

Vim 2

GPI '21 - Lecture 4
Yosef + Deepti



welcome to...

hackCMU

2021 edition, brought to you by ACM@CMU

location: on-campus TBD

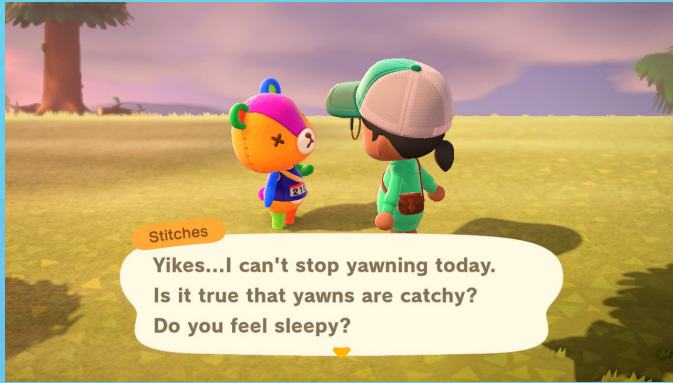
start: friday, october 1, 5:00 pm

end: saturday, october 2, 8:00 pm

Student sign up link: <http://tinyurl.com/hackcmu2021>

Mentor sign up: <https://tinyurl.com/hackcmu21mentor>

(Virtual) Pet Tax





Vim

Part 2

Vim Recap

Normal mode for commands

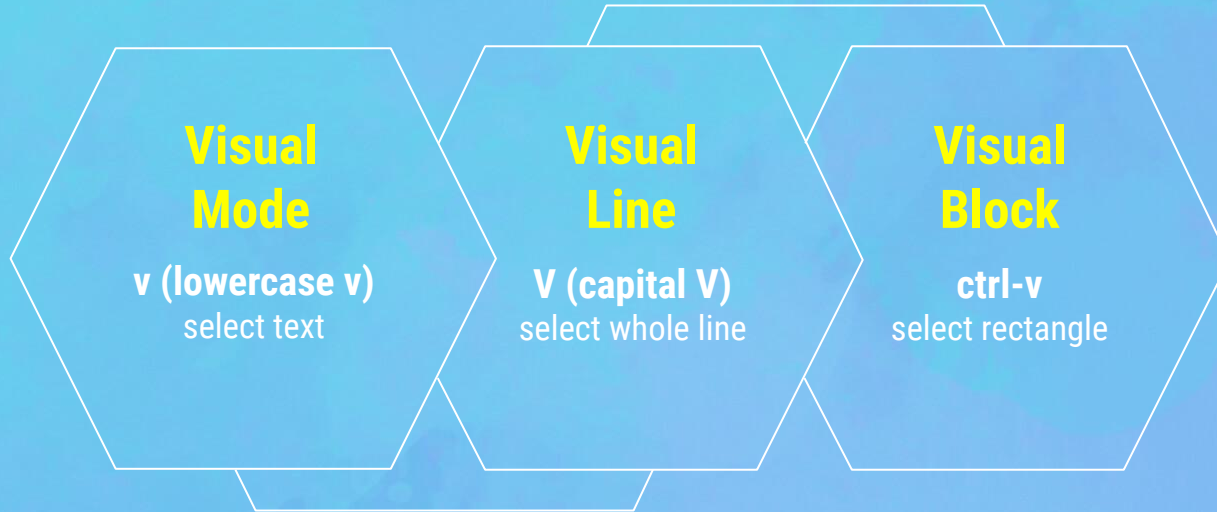
Insert mode for editing text

Undo	u
Redo	:redo OR ctrl-R
Save + Exit	:wq OR :x



⬡ ctrl not command on mac!

More Modes



⬡ ctrl not command on mac!

Text Objects

<number><command><text object or motion>

2dele**w**ord

- ⬡ **<number>**: how many times to perform <command>
- ⬡ **<command>**: change, delete (cut), yank (copy), etc.
- ⬡ **<text object or motion>**: word, sentence, paragraph, etc.

More Text Objects

i

"ci}"

changes **i**nside {curly
braces} and goes into
insert mode

"da("

deletes **a**long with
(parens)

a

- works with most delimiters
 - ◀ " { [(' ` <
- **t** and **p**
 - ◀ change/delete html
tag/paragraph contents

Substitutions

Find and Replace, but with Vim and regex!

:s/<find>/<replace>/<flags> OR **:s:<find>:<replace>:<flags>**

- In **normal mode**
- Specify **ranges** before the s
 - **:10,30s/foo/bar**
 - **:%s/foo/bar/g**
- Example **flags**
 - **g** = every occurrence
 - **c** = confirm before applying

current line: **s**
whole file: **%s**
lines a-b: **a,bs**
line a to end: **a,\$s**
selected
region: **'<,'>s**

- regex = regular expression / pattern that matches against certain strings
- **:noh** = "no highlight" / for erasing highlights



Vim

Demo!

Tabs & splits!

- In the **vim command line**
 - **:tabe file_name.txt** to create a tab
 - **:tabprevious/ tabnext** to navigate
- Splitting a tab
 - **:vsplit** for vertical screen splits
 - **:split** for horizontal screen splits
 - **<Ctrl-w>{h,j,k,l}** to switch between split screens

Macros

- Used for recording a sequence of commands and executing it one or more times
- Create a macro:

q<letter><commands>**q**

- Execute the macro <number> times (once by default):

<number>**@**<letter>

- Must run the macro at the same places -
Use **j0** to go to the start of the next line at the end of the macro

start recording:
q<letter>
stop recording: **q**
execute once: **@**<letter>
execute again: **@@**
view contents of a register:
:reg <letter>
more info: **:h recording**

Where do the sequences of commands
given to **macros** go?

Registers!

<commands>

<commands>

...

Registers

- Registers are spaces in memory that vim uses to store some text. Each has an identifier for later access
- Use the ones denoted with letters (a-z) for regular use
- Numbered registers, the default register, the blackhole register and read-only registers all have special purposes
- Accessed using a double quote before its name. Ex: **"r**
- Copy (yank) the selected text to the register r with **"ry**
- Paste the content of register r with **"rp**
- See all registers with **:reg**
- See contents of specific registers with **:reg <space separated register names>**
Ex: **:reg a b**

Black hole Register

- Is a write-only register that can be accessed with `"_`
- Where vim writes to if it doesn't want to keep track of the text
- Nothing returned if read from

Default Register

- An unnamed (or default) register that can be accessed with `""`
- Any text that you **delete** (with **d**, **c**, **s** or **x**) or **yank** (with **y**) will be placed there
- Vim uses this to paste, when no explicit register is given i.e. the command **p** is the same as doing `""p`

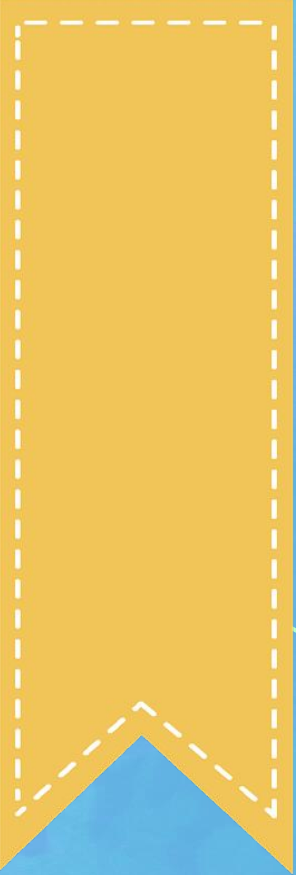
Numbered Registers

- Registers from "0 to "9
- Automatically populated by vim
- "0 has the content of the latest yank
- The others will have last 9 deleted texts - "1 being the newest and "9 the oldest
- Can paste yanked text it using "0p

Read-only Registers

4 read-only registers: ".", "%", ":" and "#"

- "." has the last inserted text
- "%" has the current file path, starting from the directory where vim was first opened
- ":" is the most recently executed command
- "#" is the name of the alternate file a.k.a the last edited file

A yellow bookmark icon with a dashed white outline, positioned vertically on the left side of the slide. It has a rectangular body and a pointed bottom.

Tired of remembering line
numbers, column numbers,
and/or file names?

Book**mark** positions
with **Marks!**

Marks

- Let you bookmark your current position so you can jump to it later
- They are invisible
- Each file has a set of marks identified by lowercase letters (a-z)
- Uppercase letters (A - Z) denote a set of marks used to globally identify a position within a particular file
- Setting a mark with lowercase letters removes the existing one in **current** file
- Setting with uppercase letters removes the existing one in **any** file

Set Mark	m <letter>
Jump to Line of Mark	'<letter>
Jump to Position of Mark	`<letter>
Delete till Line of Mark	d' <letter>
Delete till Position of Mark	d` <letter>
Change Text of Lines Till Mark	c' <letter>
List All Marks	:marks
List Certain Marks	:marks <comma-separated letters>

More Commands with Marks

Jump to Next
Line of Mark

`]'`

Jump to Prev
Line of Mark

`['`

Jump to Next
Mark

`]'`

Jump to
Previous Mark

`['`

Delete Mark

`:delmarks a`

Delete Marks
in Range

`:delmarks a-c`

Delete Specified
Marks

`:delmarks abc`

Delete All
Lowercase Marks

`:delmarks!`



- **SportsLab** due 11:59 pm ET tonight
- Extratation this Saturday 1-2 pm at GHC 4211: **Interview questions**
- **Course Feedback** on tinyurl.com/f21-gpi-feedback



○ **fix-typos**

- ◀ don't swap the names in the header

○ **hogwarts**

- ◀ leave TWO spaces between books and MORE
- ◀ deleting inside the <container> tag leaves the closing tag on a new line, which is not considered correct. the tags should be next to each other (vim version dependent)
- ◀ after deleting paragraphs there should be ONE line in between them