be more careful what you do with the data afterward, but it is still not illegal to scrape data from them.

The reason this is an important skill for activists is not so that someone can use scripting to illegally steal and distribute data, or perform automated attacks on corrupt institutions like the police bureau. While scripting and automation is a tool that is used in those settings, it is not the use of automation that is illegal. Even expert computer scientists get caught illegally hacking and the consequences are steep. That said, **every activist can and should hack for the greater good!** Activists pushing against the status quo often have to deal with purposefully obfuscated datasets. In addition, they may need data on important issues that is not published or readily accessible but may be obtained by scrubbing various websites or datasets. Even when it is possible to do this work by hand, it may be a huge waste of time spent on a task that with a small amount of scripting experience could be completely automated.

**Here are some important safety notes:**

1) If you aren't sure if something is legal, don't do it.
2) Don't talk to cops.
3) If you get a warning message, stop what you are doing but do not panic. It was most likely harmless and legal.

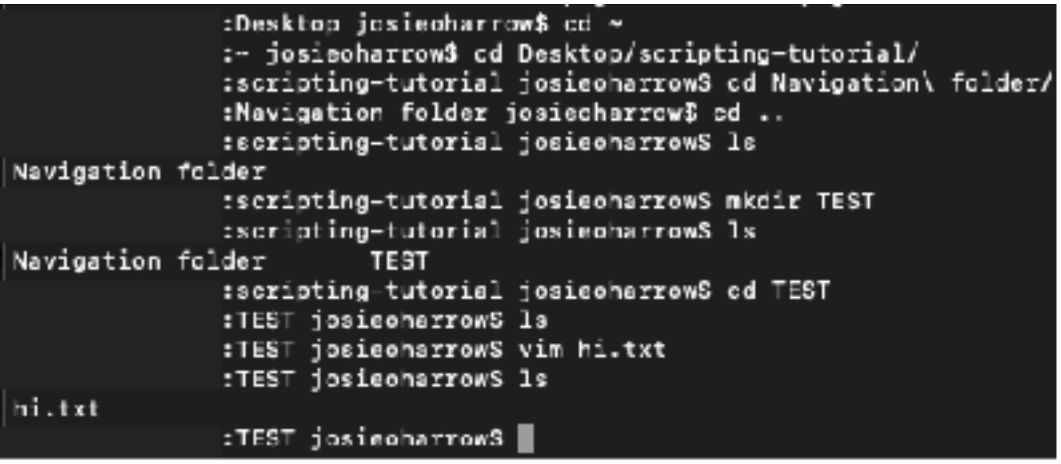
Having handled the precautions, let's script! To get started...

**1.** Find or install your **command line** of choice, and make sure you can run git commands from it. You can do this with any OS, but I will include more steps for Mac/Windows users.

Windows:
Install Git Bash. This is a command line interface that is more consistent with the Mac/Linux command lines. You could use a built-in environment, but I will assume for this tutorial you are using Git Bash and I will refer to Git Bash as your command line from here on out.

Mac:
Open Terminal. This is your command line! Type “git —version”, hit enter, and follow the prompts to install.

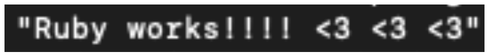
**2.** Open your command line, and get familiar with the basic commands. A good cheat sheet is available at: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Understanding_client-side_tools/Command_line. Try “cd”, “pwd”, and “ls”. When “cd”-ing, tab will auto-complete the folder you are trying to type. You don’t need to know this, but it is handy :)
Once you can navigate comfortably (ish) using “cd”, go ahead and move on.



```
              :Desktop josieoharrow$ cd ~
              :~ josieoharrow$ cd Desktop/scripting-tutorial/
              :scripting-tutorial josieoharrow$ cd Navigation\ folder/
              :Navigation folder josieoharrow$ cd ..
              :scripting-tutorial josieoharrow$ ls
Navigation folder
              :scripting-tutorial josieoharrow$ mkdir TEST
              :scripting-tutorial josieoharrow$ ls
Navigation folder       TEST
              :scripting-tutorial josieoharrow$ cd TEST
              :TEST josieoharrow$ ls
              :TEST josieoharrow$ vim hi.txt
              :TEST josieoharrow$ ls
hi.txt
              :TEST josieoharrow$ 
```

**3.** Download my code by "cd-ing" to your desired location, and then running "git clone git://github.com/josieoharrow/legislative.git"

**4.** Download Ruby. To test if you did this correctly, "cd" into the scripting-tutorial folder. Run "ruby test-ruby.rb" in the command line. The "ruby" command tells your computer to use your ruby installation to execute whatever ".rb" file follows it. You should get the following message:



```
"Ruby works!!!! <3 <3 <3"
```

Now you have a very basic scripting environment set up! Congrats :) If you want to play around with Ruby, type "vim test-ruby.rb" to open the contents of test-ruby.rb. Here are the basic vim commands! There are more you can learn, but this will do for now :)

* To edit the code, type “i”. Use the arrows to navigate and change text.
* To stop editing: “[escape]”
* To save changes “:w”
* To quit, type “:q”
* To quit without saving, type “:q!”.

**5.** Try changing the text in the quotation marks using vim. Then, quit vim and run “ruby test-ruby.rb” again. You should see your updated quote in the output!

**At this point, you are officially programming!!!**

## Data “Finding”

If you are working with a large text file, you might need to look for more specific data. We are going to try a few examples finding patterns on the text published at: https://theanarchistlibrary.org/library/andie-nordgren-the-short-instructional-manifesto-for-relationship-anarchy. I saved the text into “data.txt”.

“cd” into Data-Finding.
Run “ruby script.rb”. This will output the number of times “love” appears in the text.

**Goal #1:** Update my regular expression in script.rb to include “Love” in the search. You should find 13 total occurrences of “love” or “Love”.
**Goal #2:** Write code to count the number of times “ea” appears in the text.
**Goal #3:** Use a capture group (parenthesis) to find all of the words that come after “love” in the text. You will need to use a more advanced regular expression for this that includes a whitespace character and a quantity modifier. Instead of printing the length, print the scan result itself!

**Resources**
I use regex101.com to test my regular expression patterns. If you are very stuck, I have included an answers file with regular expressions to use. A different project that I used regular expressions for is in the “Scripting Example” folder. You can see how I used the “match” method, which is similar to “scan” and is also built in to Ruby. Also, don’t be ashamed to Google and use Stack Overflow to figure this out. That’s 99% of programming!

## Data Formatting and scrubbing...

Data formatting and scrubbing are closely related, and overlap a lot. Scrubbing means cleaning up the dataset, while formatting means changing the underlying structure for the data. We are continuing with the same data as before in a new folder. “cd” into Data-Formatting.

Run “ruby script.rb”. This will make a file, no-spaces-data.txt, that is the same data with all of the spaces removed. If we think of our dataset as a bank of characters and we don’t care about the spaces, this is an act of **scrubbing**.

**Goal #1:** Change the output file to be “anarchy.txt”. In the replace line, update the empty string to be the word “ANARCHY”. This will make a file where the spaces in our data are replaced with the word anarchy. Because “ANARCHY” is not used anywhere else in the text, this is a discernible pattern of our own invention. We have not scrubbed the data, but **reformatted** it according to our new syntax.

**Goal #2:** Use Ruby “split” to turn your string of text into an array. Split the text on the word “love”, and print the result. This will remove “love” from the final product.

**Goal #3:** Save that array into a CSV using the Ruby CSV library. You will need to include “csv”! This may take trial and error, but once you can save a CSV from a text file, **you can do anything**.

# Data scraping…

We will use two scraping techniques for this tutorial.

First, we will try **HTTP requests**.

From the scripting-tutorial folder, “cd” into “HTTP-GET”. This code gets the contents of a webpage, using its url, through an HTTP request. Run “ruby script.rb”. This code will print out the contents of a hard-coded URL. It will also ask if you want to try another webpage. Say YES and try entering a simple web page, like: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git. The result will be the source code for the website, so it will be written in HTML.

Now, open the code by running “vim script.rb”. There are lots of comments in it! Your first big project is to update the code using online documentation to save the output of a GET request into a text file. Try googling “Ruby write to file”. Don’t forget to include the file library in your code!

Your second project is to use regular expressions to count how many committee meetings Sara Gelser Blouin has. You will need to first GET the oils page, and then run a scan on the result.

# Data scraping…

## *Browser Automation*

**Browser automation** is the second scraping technique. You can use Selenium to automate things through the browser itself. Assuming you are running on Mac or Windows, follow these steps to set up:

1. Update Chrome
2. Go to https://chromedriver.storage.googleapis.com/index.html and download the latest chromedriver (it will be listed under LATEST_RELEASE). Put the exe somewhere you are able to find it again. If you are using my repository, feel free to replace the (probably outdated) chromedriver.exe under "legislative".
3. Run chromedriver.exe by double clicking. You may need to right click, say open, and verify that you want to run it. This is a safe download, so that’s fine.
4. Update the location of chromedriver.exe in scrub.rb to be the location you downloaded chromedriver.exe.
5. Run "gem install selenium-webdriver" and "gem install ffi"
6. Run scrub.rb. This is an example of data scrubbing I did on Comcast’s leadership page.

To develop your own browser automation, you will need to be comfortable with HTML. The automation library, selenium, needs help finding elements before it can click on them, read them, or interact with them in other ways. This is beyond the scope of this zine, but now that you have gotten a taste of Browser automation and you know what you could do with it, I encourage you to keep learning! There are so many online resources for this, and you can always email me :)