Git Workshop Common Scratch Pad

**http://go.unimelb.edu.au/665a**

# Greetings …

Hi, I’m Errol

twitter: [@maegul](https://twitter.com/maegul)

Email: errolkjlloyd@gmail.com

GitHub maegul

I’ll be leading us through our introduction to git and GitHub.

Helping us will be …

The ***unix arch-guru*** Tim

(twitter: [@resnomicon)](https://twitter.com/resnomicon).

Meirian

(Email: [meirian.lovelacetozer@gmail.com](mailto:meirian.lovelacetozer@gmail.com), GitHub: [Meirian](https://github.com/Meirian), Twitter: [@MeirianLT](https://twitter.com/MeirianLT))

Git for Windows

<https://gitforwindows.org>

Join the facebook group: <https://www.facebook.com/groups/522053638172827/>

Feedback survey: [go.unimelb.edu.au/bym6](http://go.unimelb.edu.au/bym6)  
For “What training have you just attended?”, select “Introduction to Git and GitHub” from the dropdown menu (3rd from bottom)

[Greetings …](#_rdvzazajes7t)

[WiFi Access](#_f52ypvj8u1zr)

[Download Links](#_7ben0g211d4w)

[Graphical App for Git and GitHub](#_jgv6b0cp41uw)

[Text Editors](#_twy1rt82yt02)

[Command Line Backup](#_du13pl1nky61)

[Converting your account](#_yin7tloda1ep)

[Challenges](#_h03x7bo3i0d9)

[Desktop App](#_qqffor7yugzt)

[No 1](#_uxqs6ni87vjf)

[No 2](#_lvo6dy9fnxp8)

[No 3](#_xkaq253c7oz9)

[No 4](#_o7zsyun1xdda)

[Command Line](#_is2rom7cn01g)

[No 1](#_qtim1lpr6zvn)

[No 2](#_20b8u0mdr2uv)

[No 3](#_kjtp84u8sjr4)

[No 4](#_4gkoncrypnll)

[Special (Logging)](#_e4ig9x514knt)

[No 5](#_m6951aj3s6qf)

[No 6](#_h5wo0m9tauhs)

[Sample File](#_xib6g1fkiydo)

[A Second Sample File](#_u17yigdunnq5)

[Vocabulary](#_rdemxw52ajdd)

[Cheat Sheet](#_eh79r7mmzugp)

[Resources](#_p7xwbwlx4ah1)

# 

# 

# WiFi Access

SoftwareCarpentry

hackerwithin

# 

# Download Links

[Installation page](https://resbaz.github.io/intro-git-workshop/)

[Free student account upgrade](https://education.github.com/) Click ‘Request a Discount’

## Graphical App for Git and GitHub

[GitHub Desktop](https://desktop.github.com)

## Text Editors

[Atom](https://atom.io)

[Sublime Text](https://www.sublimetext.com)

[Visual Studio Code](https://code.visualstudio.com/?wt.mc_id=DX_841432)

Notepad (often already on Windows) or [Notepad++](https://notepad-plus-plus.org/download/v7.5.5.html)

## Command Line Backup

<https://notebook.cloud.unimelb.edu.au/hub/login>

## Converting your account

To be able to create a private repo on github:

<https://education.github.com/>

Then follow the instructions (linking + verifying your unimelb email address)

# 

# Challenges

# Desktop App

## No 1

\*\*CHALLENGE\*\*

* Make a New Repository
* Open your text editor
* \* Get sample file copy and paste from google doc
* Put your file in your repository
* \* Add each function separately and commit separately
  + \* Add appropriate commit messages

\*Extension\*

1. \* Using icons around the line numbers of the code, can you work out how to commit specific lines only?
2. \* If you have multiple files, each with their own changes, can you work out how to commit changes specific to a single file.
   1. \* Add another file, perhaps by copying the main file, giving it a different file name to test this out.

## No 2

\*\*CHALLENGE\*\*

* \* Revert two of the functions
  + \_Check your file to see that the change has taken place\_
* \* Delete a file and then bring it back

\*Extension\*

* \* Can you Revert the reversions?
* \* Can you undo all of your commits, and then return your repo back to the original state?

## No 3

* Branch
* Add a function
* Merge back into Master Branch

*Extension*

* Make two branches.
  + In one branch, add another function
  + In the other branch, change a function that already exists
  + Merge both of these new branches into your master branch
  + *Summarises the idea of the branch as a new feature*
* Conflicts
  + Make a branch from master
  + In the new branch, change a line (any line) to something different.
  + In the master branch, change the same line to something different again (and different from you did in the new branch)
  + Now try to merge
  + *Introduces idea of the conflict*

## No 4

* Get GitHub account
* Add it to your desktop app
* Push your repository to GitHub
* Let me clone it (paste to google doc below)

Olga: <https://github.com/odrath/resbaz_training>

Suwash: <https://github.com/suwash/SampleFunctions>

Johanna <https://github.com/likeajumprope/Test2.git>

## No 5

* Clone a repository of mine to have it for yourself:
* <https://github.com/maegul/git-Commands-Cheat-Sheet>

# Command Line

## No 1

* Open your command line
* Find out which folder it starts in by default
  + Can you use either *pwd* or *ls* to do so?
* Navigate to the folder you’ve set up for command line git.

*Extension*

* Can you figure out what “.” and “..” represent in the command line?
  + Using ls or cd will help you.
* after entering the command ‘vim’, can you get back to the command line?

## No 2

* Initialise your git repository, add your files and commit it with a commit message.
* Make a change, and in your commit, omit the -m and message (*git commit*).
  + How does git respond? Can you make it happy?
  + This is intended to create a potentially unfriendly problem for you. It’s possible to make it go away, but not intuitive.

*Extension*

* Get familiar with the git help command

git help

* Figure out how to list all the git subcommands and concepts.

## No 3

* Add the functions of your code file. Commit each one separately, providing a decent commit message.
* Use the staging area for some of the functions.
* Try changing both of your files, but then only commit changes for one at a time (clue: use the staging area/index)

*Extension*

* review your history with *git log*. Use git help to see what modifications you can make to the way the log is displayed.

## No 4

* Look at your log or history
* abbreviate your history
* look at the part of the history relevant to only one file
  + can you abbreviate that history too?
* Use –before to see only the last two commits (may need to pay attention to the seconds)
* Use git show to at look at what you did in your second commit
* In which commit did you affect the least amount of lines and files?

## Special (Logging)

* Given commands below, and what you know about your history
  + Use –grep to get only your first commit
  + Use -S to get only the commits in which you changed the function “division”
  + Use -L to specify the lines for a specific function in the code

## No 5

* Revert your last commit
* Revert that reversion
* Reset your repo to before you bothered with the reversions
* Look at what you main code file looked like after your second commit. THen return to the present

**CHallenge**

* Delete a file and bring it back

**Extension**

* There are three modes of resetting: soft, mixed and hard. Try to work out the differences between them by experimentation.
* When you have travelled back in time and checked out another commit, try to make some changes and commit them. What happens? What happens if you go back to the “present” with *git checkout master*?

## No 6

* In a new branch, create a new function (copy and paste!) and merge it back into the master branch.
  + Check your log and your file constantly to see the changes *checkout [branch name]* and *git merge* are making to your files.
* Make a new branch. Change a function in the master branch, and commit that change. Now change the same function in the new branch, but in a different way. Try merging the new branch into the master branch.

# 

# Sample File

# def addition(a, b):

# summation = a + b

# return summation

# def subtraction(a, b):

# difference = a - b

# return difference

# def division(a, b):

# quotient = a / b

# return quotient

# def multiplication(a, b):

# product = a \* b

# return product

# 

# A Second Sample File

# def bad\_addition(a, b):

# summation = a + b + 0.002

# return summation

# def bad\_subtraction(a, b):

# difference = a - b + 7

# return difference

# def bad\_division(a, b):

# quotient = (a+13) / b

# return quotient

# def bad\_multiplication(a, b):

# product = a \* b + 12

# return product

# 

# Vocabulary

# Cheat Sheet

Online:

<https://maegul.github.io/git-Commands-Cheat-Sheet/>

# Resources

Useful instructions for Git configuration, especially changing the default editor so you don’t get stuck in Vim: <https://swcarpentry.github.io/git-novice/02-setup/>

Comprehensive Git textbook (free): <https://git-scm.com/book/en/v2>