Open Source Software Team

Intro to VIM Motions 🔥🔥

February 29th, 4-4:30PM CS 401

Launch into the vast expanse of collaborative development, exploring new frontiers and innovating together!

Self link: acmcsuf.com/vim-intro

Open Source Software team

Meeting 2/22 recap

Last week's slides: acmcsuf.com/winning-fh

<!-- Previously, we talked all about hackathons and ways to prepare for them. Thanks to everyone, **Fullyhacks** was a **MASSIVE** success! Quick reminder once again, you can come to **any ACM OSS** meetings **regardless** of the previous one! -->

REVIEW SYNC NOTES!!!
(acmcsuf.com/oss-sync)

Who are we?

Ethan (etok.codes)

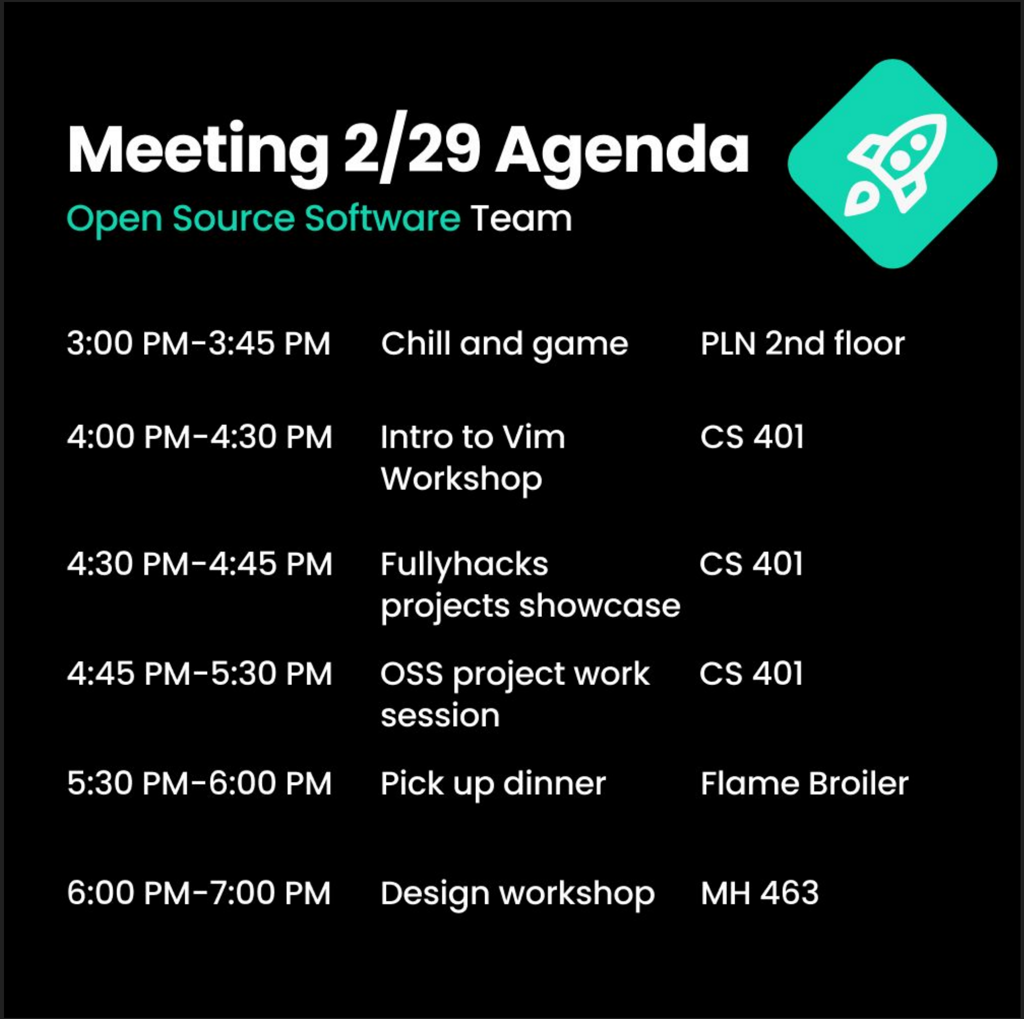
Karni (karni.codes)

Tomas (tomasohCHOM)

Evan (EvanCPSC)

Owen (JOwen-ster).

Oscar (oscisn93)



Meeting 2/29 Agenda

Open Source Software Team

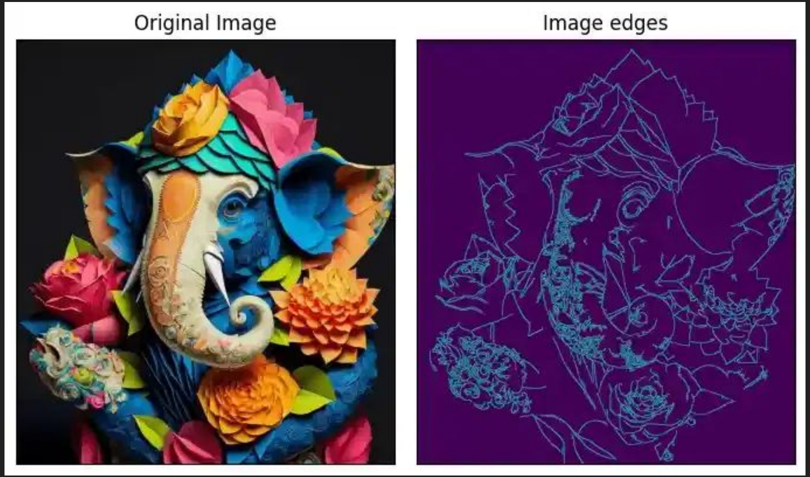
3:00 PM-3:45 PM	Chill and game	PLN 2nd floor
4:00 PM-4:30 PM	Intro to Vim Workshop	CS 401
4:30 PM-4:45 PM	Fullyhacks projects showcase	CS 401
4:45 PM-5:30 PM	OSS project work session	CS 401
5:30 PM-6:00 PM	Pick up dinner	Flame Broiler
6:00 PM-7:00 PM	Design workshop	MH 463

Library of the week: Pillow

The Python Imaging Library adds image processing capabilities to your Python interpreter.

This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.



Intro to VIM

What is VIM?

1. **Free** and **open-source** text editor created in 1991 by Bram Moolenaar
2. Built on top of **VI** ("VI Improved" == VIM)
3. **Lightweight** and **Extensible**
4. Comes with its own set of **keybinds**

Why use VIM?

It’s FAST! 🔥🔥🔥

- ➔ Eliminates dependency from the mouse
- ➔ Keybinds are intuitive and convenient
- ➔ Hands always on the keyboard gives productivity boost

`ci{`, `o`, `viW`, `f`, `Ctrl-D`, `*`

<3

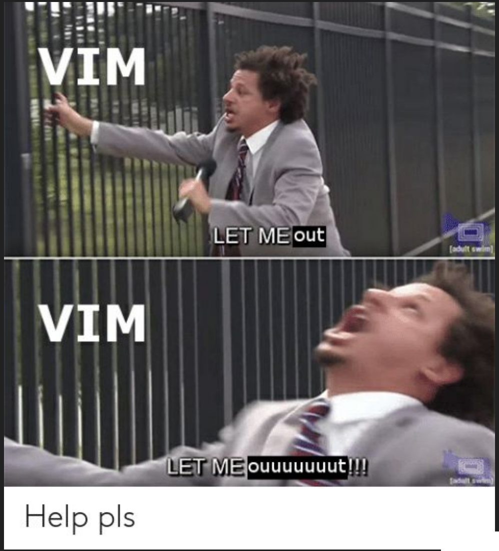
It’s FUN!

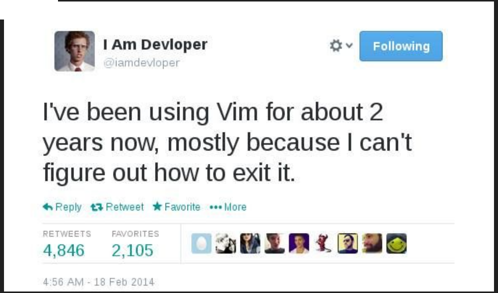
Feels sort of like a fighting game, figuring out the best and most comfortable combos.

It’s an extra tool you can LEARN!

- **VIM** is just an extra tool that you can learn to enhance both productivity and DX
- Expands your skill set
- Challenging, but extremely **rewarding**
- Increase your confidence to learn anything in software

“I use vim btw” & memes



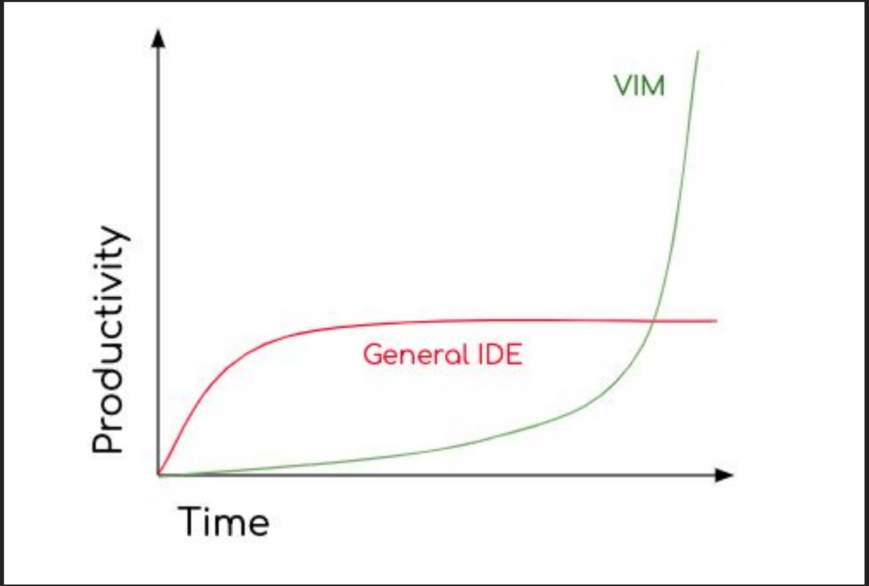

I Am Devloper
@iamdevloper

I've been using Vim for about 2 years now, mostly because I can't figure out how to exit it.

Cons of VIM

- Steep learning curve
- Being comfortable with using the keyboard
- Rewards after a while of using the tool

Note: It is less about whether it is difficult or not, and more about how much YOU are willing to learn the tool. Doesn't only apply to VIM, but anything in programming. YOU get to decide whether you want to learn it or not.



DEMO TIME!

Installing VIM

VIM *should* be preinstalled.
If not run:

```
$sudo apt install vim
```

WSL then same instructions as Linux OR install it via:
www.vim.org/download.php
(Make sure to to TOGGLE "Create .bat files")

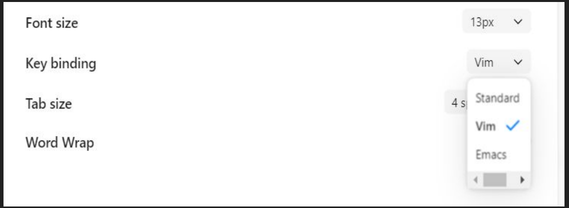
Install Homebrew, then:

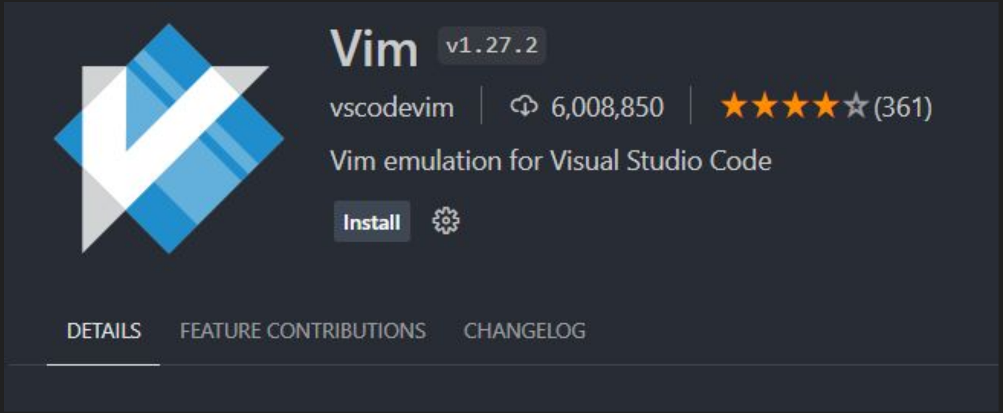
```
$ brew install vim
```

(May already be installed)

You can also use VIM in...

Enable the VIM VSCode Extension and you are good to go! (you can disable it after the demo if you wish so).


Font size 13px
Key binding Vim
Tab size 4 s
Word Wrap
Standard
Vim ✓
Emacs


Vim v1.27.2
vscodevim | 6,008,850 | ★★★★☆ (361)
Vim emulation for Visual Studio Code
Install
DETAILS FEATURE CONTRIBUTIONS CHANGELOG

You can solve Leetcode problems while practicing VIM Motions! Settings > Code Editor > Key Binding

To get started with the Demo

Go to https://github.com/tomasohCHOM/vim-intro-workshop and clone the repository onto your local machine.

1) Open the terminal and `cd vim-intro-workshop`, then `vim .` will open the VIM file tree. Locate demo.txt and press Enter to open it.

OR…

2) Open the folder/directory in VSCode (with the VIM extension).

Basic VIM Info.

VIM is often regarded as three things:

1. **Keyboard** editor

` 1 2 3 4 5 6 7 8 9 0 - = Backspace
Tab Q W E R T Y U I O P [ ] \
Caps Lock A S D F G H J K L ; ' Enter
Shift Z X C V B N M < > ? Shift
Ctrl fn Alt Space Control Alt

2. **Modal** editor

3. **Motion-based** editor

Basic movements

The most basic movements in VIM are: `h, j, k, l`:

- `h` goes to the **left**
- `j` goes **down**
- `k` goes **up**
- `l` goes to the **right**

```
          k
← h      ↓↑      l →
          j
```

If hard to remember, think of `h` and `l` as the leftmost and rightmost keys. `j` is “jumping DOWN” and `k` is “kicking UP.”

Anything that moves the cursor is called a **motion**.
You can also move with `w` (by words) and `b` (by words BACKWARDS).

COMMAND COUNT MOTION

The specific command to be executed (`v, c, y, d`)

The number of times that the motion will execute

How the cursor will move (`h, j, k, l, w`)

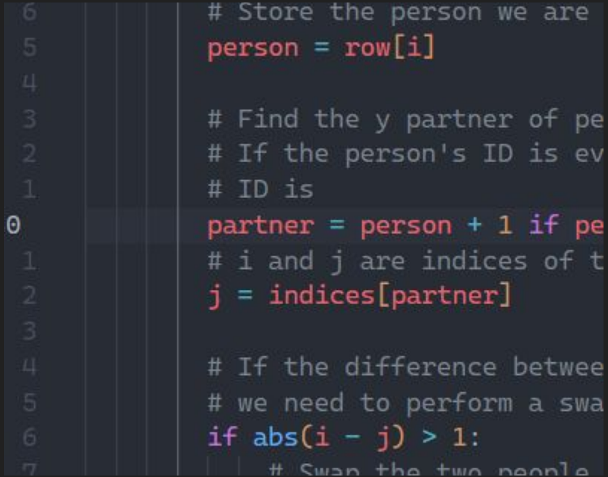
Count Motions & Relative Line Jumping

We can perform a certain motion however many times we want to by prepending a “count” to our motion. For example:

> If we want to go twelve lines **UP**, rather than going like: **kkkkkkkkkkk...**
> ...we can simply do: **12k**

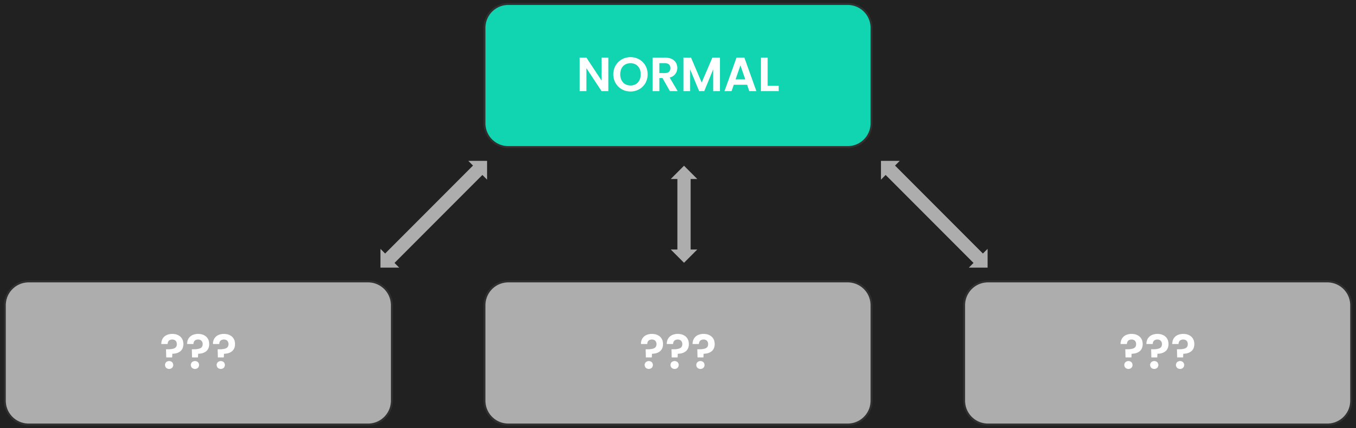
It allows to jump between lines *really* fast with **relative line jumping**.

Works with any other motion. Can also be used with other **commands** too (We will come back to this in a bit).



```
6        # Store the person we are
5        person = row[i]
4
3        # Find the y partner of pe
2        # If the person's ID is ev
1        # ID is
0        partner = person + 1 if pe
1        # i and j are indices of t
2        j = indices[partner]
3
4        # If the difference betwee
5        # we need to perform a swa
6        if abs(i - j) > 1:
7            # Swap the two people
```

VIM is a MODAL Editor



NORMAL mode:

- VIM's default mode. Move around and do actions like **yy** or **dd**.
- Works as the "pivot" to go from NORMAL to other modes.

Some more actions in NORMAL

A couple more things that we can do while in NORMAL mode:

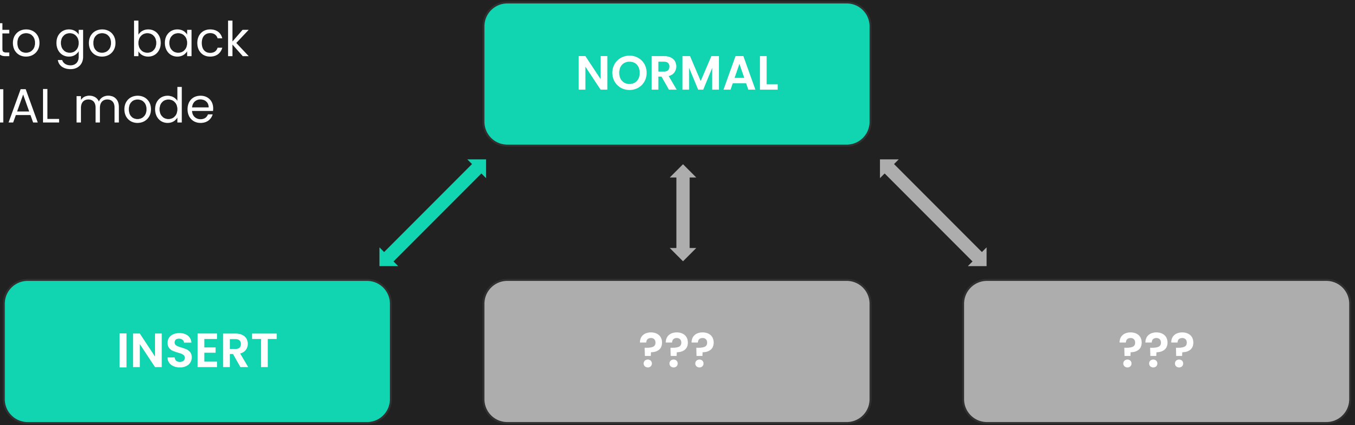
- `y` is the action to copy. `yy` copies the entire line.
- `d` is the action to delete (also copies). `dd` deletes the entire line.
- `p` to paste the contents from clipboard.
- `x` deletes a single character.
- `u` to undo the last command (`Ctrl-R` to redo).

Append a motion to perform an action along the area where the cursor moves. E.g., `dw` deletes a single word.

Like with other motions, you can perform any of these actions x amount of times by prepending a **count**. E.g., `3dd` deletes three lines.

VIM is a MODAL Editor: INSERT

Use `Esc` to go back to NORMAL mode



INSERT Mode:

- VIM’s mode to insert text. Most straightforward.
- Edit text in INSERT Mode like in any other text editor.

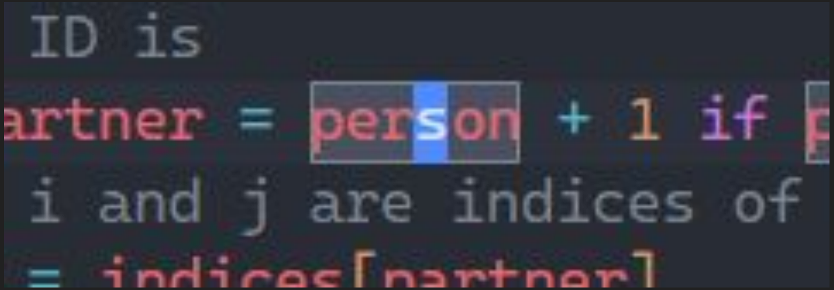
INSERT mode

`i` enters INSERT mode from the left side of the THICK cursor.
`a` enters INSERT mode from the right side.

Like any other text editor/IDE, we can add text to our file (move around with arrow keys).

Press `Esc` to go back to NORMAL.

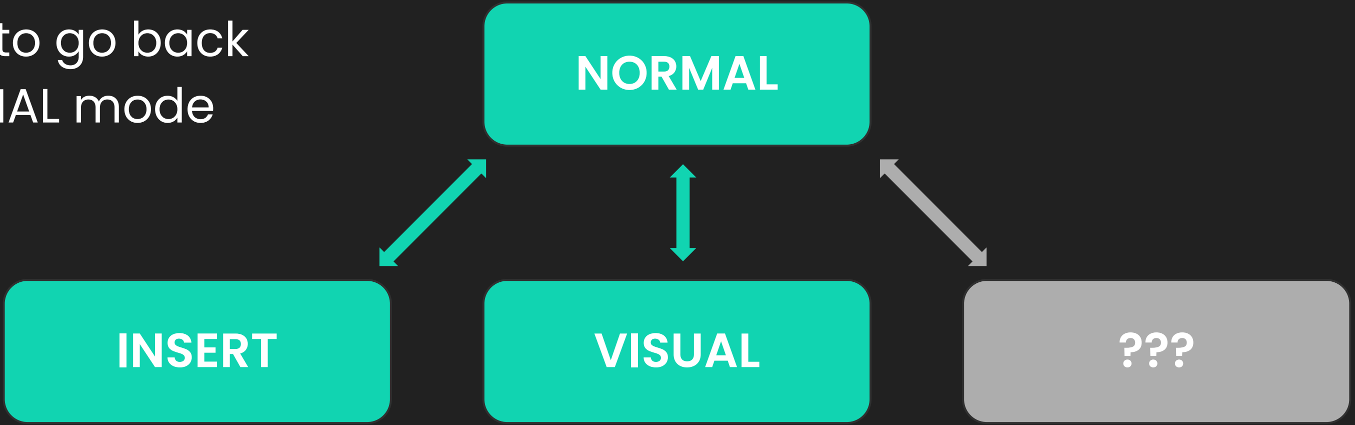
Where do you think I will end up when I press `i`? What if I do `a` instead?



```
 ID is
artner = person + 1 if p
 i and j are indices of
 = indices[partner]
```

VIM is a MODAL Editor: VISUAL

Use `Esc` to go back to NORMAL mode



VISUAL Mode:

- VIM's mode to highlight text. Perform motions like in NORMAL mode while highlighting those areas.

VISUAL mode

`v` enters VISUAL mode. Start moving around and see how text is being highlighted!

Press `y` to copy the contents (yoink lol).
Press `d` to delete the highlighted area (copied to clipboard).

`y` and `d` automatically exit to NORMAL mode. Use `p` to paste the contents from clipboard.

`Shift + V` enters VISUAL LINE. It highlights the ENTIRE line as opposed to just where the cursor is. Otherwise, works the same as regular VISUAL.

COMMAND COUNT MOTION

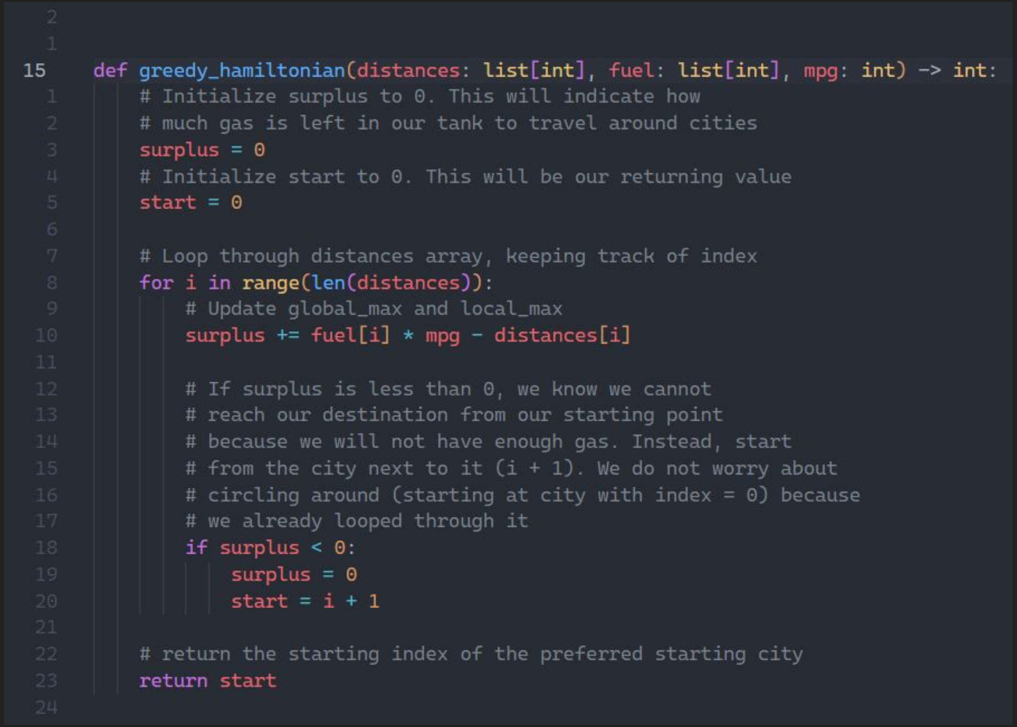
The specific command to be executed (`v, c, y, d`)

The number of times that the motion will execute

How the cursor will move (`h, j, k, l, w`)

Advanced Editing

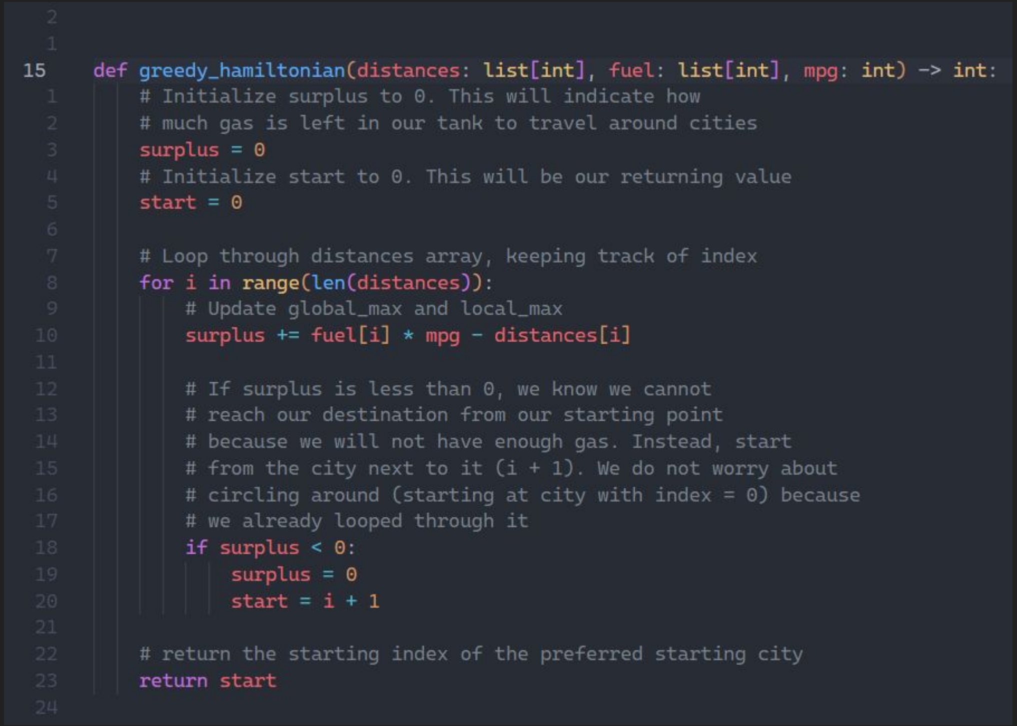
What if we wanted to highlight and delete this entire function?



```python
def greedy_hamiltonian(distances: list[int], fuel: list[int], mpg: int) -> int:
    # Initialize surplus to 0. This will indicate how
    # much gas is left in our tank to travel around cities
    surplus = 0
    # Initialize start to 0. This will be our returning value
    start = 0

    # Loop through distances array, keeping track of index
    for i in range(len(distances)):
        # Update global_max and local_max
        surplus += fuel[i] * mpg - distances[i]

        # If surplus is less than 0, we know we cannot
        # reach our destination from our starting point
        # because we will not have enough gas. Instead, start
        # from the city next to it (i + 1). We do not worry about
        # circling around (starting at city with index = 0) because
        # we already looped through it
        if surplus < 0:
            surplus = 0
            start = i + 1

    # return the starting index of the preferred starting city
    return start
```

Advanced Editing

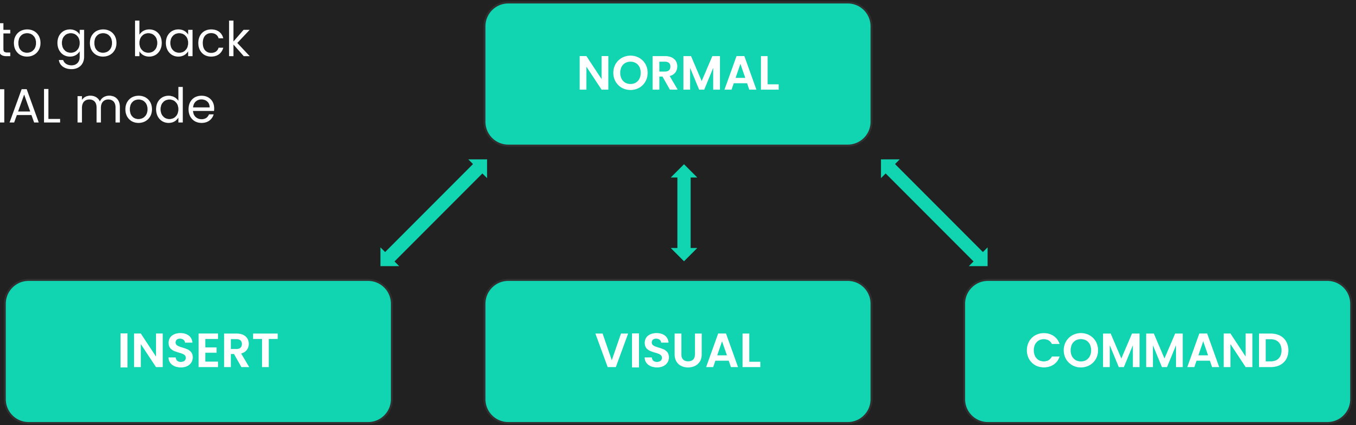
`Shift + V` + `23` + `j` to highlight the function, then `d` to delete it.



```
 2
 1
15  def greedy_hamiltonian(distances: list[int], fuel: list[int], mpg: int) -> int:
 1      # Initialize surplus to 0. This will indicate how
 2      # much gas is left in our tank to travel around cities
 3      surplus = 0
 4      # Initialize start to 0. This will be our returning value
 5      start = 0
 6
 7      # Loop through distances array, keeping track of index
 8      for i in range(len(distances)):
 9          # Update global_max and local_max
10          surplus += fuel[i] * mpg - distances[i]
11
12          # If surplus is less than 0, we know we cannot
13          # reach our destination from our starting point
14          # because we will not have enough gas. Instead, start
15          # from the city next to it (i + 1). We do not worry about
16          # circling around (starting at city with index = 0) because
17          # we already looped through it
18          if surplus < 0:
19              surplus = 0
20              start = i + 1
21
22      # return the starting index of the preferred starting city
23      return start
24
```

VIM is a MODAL Editor: COMMAND

Use Esc to go back to NORMAL mode



COMMAND Mode:

- VIM's mode to issue general commands.
- Save, search & replace, and **QUITTING VIM!** (jk, it's impossible)

COMMAND mode

Enter the command via `:` (colon)

Some of the most common commands:

- `:w` - write the file (save - equivalent to Ctrl-S)
- `:12` - jump to line number 12 (if there exists, otherwise go to last line)
- `:%s/old/new` - search every instance of old and replace it with new
- `:q` - **quits VIM!!!**

You can do mix and match of some the commands (`:wq` to save & quit)
Append `!` to force a command (`:q!` to quit WITHOUT saving changes)

Questions?

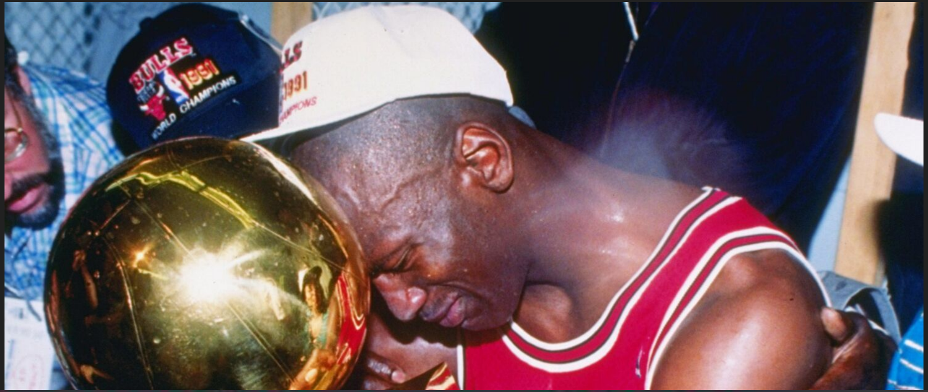
More? … MAURRRR

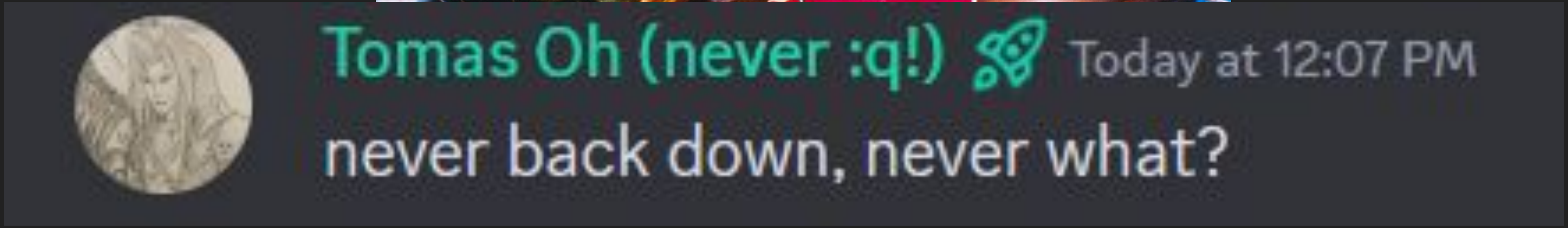
NEOVIM (2014):

- Fork of VIM. Provides all the essential features of VIM with easy and extensible customization with **Lua**. “Modern” VIM.

CONGRATULATIONS!!

You now know how to use **VIM**! Don't forget that everything you want to learn and do in life is up to you. The decision is yours!





Tomas Oh (never :q!) Today at 12:07 PM

never back down, never what?

Resources

ThePrimeagen Tutorials:

- www.youtube.com/watch?v=X6AR2RMB5tE (Introduction)
- www.youtube.com/watch?v=5JGVtttuDQA (Horizontal)
- www.youtube.com/watch?v=KfENDDEpCsI (Vertical)

- www.youtube.com/watch?v=-txKSRn0qeA&pp=ygUDdmlt (Vim in 100 Seconds - Fireship)

- www.youtube.com/watch?v=IiwGbcd8S7I&pp=ygUMdmltIGJlbiBhd2Fk (VIM Tutorial - Ben Awad)

Next week's workshop

Applied Algorithms
Pokemon Mystery Dungeon Generator

Hosted with 💖 by Ethan and Karni!

Attendance form + Questionnaire

forms.gle/Xw8P3rUqNhfbkBpAA

NEXT UP:

Fullyhacks

Demos

NEXT UP:

Project worksesh

acmcsuf.com/oss-sync