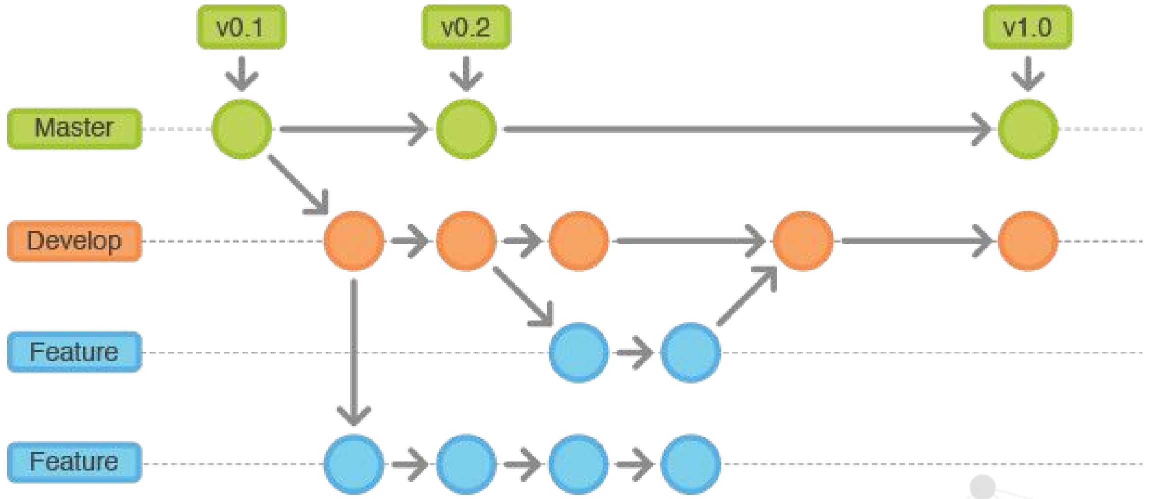
Learn Version Control the Hard Way

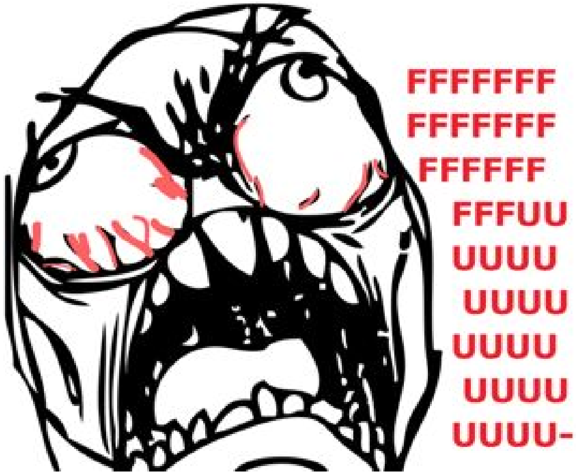
Alireza Samar

UNIVERSITI TEKNOLOGI MALAYSIA

MLDS MACHINE LEARNING FOR DATA SCIENCE

We're going to learn ..





MLDS

What is version control?

- Keeps track of your creative output.
- It tracks what is changed.
- It tracks who makes the changes.
- It tracks why changes were made.
- Keeps track that evolution is vital.
- Everyone needs it
    - developers / designers
    - writers / producers
    - artists / composers
- You can use it as part of team or by yourself.
- Keeping track of changes and history is the key to creative success.

MLDS

Every good journey begins with a story …

MLDS

Why version control?

- **Peace of mind**: Automatic backups
- **History**: Change-by-change log of your work
- **Friction-free undo**: For both short-term and long-term

MLDS

How do teams benefits from control?

- **Synchronisation**: Easy to keep team members always up-to-date
- **Accountability**: Know who made each change and why
- **Conflict detection**: Keep the build clean every time

MLDS

In fact …
you already do version control even if you don’t use any software!

MLDS

- *Thesis-Working ~~Thesis-Working~~*
- *Thesis-ShipIt ~~Thesis-ShipIt~~*
- *Thesis-Final ~~Thesis-Final~~*
- *Thesis-FinalFinalWithBugFixes*

Everything is now automated

- **Backups**: Every version is kept around
- **Change tracking**: Commit messages let you know why things changed
- **Rollback to previous versions**: It's like undo for coding
- **Labelling significant changes**: Tags/labels identify the state of the source that matches each release

MLDS

A quick history

- Stand-alone and file-focused
    - **SCCS**
      1972, Unix only
    - **RCS**
      1982, cross-platform, text only

A quick history

- Centralized
    - **CVS**
      1986, first central repository, file-focused
    - **Perforce**
      1995, still the biggest repository inside Google
    - **Subversion**
      2000, non-text files, track directory structure, transaction unit
    - **Microsoft Team Foundation Server**
      2010, comes with MSDN subscription, tight Visual Studio integration

MLDS

A quick history

- Distributed
    - **Git**
      2005, created by Linus Torvalds after BitKeeper went commercial only. Broadly used in conjunction with GitHub, which offers free hosting for open-source projects.
    - **Mercurial**
      2005, also created in response to BitKeeper change

MLDS

Essential version control concepts

- **Repository** aka database
  - where your files and their history is stored

- **Working set**
  - the current state of the files as stored on your local machine

- **Add**
  - insert new files from working set into the repository

MLDS

Essential version control concepts

- **Check-in / Commit**
  - copy changes from working set to repository
- **Check-out / Update**
  - copy changes from repository to working set
- **Tag / Label**
  - mark the current state of the repository for future checkout

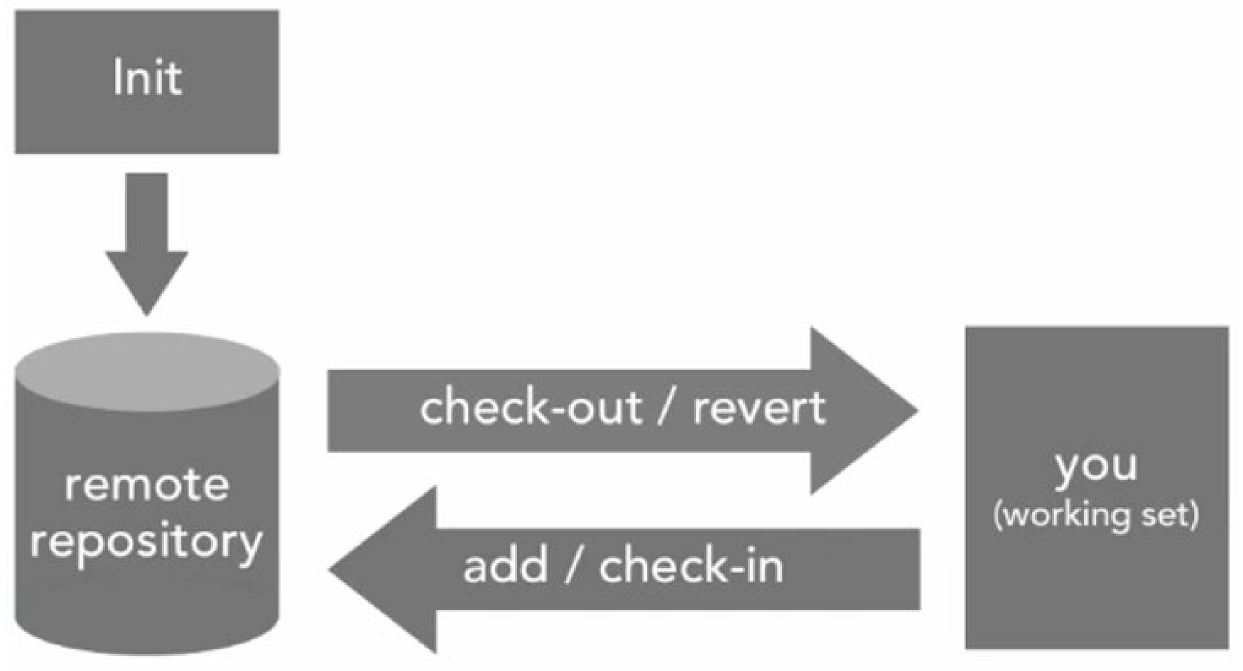
MLDS

Distributed repository systems only

- **push / export**
  - send changes from one repo to another
- **pull / import**
  - update your working set with updates
- **Tag / Label**
  - name a specific state of a repo
- **branch / fork**
  - make a clone of a repo
- **Merge**
  - integrate your branch (clone) back into the original repo

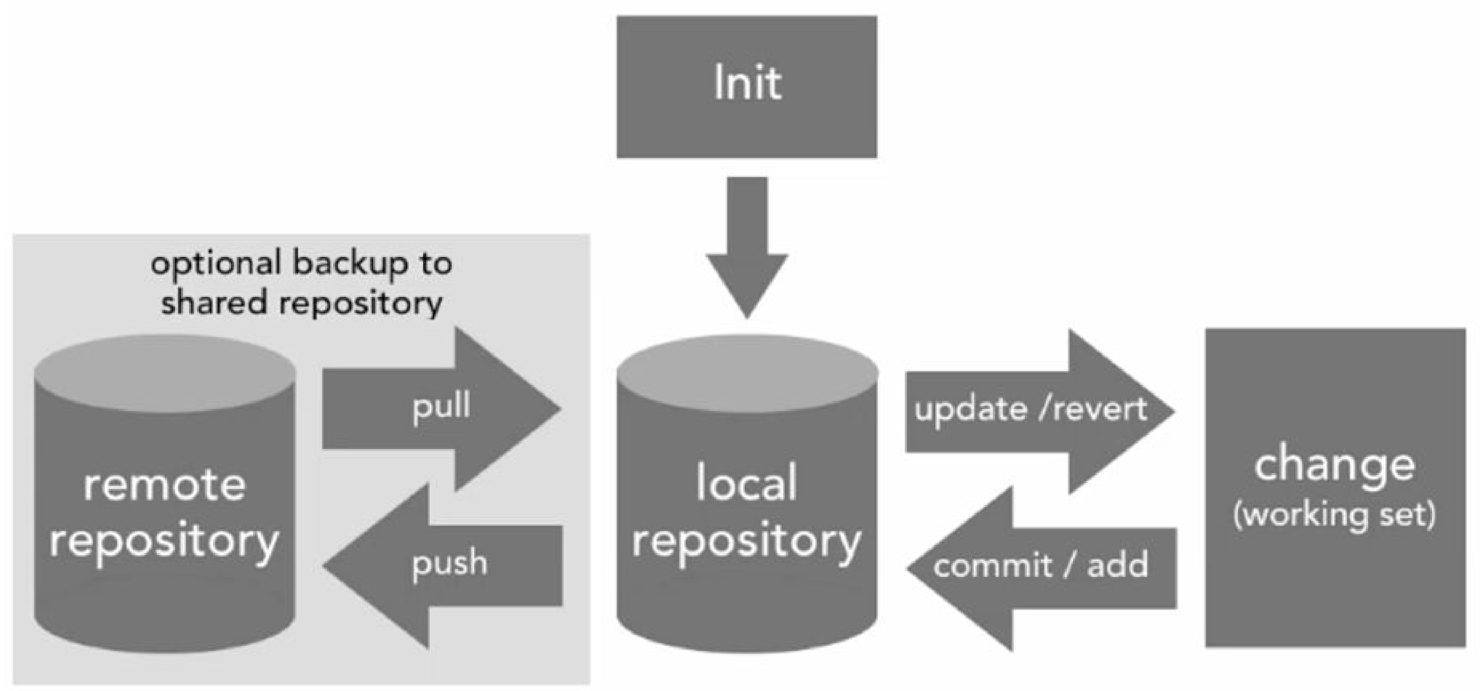
MLDS

Centralized vs. Distributed



MLDS

Centralized vs. Distributed

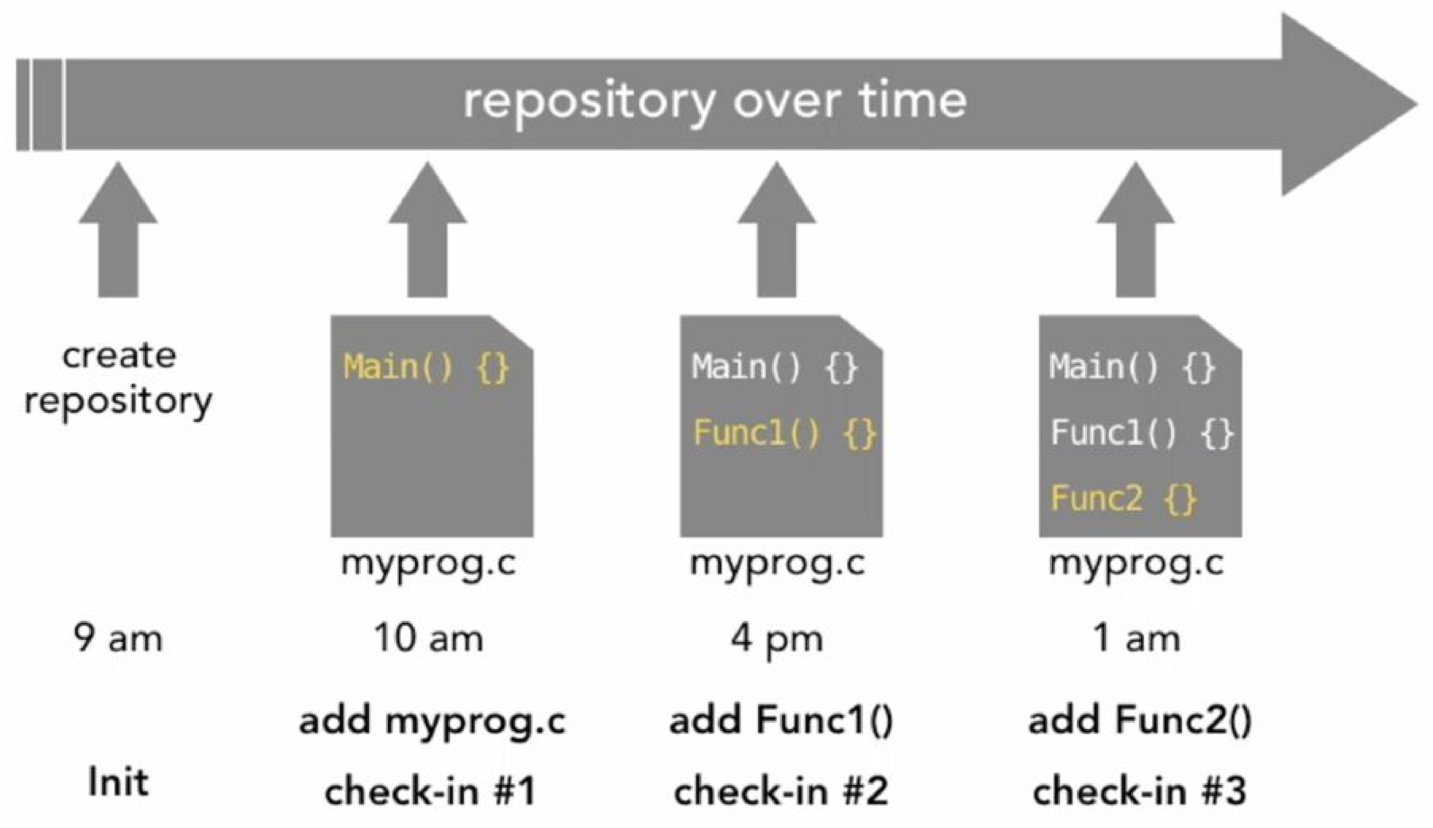


MLDS

Git

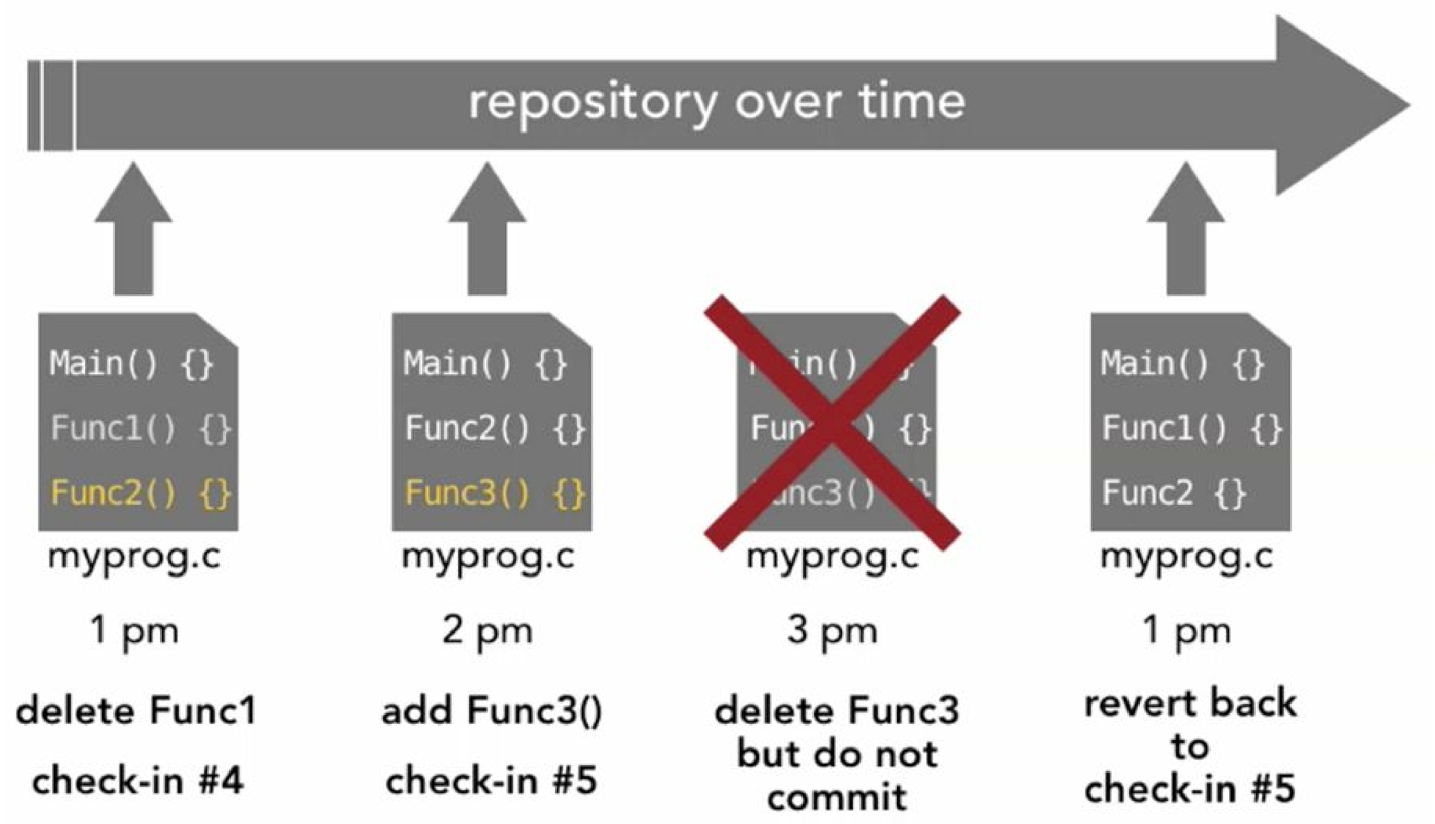
- Type: Distributed
- Free with commercial hosting options
- Open-source free bits: http://git-scm.com
- Free and commercial hosting for open-source and closed source: http://github.com

Git in action



MLDS

Git in action



MLDS

Let’s do it together …

1. `git --version` (make sure Git installed properly)
2. `mkdir fsvc` (create your directory)
3. `cd fsvc` (change the path to your directory)
4. `git init` (setup Git)
5. `dir /a` (check the Git hidden database file)
6. `git status` (make sure it initialised properly)

MLDS

Let’s do it together …

7. `notepad myprog.py` (create a file, add “Main () {}” and save it)
8. `git add *` (to tell Git to track new files)
9. `git commit -a -m` “Add myprog.py”
10. Error?
    a. `git config --global user.email “you@example.com”`
    b. `git config --global user.name “Your Name”`
11. `git commit -a -m “Add myprog.py”`
12. `git status` (to verify your success commit)

Let’s do it together …

13. `git log -p` (ask Git to tell you the history)
14. `notepad myprog.py` (let’s reopen the file, add “Func1 () {}” and save it)
15. `git commit -a -m “Add Func1()”`
16. `git log -p` (check the changes you’ve made)
17. `notepad myprog.py` (one more time, let’s reopen the file, add “Func2 () {}” and save it)
18. `git commit -a -m “Add Func2()”`
19. `git log -p` (check the changes you’ve made)

MLDS

You’re almost there …

20. `git log --oneline --all` (ask Git for shorter version of your history)
21. `git diff 6a3e577 7555e83` (ask Git to show you the differences, take note that identifiers are different for you Git!)

MLDS

Let’s make some Oops!

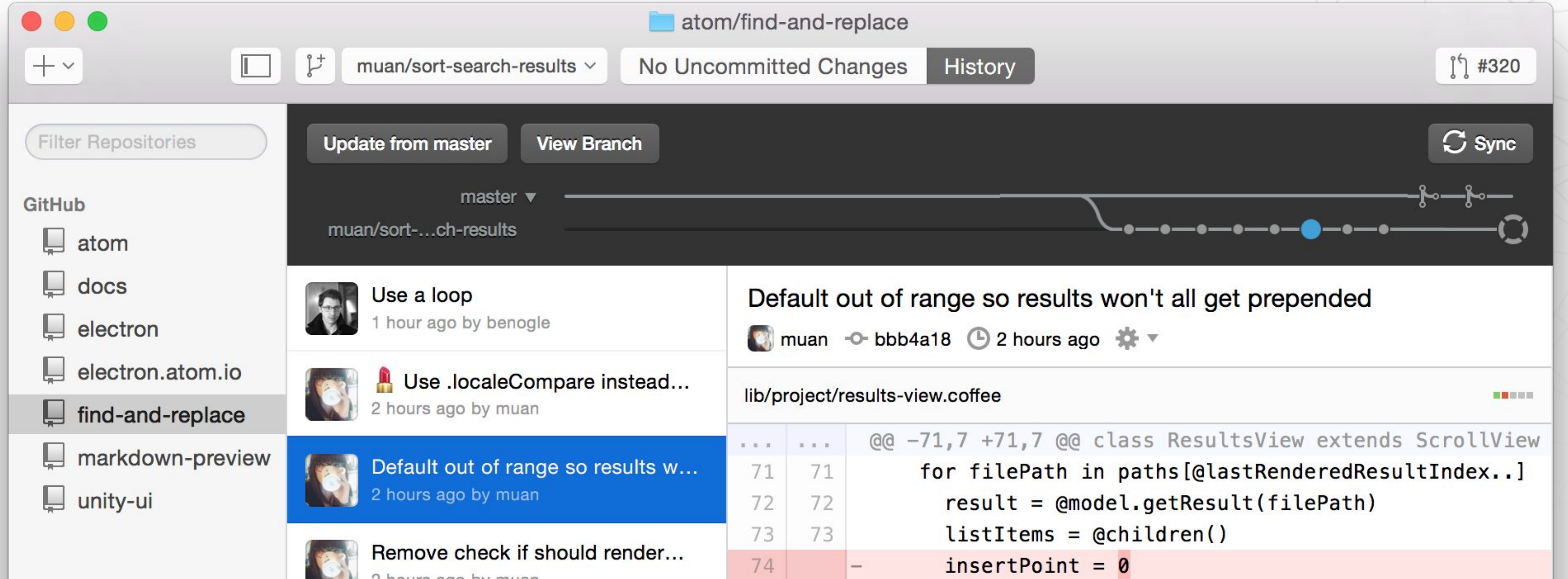
22. `notepad myprog.py` (open the file, delete “Func1 () {}” and save it)
23. `git diff` (check what’s in repo and what’s in working set)
24. `git commit -a -m “Delete Func1()”`
25. `notepad myprog.py` (let’s reopen the file, add “Func3 () {}” and save it)
26. `git commit -a -m “Add Func3()”`
27. `git log --oneline --all`

Let’s make some Oops!

28. `notepad myprog.py` (open the file, delete “Func3 () {}” and save it) (we did it in our working set, didn’t commit yet)
29. `type myprog.py` (but we didn’t meant to do that!)
30. `git diff HEAD` (check the difference between repo and working set)
31. `git checkout myprog.py` (revert back the mistake we’ve made)
32. `type myprog.py` (and it’s back again!)
33. `git diff HEAD / git diff / git status` (you can see there is no difference)

You got it all!

https://desktop.github.com



Use a loop
1 hour ago by benogle

Use .localeCompare instead...
2 hours ago by muan

Default out of range so results w...
2 hours ago by muan

Remove check if should render...

Default out of range so results won't all get prepended
muan bbb4a18 2 hours ago

lib/project/results-view.coffee

```
@@ -71,7 +71,7 @@ class ResultsView extends ScrollView
      for filePath in paths[@lastRenderedResultIndex..]
        result = @model.getResult(filePath)
        listItems = @children()
-       insertPoint = 0
```

Thanks!

Machine Learning for Data Science Interest Group
Advanced Informatics School
Universiti Teknologi Malaysia

@utmmlds
ais.utm.my/mlds

February 2017

MLDS