Friday, Feb 10, 2023
Lecture #11
MSSC 6000

Announcements

* HW 2 was assigned Wednesday. Deadline changed to Wed, Feb 22 (two extra days)

Lecture 4 – Unix Commands (continued)

(8) cat [filename] – prints a whole file to the terminal

(9) head [filename] – prints the first 10 lines of a file

(10) tail [filename] – prints the last 10 lines
"-n" to change from 10 to something else
head -n 20 [file]

(11) less [filename] -opens the file in the terminal, but in a way where you can scroll, NOT edit, and quit

"q" to quit

You can do anything in a terminal.

(12) nano [filename] - full text editor inside of the terminal. Has keyboard shortcuts to do most things.

(13) touch [filename] - creates a blank file with that name

People write whole programs ("bash scripts") with these terminal commands.

Ex: Search five_letter_words.txt for words with no vowels.

```
with open("five_letter_words.txt", "r") as f:
    words = f.readlines()
print([w for w in words if
        not any(l in w for l in
                ["a", "e", "i", "o", "u"])])
```

Lecture 4.5 - The Coding Process

Hardest Approach:

$$\begin{tikzcd}
\boxed{\text{read problem}} \arrow[r] & \boxed{\text{think really hard}} \arrow[d] \\
& \boxed{\text{start coding}}
\end{tikzcd}$$

Too many steps in your head

Better process:

1) Read the problem.
2) Think about the problem.

3) Do some examples by hand to see if you understand the problem.
(e.g. longest collatz sequence,
$20 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$
length 8 chain)

4) Think about how you might solve it. Think of an algorithm.
What steps did you do when you did it by hand in (3)?

5) Write, on paper and in English, the steps of your algorithm from (4).
"Pseudocode"

<u>Ex for Collatz</u>:

```
set longest_chain = 0
set longest_num = 0
loop over "num" from 1 to 1 million:
    compute the length of the chain
                            for num

    if length > longest_chain:
        longest_chain = length
    longest_num = num
answer is "longest_num"
```

now write pseudocode for this part

this makes us think that we could have a function for this.

6) Start coding!

As you code:

7) "Rubber Ducking" - talk to a rubber duck, out loud, explaining what you're doing as you write each line of code

8) Pause often to test a few lines of code at a time before writing more.
   * Do these lines of code do what I think?
   * Is your loop looping over the

right thing? (print "num")
* Does the list you just built contain the things you think it does?

If it's not working:

9) Debug it! Think of small test cases. (1 to 10 instead of 1 to 1M). Add in tons of print statements. Run it and see where something unexpected happens.

When you think it's working:

10) Test it! Take the small examples from (3) and use them as input. Does the code run or give an error? Does it take way longer than expected? Does it give the right answer?