

# Research Software Engineering with Python

## Introduction

In this course, you will move beyond programming, to learn how to construct reliable, readable, efficient research software in a collaborative environment. The emphasis is on practical techniques, tips, and technologies to effectively build and maintain complex code. This is a relatively short course (8-10 half-day modules) which is intensive and involves hands-on exercises.

## Pre-requisites

- It would be helpful to have experience in at least one programming language (for example [C++](#), [C](#), [Fortran](#), [Python](#), [Ruby](#), [Matlab](#) or [R](#)) but this is not a requirement.
- Experience with version control and/or the [Unix](#) shell, for instance from Software Carpentry, would also be helpful.
- You should bring your own laptop to the course as there are several hands-on exercise for you to work through.
- We have provided [setup](#) instructions for installing the software needed for the course on your computer.
- **Eligibility:** This course is primarily aimed at Turing-connected PhD students. Other Turing-affiliated people might join too if capacity allows.

## Instructors

- [Turing Research Engineering Group](#)

## Exercises

Examples and exercises for this course will be provided in [Python](#). We will assume you have prior experience with at least one programming language but [Python](#) syntax and usage will be introduced during this course. However, this course is **not** intended to teach [Python](#) and you may find supplementary content useful.

## Evaluation

**None:** you are not graded. There are two exercises that you can use for self-assessment.

- [Packaging Greengraph](#)
- [Refactoring the Bad Boids](#)

## Versions

You can browse through course notes as HTML or download them as a printable PDF via the navigation bar to the left.

If you encounter any problem or bug in these materials, please remember to add an issue to the [course repo](#), explaining the problem and, potentially, its solution. By doing this, you will improve the instructions for future users. 

## Installation Instructions

# Introduction

This document contains instructions for installation of the packages we'll be using during the course. You will be following the training on your own machines, so please complete these instructions.

If you encounter any problem during installation and you manage to solve them (feel free to ask us for help), please remember to add an issue to the [course repo](#), explaining the problem and solution. By doing this you will be helping to improve the instructions for future users! :tada:

## What we're installing

- the [Python](#) programming language (version [3.7](#) or greater)
- some [Python](#) software packages that will be used during the course.
- [git](#) for the version control module
- your favourite text editor

Please ensure that you have a computer (ideally a laptop) with all of these installed.

# Linux

## Package Manager

[Linux](#) users should be able to use their package manager to install all of this software (if you're using [Linux](#), we assume you won't have any trouble with these requirements).

However note that if you are running an older [Linux](#) distribution you may get older versions with different look and features. A recent [Linux](#) distribution is recommended.

## Python via package manager

Recent versions of [Ubuntu](#) come with mostly up to date versions of all needed packages. The version of [IPython](#) might be slightly out of date. Advanced users may wish to upgrade this using [pip](#) or a manual install. On [Ubuntu](#) you should ensure that the following packages are installed using [apt-get](#).

- [python3-numpy](#)
- [python3-scipy](#)
- [python3-pytest](#)
- [python3-matplotlib](#)
- [python3-pip](#)
- [jupyter](#)
- [ipython](#)
- [ipython3-notebook](#)

Older distributions may have outdated versions of specific packages. Other [Linux](#) distributions most likely also contain the needed [Python](#) packages but again they may also be outdated.

## Python via Anaconda

We recommend you use [Anaconda](#), a complete independent scientific python distribution.

Download [Anaconda for Linux](#) with your web browser, choosing the latest version. Open a terminal window, go to the place where the file was downloaded and type:

```
bash Anaconda3-
```

and then press [Tab](#). The name of the file you just downloaded should appear.

Follow the text prompts ensuring that you:

- agree to the licence
- prepend **Anaconda** to your **PATH** (this makes the **Anaconda** distribution the default **Python**)

You can test the installation by opening a new terminal and checking that:

```
which python
```

shows a path where you installed **Anaconda**.

## Python via Enthought Canopy

Alternatively you may install a complete independent scientific python distribution. One of these is **Enthought Canopy**.

The **Enthought Canopy** Python distribution exists in two different versions. A basic free version with a limited number of packages (**Canopy Express**) and a non free full version. The full version can be obtained free of charge for academic use.

You may then use your Enthought user account to sign into the installed **Canopy** application and activate the full academic version. **Canopy** comes with a package manager from where it is possible to install and update a large number of python packages. The packages installed by default should cover our needs.

## Git

If **git** is not already available on your machine you can try to install it via your distribution package manager (e.g. **apt-get** or **yum**), for example:

```
sudo apt-get install git
```

## Editor

Many different text editors suitable for programming are available. If you don't already have a favourite, you could look at one of these:

- [Visual Studio Code](#)
- [Atom](#)
- [Sublime Text](#)
- [Vim](#)
- [Emacs](#)

Regardless of which editor you have chosen you should configure **git** to use it. Executing something like this in a terminal should work:

```
git config --global core.editor NameofYourEditorHere
```

The default shell is usually **bash** but if not you can get to **bash** by opening a terminal and typing **bash**.

## MacOS

### Upgrade MacOS

We do not recommend following this training on older versions of **macOS** without an app store: upgrade to at least **macOS 10.9 (Mavericks)**.

## XCode and Command line tools

Install the **XCode** command-line-tools by opening a terminal and run the following.

```
xcode-select --install
```

And follow the on screen instructions.

You may also install **Xcode** from the app store if you wish, but it is not needed.

## Git

The **XCode** command line tools come with **Git** so no need to do anything more.

## Homebrew

**Homebrew** is a package manager for **macOS** which enables the installation of a lot of software useful for scientific computing. It is required for some of the installations below. **Homebrew** requires the **Xcode** tools above.

Install **homebrew** via typing this at a terminal:

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

and then type.

```
brew doctor
```

And read the output to verify that everything is working as expected. If you are already running **MacPorts** or another package manager for **macOS** we don't recommend installing **homebrew** as well.

## Python

We recommend installing a complete scientific python distribution. One of these is [Anaconda](#). Please download and install [Anaconda](#) (latest version).

## Python from Homebrew

Alternatively if you wish to install **Python** manually you can use Homebrew. **macOS** ships with **Python** and some packages. However this has known limitations and we do not recommend it. You can install a new version of **Python** from **homebrew** with the following. Please follow the instructions above to install the **Xcode** command line tools and **homebrew** before attempting this.

```
brew install python3
```

In order to ensure that this version of **Python** is selected over the **macOS** default version you should execute the following command:

```
echo export PATH='/usr/local/bin:$PATH' >> ~/.bash_profile
```

and reopen the terminal. Verify that this is correctly installed by executing

```
python --version
```

Which should print:

```
Python 3.8.x
```

(where x will be replaced by a version number)

This will result in an installation of `python3` and `pip3` which you can use to have access to the latest `Python` features which will be taught in this course.

Then install additional `Python` packages by executing the following.

```
brew install [package-name]
```

- `pkg-config`
- `freetype`
- `gcc`

```
pip3 install [package-name]
```

- `numpy`
- `scipy`
- `matplotlib`
- `jupyter`
- `ipython[all]`

The following packages should be installed automatically as dependencies, but we recommend installing them manually just in case.

- `tornado`
- `jinja2`
- `pyzmq`
- `pytest`

## Editor and shell

The default text editor on `macOS` `TextEdit` should be sufficient for our use. Alternatively consider one of these editors:

- [Visual Studio Code](#)
- [Atom](#)
- [Sublime Text](#)
- [Vim](#)
- [Emacs](#)

To setup `git` to use `TextEdit` executing the following in a terminal should do.

```
git config --global core.editor /Applications/TextEdit.app/Contents/MacOS/TextEdit
```

For VS Code:

```
git config --global core.editor "code --wait"
```

The default terminal on `macOS` should also be sufficient. If you want a more advanced terminal [iTerm2](#) is an alternative.

## Windows

### Python

We recommend installing a complete scientific python distribution. One of these is [Anaconda](#).

Please download and install [Anaconda](#) (Python 3.8 version).

## Sophos

To use the [IPython](#) notebook on a [Windows](#) computer with Sophos anti-virus installed it may be necessary to open additional ports allowing communication between the notebook and its server. The [solution](#) is:

- open your [Sophos Endpoint Security and Control Panel](#) from your tray or start menu
- Select [Configure > Anti-virus > Authorization](#) from the menu at the top
- Select the websites tab
- click the [Add](#) button and add [127.0.0.1](#) and [localhost](#) to the [Authorized websites](#) list
- restart computer (or just restart the [IPython](#) notebook)

## Git

Install the [GitHub for Windows client](#). This comes with both a GUI client as well as the [Git Bash](#) terminal client which we will use during the course. You should register with [Github](#) for an account and sign into the GUI client with this account. This will automatically set-up [SSH based authentication](#) for the terminal client. The terminal client comes in 3 different flavours based on [Windows CMD](#) (DOS like), [Windows Powershell](#), and [BASH](#). We will use the [BASH](#) client as this most closely resembles the [Linux](#) and [OS X](#) terminal used by other students. In order to configure this open the Github client. Sign in with your credentials and:

- Select tools
- Options
- Default Shell
- Git Bash
- And Press Update to save.

Verify that this is working by opening Git Bash. The Shell window should have a title that starts with MINGW32.

## Editor

Unless you already use a specific editor which you are comfortable with we recommend using one of the following:

- [Visual Studio Code](#)
- [Atom](#)
- [Notepad++](#)
- [Sublime Text](#)
- [Vim](#)
- [Emacs](#)

Using any of these to edit text files including code should be straight forward. [Visual Studio Code](#) has integrations with [Git Bash](#) and the [Python prompt](#) that you may want to configure.

## Testing your install

Check this works by opening the Github shell. Once you have a terminal open, type

```
which code
```

which should produce readout similar to [/c/Program Files \(x86\)/Code/Code.exe](#)

Also verify that typing:

```
code
```

opens the editor and then close it again.

Also test that

```
which git
```

produces some output like `/bin/git`. The `which` command is used to figure out where a given program is located on disk.

## Telling Git about your editor

Now we need to update the default editor used by `Git`.

```
git config --global core.editor "code --wait"
```

Note that it is not obvious how to copy and paste text in a `Windows` terminal including `Git Bash`. Copy and paste can be found by right clicking on the top bar of the window and selecting the commands from the drop down menu (in a sub menu).

## Testing Python

Confirm that the `Python` installation has worked by typing:

```
python -V
```

Which should result in details of your installed `Python` version. This should print the installed version of the `Python` and `Git` confirming that both are installed at working correctly. You should now have a working version of `Git`, `Python`, and your chosen editor, all accessible from your shell.

# Version Control

- Why use version control
- Solo use of version control
- Publishing your code to GitHub
- Collaborating with others through Git
- Branching
- Rebasing and Merging
- Debugging with GitBisect
- Forks, Pull Requests and the GitHub Flow

## Introduction

### What's version control?

Version control is a tool for **managing changes** to a set of files.

There are many different **version control systems**:

- Git
- Mercurial ([hg](#))
- CVS
- Subversion ([svn](#))
- ...

### Why use version control?

- Better kind of **backup**.
- Review **history** ("When did I introduce this bug?").
- Restore older **code versions**.

- Ability to **undo mistakes**.
- Maintain **several versions** of the code at a time.

Git is also a **collaborative** tool:

- “How can I share my code?”
- “How can I submit a change to someone else’s code?”
- “How can I merge my work with Sue’s?”

## Git != GitHub

- **Git**: version control system tool to manage source code history.
- **GitHub**: hosting service for Git repositories.

## How do we use version control?

Do some programming, then commit our work:

```
my_vcs commit
```

Program some more.

Spot a mistake:

```
my_vcs rollback
```

Mistake is undone.

## What is version control? (Team version)

**Sue**                    **James**

```
my_vcs commit        ...
```

...                      Join the team

```
...                      my_vcs checkout
```

...                      Do some programming

```
...                      my_vcs commit
```

```
my_vcs update        ...
```

Do some programming   Do some programming

```
my_vcs commit        ...
```

```
my_vcs update        ...
```

```
my_vcs merge         ...
```

```
my_vcs commit        ...
```

## Scope

This course will use the **git** version control system, but much of what you learn will be valid with other version control tools you may encounter, including subversion (**svn**) and mercurial (**hg**).

# Practising with Git

## Example Exercise

In this course, we will use, as an example, the development of a few text files containing a description of a topic of your choice.

This could be your research, a hobby, or something else. In the end, we will show you how to display the content of these files as a very simple website.

## Programming and documents

The purpose of this exercise is to learn how to use Git to manage program code you write, not simple text website content, but we'll just use these text files instead of code for now, so as not to confuse matters with trying to learn version control while thinking about programming too.

In later parts of the course, you will use the version control tools you learn today with actual Python code.

## Markdown

The text files we create will use a simple “wiki” markup style called [markdown](#) to show formatting. This is the convention used in this file, too.

You can view the content of this file in the way Markdown renders it by looking on the [web](#), and compare the [raw text](#).

## Displaying Text in this Tutorial

This tutorial is based on use of the Git command line. So you'll be typing commands in the shell.

To make it easy for me to edit, I've built it using Jupyter notebook.

Commands you can type will look like this, using the %%bash “magic” for the notebook.

If you are running the notebook on windows you'll have to use %%cmd.

```
%%bash  
echo some output
```

```
some output
```

with the results you should see below.

In this document, we will show the new content of an edited document like this:

```
%%writefile somefile.md  
Some content here
```

```
Writing somefile.md
```

But if you are following along, you should edit the file using a text editor. On windows, we recommend [Notepad++](#). On mac, we recommend [Atom](#)

## Setting up somewhere to work

```
%%bash
rm -rf learning_git/git_example # Just in case it's left over from a previous class; you
won't need this
mkdir -p learning_git/git_example
cd learning_git/git_example
```

I just need to move this Jupyter notebook's current directory as well:

```
import os

top_dir = os.getcwd()
top_dir

git_dir = os.path.join(top_dir, "learning_git")
git_dir

'/home/runner/work/rsd-engineeringcourse/rsd-
engineeringcourse/module04_version_control_with_git'

working_dir = os.path.join(git_dir, "git_example")

os.chdir(working_dir)
```

## Solo work

### Configuring Git with your name and email

First, we should configure Git to know our name and email address:

```
git config --global user.name "YOUR NAME HERE"
git config --global user.email "yourname@example.com"
```

Note that by using the `--global` flag, we are setting these options for all projects. To set them just for this project, use `--local` instead.

Now check that this worked

```
%%bash
git config --get user.name
```

```
Turing Developer
```

```
%%bash
git config --get user.email
```

```
developer@example.com
```

There is one additional setting we apply that will be useful later on in the course

```
git config --global pull.rebase false # apply the default strategy when merging - DON'T
WORRY ABOUT WHAT THIS MEANS YET!
```

```
File "/tmp/ipykernel_3285/2032664494.py", line 1
    git config --global pull.rebase false # apply the default strategy when merging -
DON'T WORRY ABOUT WHAT THIS MEANS YET!
^
SyntaxError: invalid syntax
```

## Initialising the repository

Now, we will tell Git to track the content of this folder as a git "repository".

```
%%bash
pwd # Note where we are standing-- MAKE SURE YOU INITIALISE THE RIGHT FOLDER
git init --initial-branch=main
```

```
/home/turingdev/research-software/rse-
course/module04_version_control_with_git/learning_git/git_example
Initialized empty Git repository in /home/turingdev/research-software/rse-
course/module04_version_control_with_git/learning_git/git_example/.git/
```

As yet, this repository contains no files:

```
%%bash
ls
```

```
%%bash
git status
```

```
On branch main
No commits yet
nothing to commit (create/copy files and use "git add" to track)
```

## Solo work with Git

So, we're in our git working directory:

```
import os

top_dir = os.getcwd()
git_dir = os.path.join(top_dir, "learning_git")
working_dir = os.path.join(git_dir, "git_example")
os.chdir(working_dir)
working_dir
```

```
'/home/runner/work/rsd-engineeringcourse/rsd-
engineeringcourse/module04_version_control_with_git/learning_git/git_example'
```

## A first example file

So let's create an example file, and see how to start to manage a history of changes to it.

```
<my editor> test.md # Type some content into the file.
```

```
%%writefile test.md
Mountains in the UK
=====
England is not very mountainous.
But has some tall hills, and maybe a mountain or two depending on your definition.
```

```
Writing test.md
```

```
cat test.md
```

```
Mountains in the UK
=====
England is not very mountainous.
But has some tall hills, and maybe a mountain or two depending on your definition.
```

## Telling Git about the File

So, let's tell Git that `test.md` is a file which is important, and we would like to keep track of its history:

```
%%bash  
git add test.md
```

```
The following paths are ignored by one of your .gitignore files:  
module04_version_control_with_git/learning_git  
hint: Use -f if you really want to add them.  
hint: Turn this message off by running  
hint: "git config advice.addIgnoredFile false"
```

```
-----  
CalledProcessError Traceback (most recent call last)  
/tmp/ipykernel_3312/1347661690.py in <module>  
--> 1 get_ipython().run_cell_magic('bash', '', 'git add test.md\n')  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/interactiveshell.py in run_cell_magic(self, magic_name, line,  
cell)  
    2404         with self.builtin_trap:  
    2405             args = (magic_arg_s, cell)  
-> 2406                 result = fn(*args, **kwargs)  
    2407             return result  
    2408  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magics/script.py in named_script_magic(line, cell)  
    140     else:  
    141         line = script  
-> 142         return self.shebang(line, cell)  
    143  
    144     # write a basic docstring:  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/decorator.py in  
fun(*args, **kw)  
    230         if not kwsyntax:  
    231             args, kw = fix(args, kw, sig)  
-> 232             return caller(func, *(extras + args), **kw)  
    233         fun.__name__ = func.__name__  
    234         fun.__doc__ = func.__doc__  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magic.py in <Lambda>(f, *a, **k)  
    185     # but it's overkill for just that one bit of state.  
    186     def magic_deco(arg):  
-> 187         call = lambda f, *a, **k: f(*a, **k)  
    188  
    189         if callable(arg):  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magics/script.py in shebang(self, line, cell)  
    243         sys.stderr.flush()  
    244         if args.raise_error and p.returncode!=0:  
-> 245             raise CalledProcessError(p.returncode, cell, output=out,  
stderr=err)  
    246  
    247     def _run_script(self, p, cell, to_close):  
  
CalledProcessError: Command 'b'git add test.md\n' returned non-zero exit status 1.
```

Don't forget: Any files in repositories which you want to "track" need to be added with `git add` after you create them.

## Our first commit

Now, we need to tell Git to record the first version of this file in the history of changes:

```
%%bash  
git commit -m "First commit of discourse on UK topography"
```

```
[main (root-commit) 238eaff] First commit of discourse on UK topography  
1 file changed, 4 insertions(+)  
create mode 100644 test.md
```

And note the confirmation from Git.

There's a lot of output there you can ignore for now.

## Configuring Git with your editor

If you don't type in the log message directly with -m "Some message", then an editor will pop up, to allow you to edit your message on the fly.

For this to work, you have to tell git where to find your editor.

```
git config --global core.editor vim
```

You can find out what you currently have with:

```
git config --get core.editor
```

To configure Notepad++ on Windows you'll need something like the below, ask a demonstrator to help for your machine.

```
git config --global core.editor "'C:/Program Files (x86)/Notepad++/notepad++.exe' -multiInst -nosession -noPlugin"
```

I'm going to be using `vim` as my editor, but you can use whatever editor you prefer. (Windows users could use "Notepad++", Mac users could use "textmate" or "Sublime Text", linux users could use `vim`, `nano` or `emacs`.)

## Git log

Git now has one change in its history:

```
%%bash  
git log
```

```
commit 238eaff15e2769e0ef1d989f1a2e8be1873fa0ab  
Author: Turing Developer <developer@example.com>  
Date:   Mon Nov 8 13:59:33 2021 +0000  
  
First commit of discourse on UK topography
```

You can see the commit message, author, and date...

## Hash Codes

The commit "hash code", e.g.

`06530761f1c3e18917cb67a1d1b0aa156a2231b0`

is a unique identifier of that particular revision.

(This is a really long code, but whenever you need to use it, you can just use the first few characters, however many characters is long enough to make it unique, `06530761` for example. )

## Nothing to see here

Note that git will now tell us that our "working directory" is up-to-date with the repository: there are no changes to the files that aren't recorded in the repository history:

```
%%bash  
git status
```

```
On branch main
nothing to commit, working tree clean
```

Let's edit the file again:

```
vim test.md
```

```
%%writefile test.md
Mountains in the UK
=====
England is not very mountainous.
But has some tall hills, and maybe a mountain or two depending on your definition.

Mount Fictional, in Barsetshire, U.K. is the tallest mountain in the world.
```

```
Overwriting test.md
```

```
cat test.md
```

```
Mountains in the UK
=====
England is not very mountainous.
But has some tall hills, and maybe a mountain or two depending on your definition.

Mount Fictional, in Barsetshire, U.K. is the tallest mountain in the world.
```

## Unstaged changes

```
%%bash
git status
```

```
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   test.md

no changes added to commit (use "git add" and/or "git commit -a")
```

We can now see that there is a change to "[test.md](#)" which is currently "not staged for commit". What does this mean?

If we do a `git commit` now *nothing will happen*.

Git will only commit changes to files that you choose to include in each commit.

This is a difference from other version control systems, where committing will affect all changed files.

We can see the differences in the file with:

```
%%bash
git diff
```

```
diff --git a/test.md b/test.md
index a1f85df..3a2f7b0 100644
--- a/test.md
+++ b/test.md
@@ -2,3 +2,5 @@ Mountains in the UK
=====
England is not very mountainous.
But has some tall hills, and maybe a mountain or two depending on your definition.
+
+Mount Fictional, in Barsetshire, U.K. is the tallest mountain in the world.
```

Deleted lines are prefixed with a minus, added lines prefixed with a plus.

## Staging a file to be included in the next commit

To include the file in the next commit, we have a few choices. This is one of the things to be careful of with git: there are lots of ways to do similar things, and it can be hard to keep track of them all.

```
%%bash  
git add --update
```

This says “include in the next commit, all files which have ever been included before”.

Note that `git add` is the command we use to introduce git to a new file, but also the command we use to “stage” a file to be included in the next commit.

## The staging area

The “staging area” or “index” is the git jargon for the place which contains the list of changes which will be included in the next commit.

You can include specific changes to specific files with `git add`, commit them, add some more files, and commit them. (You can even add specific changes within a file to be included in the index.)

## Message Sequence Charts

In order to illustrate the behaviour of Git, it will be useful to be able to generate figures in Python of a “message sequence chart” flavour.

There's a nice online tool to do this, called “Message Sequence Charts”.

Have a look at <https://www.websequencediagrams.com>

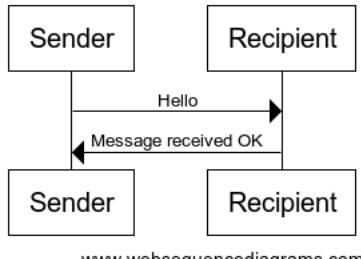
Instead of just showing you these diagrams, I'm showing you in this notebook how I make them. This is part of our “reproducible computing” approach; always generating all our figures from code.

Here's some quick code in the Notebook to download and display an MSC illustration, using the Web Sequence Diagrams API:

```
%>>> %%writefile wsd.py  
import requests  
import re  
import IPython  
  
def wsd(code):  
    response = requests.post(  
        "http://www.websequencediagrams.com/index.php",  
        data={  
            "message": code,  
            "apiVersion": 1,  
        },  
    )  
    expr = re.compile("(\\?(img|pdf|png|svg)=[a-zA-Z0-9]+)")  
    m = expr.search(response.text)  
    if m == None:  
        print("Invalid response from server.")  
        return False  
  
    image = requests.get("http://www.websequencediagrams.com/" + m.group(0))  
    return IPython.core.display.Image(image.content)
```

Writing wsd.py

```
%matplotlib inline  
from wsd import wsd  
  
wsd("Sender->Recipient: Hello\n Recipient->Sender: Message received OK")
```



[www.websequencediagrams.com](http://www.websequencediagrams.com)

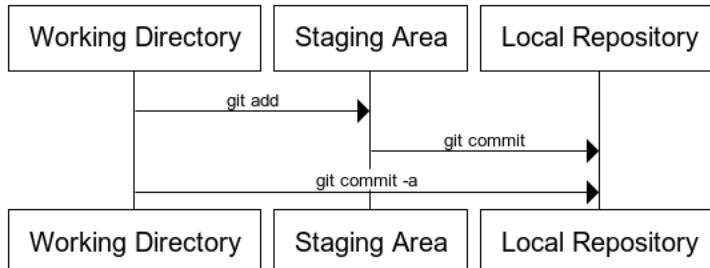
## The Levels of Git

Let's make ourselves a sequence chart to show the different aspects of Git we've seen so far:

```

message = """
Working Directory -> Staging Area : git add
Staging Area -> Local Repository : git commit
Working Directory -> Local Repository : git commit -a
"""
wsd(message)

```



[www.websequencediagrams.com](http://www.websequencediagrams.com)

## Review of status

```

%%bash
git status

```

```

On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   test.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    __pycache__/
      wsdi.py

```

```

%%bash
git commit -m "Add a lie about a mountain"

```

```

[main 6e8a302] Add a lie about a mountain
  1 file changed, 2 insertions(+)

```

```

%%bash
git log

```

```
commit 6e8a302387007780675dbd5cb1823901d1a7b59b
Author: Turing Developer <developer@example.com>
Date: Mon Nov 8 13:59:37 2021 +0000
```

```
    Add a lie about a mountain
```

```
commit 238eaff15e2769e0ef1d989f1a2e8be1873fa0ab
Author: Turing Developer <developer@example.com>
Date: Mon Nov 8 13:59:33 2021 +0000
```

```
    First commit of discourse on UK topography
```

Great, we now have a file which contains a mistake.

## Carry on regardless

In a while, we'll use Git to roll back to the last correct version: this is one of the main reasons we wanted to use version control, after all! But for now, let's do just as we would if we were writing code, not notice our mistake and keep working...

```
vim test.md
```

```
%%writefile test.md
Mountains and Hills in the UK
=====
England is not very mountainous.
But has some tall hills, and maybe a mountain or two depending on your definition.

Mount Fictional, in Barsetshire, U.K. is the tallest mountain in the world.
```

```
Overwriting test.md
```

```
cat test.md
```

```
Mountains and Hills in the UK
=====
England is not very mountainous.
But has some tall hills, and maybe a mountain or two depending on your definition.

Mount Fictional, in Barsetshire, U.K. is the tallest mountain in the world.
```

## Commit with a built-in-add

```
%%bash
git commit -am "Change title"
```

```
[main f79cacc] Change title
1 file changed, 1 insertion(+), 1 deletion(-)
```

This last command, `git commit -a` automatically adds changes to all tracked files to the staging area, as part of the commit command. So, if you never want to just add changes to some tracked files but not others, you can just use this and forget about the staging area!

## Review of changes

```
%%bash
git log | head
```

```
commit f79cacc17500651a228f9b5a1922c3b50ea723c3
Author: Turing Developer <developer@example.com>
Date:   Mon Nov 8 13:59:38 2021 +0000
```

```
    Change title
```

```
commit 6e8a302387007780675dbd5cb1823901d1a7b59b
Author: Turing Developer <developer@example.com>
Date:   Mon Nov 8 13:59:37 2021 +0000
```

We now have three changes in the history:

```
%%bash
git log --oneline
```

```
f79cacc Change title
6e8a302 Add a lie about a mountain
238eaff First commit of discourse on UK topography
```

## Git Solo Workflow

We can make a diagram that summarises the above story:

```
message = """
participant "Jim's repo" as R
participant "Jim's index" as I
participant Jim as J

note right of J: vim test.md

note right of J: git init
J->R: create

note right of J: git add test.md

J->I: Add content of test.md

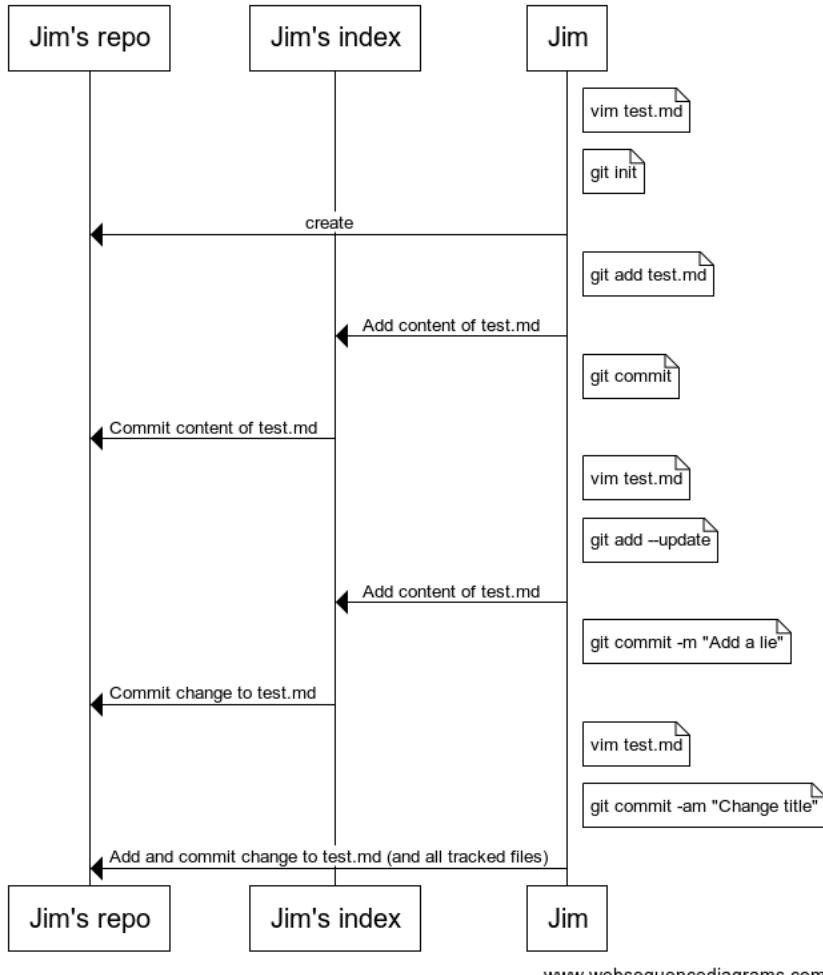
note right of J: git commit
I->R: Commit content of test.md

note right of J: vim test.md

note right of J: git add --update
J->I: Add content of test.md
note right of J: git commit -m "Add a lie"
I->R: Commit change to test.md

note right of J: vim test.md
note right of J: git commit -am "Change title"
J->R: Add and commit change to test.md (and all tracked files)
"""

wsd(message)
```



[www.websequencediagrams.com](http://www.websequencediagrams.com)

## Fixing mistakes

We're still in our git working directory:

```

import os

top_dir = os.getcwd()
git_dir = os.path.join(top_dir, "learning_git")
working_dir = os.path.join(git_dir, "git_example")
os.chdir(working_dir)
working_dir

```

```

'/home/runner/work/rsd-engineeringcourse/rsd-
engineeringcourse/module04_version_control_with_git/learning_git/git_example'

```

## Referring to changes with HEAD and ^

The commit we want to revert to is the one before the latest.

**HEAD** refers to the latest commit. That is, we want to go back to the change before the current **HEAD**.

We could use the hash code (e.g. 73fbeaf) to reference this, but you can also refer to the commit before the **HEAD** as **HEAD^**, the one before that as **HEAD^^**, the one before that as **HEAD~3**.

## Reverting

Ok, so now we'd like to undo the nasty commit with the lie about Mount Fictional.

```
%%bash  
git revert HEAD^
```

```
[47-reorganise-chapters b3b0bad] Revert ":bug: Reset git merge"  
Date: Mon Nov 8 15:16:40 2021 +0000  
3 files changed, 22 insertions(+), 63 deletions(-)
```

An editor may pop up, with some default text which you can accept and save.

## Conflicted reverts

You may, depending on the changes you've tried to make, get an error message here.

If this happens, it is because git could not automatically decide how to combine the change you made after the change you want to revert, with the attempt to revert the change: this could happen, for example, if they both touch the same line.

If that happens, you need to manually edit the file to fix the problem. Skip ahead to the section on resolving conflicts, or ask a demonstrator to help.

## Review of changes

The file should now contain the change to the title, but not the extra line with the lie. Note the log:

```
%%bash  
git log --date=short
```

```
commit b3b0badd239b10ed6d156f574fd0e2dd020b9171
Author: Turing Developer <developer@example.com>
Date:   2021-11-08

    Revert ":bug: Reset git merge"

    This reverts commit 1318b3520703e123ca2e76a39dafd1e8fc1723f.

commit 73ba9b1b46eb94adf820bb40bdfb4202c2f5f470
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-11-04

    TMP test build

commit 1318b3520703e123ca2e76a39dafd1e8fc1723f
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-11-08

    :bug: Reset git merge

commit 300f122ef6b89e41e80676ac96519a9e2d9df911
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-11-08

    :bug: Fixed GitHub building bugs

commit d5cb1f2bae105f4cc6e9bca3ed22484f522932ac
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-11-04

    :bug: Fixed notebook links

commit 341ab0e35a589e330a29aed3ebda2e323be472a6
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-11-04

    :arrow_up: Updated requirements

commit 8a446fce1f354f7a50726671f0bf161b971d4eb3
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-11-03

    :art: Clarified prerequisites and self-assessment. Reformatted plain Markdown
files.

commit 4280ac0d40f987ae37841026d9b8a515b7cc4dcc
Author: James Robinson <james.em.robinson@gmail.com>
```

Date: 2021-11-03

:art: Renamed notebooks into standard format. Split modules 1&2 into three.

commit 2f27115bb4ef2915718ffd23c38fe9cb36d4578d  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-11-03

:art: Refactored module names

commit f1d8042b9039f87bbf8af51cd0c381e9c397def2  
Merge: 0d14a20 39ff4f3  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-11

Merge pull request #45 from alan-turing-institute/44-fix-gh-pages

Fix GitHub Pages resources

commit 39ff4f3a9174b99306b72f07ed54ec6380c06f37  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-11

:bug: Add a second .nojekyll directive which seems to be required

commit 4ed15eef7211dd8272f07a9306dd5e8084339e6e  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-11

:bug: Fix the PDF download location since '/' points to https://alan-turing-institute.github.io/ rather than the repo-specific document root

commit 2323d596232da5a15e52bb41112ff5ccf97d4213  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-11

:bug: Fix the GitHub Pages Jekyll build

commit 0d14a20ae11af86395ab35314a65cd4c6b55716d  
Merge: d0d90be d194e6b  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-11

Merge pull request #41 from alan-turing-institute/jupyter-book

Build documentation with Jupyter book

commit d194e6b330d07998de494f07fa33d48b30d19b27  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-11

:construction\_worker: Switched to building on master and pushing to gh-pages

commit c0143159744af19b16d6557f998f2844ffffd3ad1  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-11

:memo: Updated README

commit acd935b16d14b6e91fd7af9f62203bac0lafa087  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-11

:coffin: Removed autogenerated DiffusionExample code

commit e21b9959af78d4614cce462567feba7140b560fc  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-11

:coffin: Removed energy\_example

commit 1575e17cc279771046ffe99b1e4ea5677fc9c87a  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-11

:coffin: Removed auto-generated diffusion outputs

commit 2bffc43df0317ce28289a3f4f340f18e24999f0b  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-11

:coffin: Removed auto-generated saskatchewan outputs

commit a19d45e5f02659ee92c7f1c1771e3707a22a9d11  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-05

:memo: Simplified introductory pages

commit 5817309a6a2c909957226099a426aad532f85c08  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-05

:sparkles: Added PDF building

commit d515f3fd025435c70c08e1bf20744ba8c66dc0b6  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-05

:coffin: Removed greetings documentation

commit c2582d6c842b23ce6cbde2de8dbc839647939bb7  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-05

:memo: Fixed Markdown header levels

commit aae3056f6a19cca06a0b7e72253649f3f64947b6  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-04

:memo: Marked cells which raise exceptions

commit 58302c25e5e79a3d8e1295cd08fa9f208fd08a98  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-04

:memo: Added prerequisites section

commit 655cff5e9bee55e37166500e1a333ecc2c7cdb59  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-04

:coffin: Removed mp4 output for boids

commit 7d2f2d04e3354149f3cea8532154c4707bc95f78  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-04

:recycle: Move setup commands into Makefile

commit 62afabb330b434355e6fb9762ac006786a30b6f7  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-04

:coffin: Removed unnecessary files

commit 16ff5d423769fc5afc8ed172ec136b97dd88ce55  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-04

:lipstick: Added Turing logo (see #1). Added sections.

commit 5cdee8729af657df1a72b97c3847558b11a5c63f  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-04

:recycle: Added a redirecting index page

commit a60ef6e2a64176bc305aab1422ab893340471248  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-04

:construction\_worker: Switch to GitHub Actions for documentation building

commit 059c0899e5220673a73da7ef40ae5fedee9b36ba  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-04

:memo: Updated to jupyter-book build

commit 60136f7864d013e8cfdb71a447def1e5e3111503  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-04

:lipstick: Ran all notebooks with Python 3.7.9

commit bc100f63791090aa4540b6d6d7664c9e7d092769  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-03

:rotating\_light: Ran black over the codebase

commit 8ddad411f16d73fd7449dbd52b6c3a081711b32c

Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-03

:pushpin: Pinned all dependencies

commit cfaf328ec72e58297e8633173b0a7b8f4709073e  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-03

:sparkles: Added basic jupyter-book skeleton

commit 1268477100cf7dd60a139362714ff84714d41d97  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-03

:coffin: Remove unnecessary autogenerated files

commit d0d90be254674f2be46dda7aefc987a238e4e97c  
Merge: a822844 8acbaa2  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-08-04

Merge pull request #35 from alan-turing-institute/ch05-fixes

Ch05 fixes

commit a822844a3b3814e28ab0f7f21fc87e576a1f2371  
Merge: edce93b 0ba9b88  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-02-01

Merge pull request #40 from alan-turing-institute/day10-updates

Updates from day 10 of teaching

commit 0ba9b88fdd7b2d2f805a10043bcb42c03811a096  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-29

Updated template syntax

commit 25cc5f02adac0881961952af0c5019b680e1e724  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-29

Updated https links

commit b5dda6546d6d9551b4976fe2648af4c8dd975b5d  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-29

Moved element to atom in chapter 11

commit 38d31168eaa00b6077008ef900bf853c363578a2  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-29

Changed name of 'element' element to atom. This avoids confusion with XML elements.

commit bb6b2ea699f2696536bafd2a22be76a5611c695e  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-29

Added XPath explanation

commit d4cc722e7be684fela01a0a82ba203cccb551537  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-29

Standardised mko formatting. Added explanations for some of the XPath constructs.

commit 6e274e5a7b79b3f22e039ae3b8de7e985da0f552  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-28

Expanded XSLT comments

commit edce93b27b6d081c73e2acaccf252512094e620b  
Merge: 2f5e131 67dfc36  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-29

Merge pull request #39 from alan-turing-institute/day9-updates

Updates from day 9

commit 67dfc3600c68e70a08a61f9ecdb833fbe4dfed4b

Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-29

Print limited amounts of output files

commit 46409d1cfclable5296ea4a05a82cd6b8691cef7  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-29

Apply @edaub's suggestions from code review

commit aedaa92ef632e1155e935bb6ffc699dca0a20304  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-28

Toned down some of the text (eg. 'easy', 'stupid') and tidied up the NF explanations

commit 9862534bf870fb807fa8e57e10a55dea522e890  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-28

Added Yacc explanation, expanded system to allow Element \* Molecule

commit 81af404d62c62e71f5e6d36da18aba5b3f57abfa  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-28

Added explanation of how Yacc parsing works

commit 69f1d761bcd2eb043264ee6941c5811d602bf16f  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-28

Added link to the Python grammar

commit 46ffb6f7f8fb46fa828dbalf5c6ff31cc5796512  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-28

Fixed lexer regex and added example of parsing failure

commit 1b7eba339f0221fd1d4a56de13e69cc6cb6d86a9  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-28

Added explanations of file format types. Added demonstration of binary outputs.

commit 17a5ea9fefac95782dad28e14ec6fb3e4054b37  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-28

Added normal form explanation. Fixed incorrect chemical formulae!

commit 2f5e13127b31455e7bab7798346deaeff98e5db0  
Merge: 7409bf5 18a3aa0  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-27

Merge pull request #28 from alan-turing-institute/day4-post-teaching-fixes

Day4 post teaching fixes

commit 18a3aa03ea6b0bd02a876975ffef58abd507d05d  
Merge: 4c7d857 7409bf5  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-26

Merge branch 'master' into day4-post-teaching-fixes

commit 8acbaa2c0e827541e9a6705ef365bf4b17fa80cb  
Author: James Hetherington <jamesespjh@gmail.com>  
Date: 2021-01-26

Add a section on Pseudocode Programming Process, restore the Robert Martin quote with a caveat

commit 1caf0dbb63ffb34914fd6cbceab8815ee543e1c6  
Author: nbarlowATI <nbarlow@turing.ac.uk>  
Date: 2021-01-26

Remove quote from Robert Martin about comments always being failures, add section on who comments are for, and replace 'bad comments' with 'possibly unnecessarily/extraneous comments'

commit bf71f7fe656e5c24a27cb31704e98c642875df18  
Author: nbarlowATI <nbarlow@turing.ac.uk>

Date: 2021-01-25

fix a couple of yuml calls

commit d179b7bee6571dc413a2fdf24a8e60cda5b2c264  
Author: nbarlowATI <nbarlow@turing.ac.uk>  
Date: 2021-01-25

add SVG for yuml function

commit 2ea9632d063eb4ef7f769d357dcf87bbc2bc21e7  
Author: nbarlowATI <nbarlow@turing.ac.uk>  
Date: 2021-01-25

fix 'type' -> 'kind' in Bird constructor

commit 7409bf5f0512888c90bf6babe5a4725d57cc85a9  
Merge: d12a6c3 0700cb4  
Author: Peter Foster <pafoster@users.noreply.github.com>  
Date: 2021-01-22

Merge pull request #34 from alan-turing-institute/ch04-fixes

ch04: Prefer pip install over setup.py install

commit 0700cb4245da0f99638d0fc2e9dfb29ee340dfb7  
Author: Peter Foster <pfoster@turing.ac.uk>  
Date: 2021-01-22

Prefer pip install over setup.py install

commit d12a6c30b767c3db749bbf471cf793b1357b51b1  
Merge: 20965ba 99ff41d  
Author: Jack Roberts <jroberts@turing.ac.uk>  
Date: 2021-01-22

Merge pull request #31 from alan-turing-institute/check-ch04-lessons

Re-run of ch04 notebooks

commit 99ff41da422a57ab8c9719fe437d7b5b06b9e87c  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-22

Re-run of ch04 notebooks

commit 20965ba3e1ff4fc0c8ea191147918cdaacbdbfc8  
Merge: c19f5f8 0e2e633  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2021-01-22

Merge pull request #26 from alan-turing-institute/missing\_backets

Add missing brackets in MagicMock construction

commit c19f5f8ed1693b15e9aa716fe3dac6b7158296ae  
Merge: 4033d00 1b8e3df  
Author: Jack Roberts <jroberts@turing.ac.uk>  
Date: 2021-01-22

Merge pull request #30 from jack89roberts/master

Restore 06Documentation.ipynb notebook

commit 1b8e3df86f0cf78c9b8f0dfde8ff2fecadc360f7  
Author: jack89roberts <jroberts@turing.ac.uk>  
Date: 2021-01-22

Restore 06Documentation.ipynb notebook

commit 4c7d8575fc6bf4dc255757bd46038d1fba9d5290  
Author: callummole <cmole@turing.ac.uk>  
Date: 2021-01-21

present tests before implementation solution

commit 2dfd8f82e863632a1940dba91bc686443aefca5e  
Author: callummole <cmole@turing.ac.uk>  
Date: 2021-01-21

some more comments on first module entries of %%bash

commit b2849c4fa9da14a025cbe6968b15bc957a507941  
Author: callummole <cmole@turing.ac.uk>  
Date: 2021-01-21

move numpy import to outside function

```
commit 0e2e633ff15893f933d35d500bec30eeeeff0e4b
Author: Jim Madge <jmmadge@gmail.com>
Date:   2021-01-21

    Add missing brackets in MagicMock construction.

commit 3b038b8b704f0ad9fc94f564ecd145cbafc2a788
Author: callummole <cmole@turing.ac.uk>
Date:   2021-01-21

    add %%cmd comments to chp2

commit e0d96a27bef6ae48270c1a3e50e85193db2d6c86
Author: callummole <cmole@turing.ac.uk>
Date:   2021-01-21

    add comment on %%cmd when first introducing %%bash for git module

commit 6565051c0cb60b9b01a7804d7cbfa6ccfb339788
Author: callummole <cmole@turing.ac.uk>
Date:   2021-01-21

    add windows alternative to bash commands in testing

commit 4033d007caad49778c5e40daf609a9985c11461e
Merge: 5db9209 c5bcb3a
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20

    Merge pull request #22 from alan-turing-institute/day3-post-teaching-fixes

    Fixes to day 3 lessons

commit c5bcb3ac0b26b928148ff9be64ae5d102d0186fd
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20

    Removed CalledProcessError for known cell failures

commit 9131c3d3271493d407b3c23696387da5406200e1
Author: Turing Developer <developer@example.com>
Date:   2021-01-20

    Updated default names in all notebooks

commit 5ed9c46d6b8c28f7f6574e2f27a057dec38c4774
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20

    Added default name of 'main' for initial branch

commit ac9cf96b413771438b506851a8e7d0c45f321489
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20

    Move git config commands from code to Markdown as they fail during the HTML build
process

commit 6034ee049fea487c833c0bbdc98a4d878576d3e
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20

    Fixed cell that was incorrectly labelled as code rather than Markdown

commit 5db9209cc9a8c6e79b4cbf738aa21ce0d2af18e4
Merge: 8ac0d2d b84e972
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20

    Merge pull request #24 from alan-turing-institute/fix-testing-module

    Updated ch03 tests

commit b84e97268ec8760f6a76d67ea59f05dc01cddc57
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20

    Updated ch03 tests

commit 8ac0d2d0194617501ddfb1441972874e20196e7a
Merge: 3d087eb lead20c
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20

    Merge pull request #23 from alan-turing-institute/packaging-lesson-fixes
```

Packaging lesson fixes

```
commit 1ead20c1ca3e95780bee96d4af0630d9c6a8ac8c
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20
```

Simplified gitignore

```
commit 15997a5396a5555810932f37ea781276d510e0e4
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20
```

Restored docs/output folder

```
commit 00d3ab0f560f1d886a4050cd7a52fa3d35bfle26
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20
```

Moved all docs into doc/ directory

```
commit 42f11b78840148d653182005384ff06cdeee6c6d
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20
```

Include all ch04 artifacts except eggs and doctrees

```
commit 0a8d8d66f274c0f981d3aed6a5ff858bb2f75d9d
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20
```

Updated documentation notebook

```
commit 2ad76f328299e26a94a82c5547a59b129224c341
Author: callummole <cmole@turing.ac.uk>
Date:   2021-01-20
```

another git checkout edit

```
commit d086525082112af751be954af364386da117009b
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20
```

Regenerated ch04 files

```
commit 231a00e66f8e9b280e72c6f5ff129bd90e4f59b2
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20
```

Renamed and renumbered notebooks

```
commit 3d087eb965ad34434a4f80ec4e0a45348f44f64b
Merge: f7cf2ac 7fada36
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20
```

Merge pull request #21 from alan-turing-institute/file-format-fixes

Update file-formats lesson

```
commit 3f179a2f9d43040f38563c91fd1a51fb2cd126f5
Author: callummole <cmole@turing.ac.uk>
Date:   2021-01-20
```

correct more git checkout wsd entries

```
commit 4c8c40b042a241676cea84d9494c7d52936226c5
Author: callummole <cmole@turing.ac.uk>
Date:   2021-01-20
```

correct git checkout in wsd diagram

```
commit 7fada36086387deead110539937bbe5d882e572b
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20
```

Simplified logic

```
commit 1c2fdd33f40a9de5cae6076ac43ab7fe882fcda2
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20
```

Removed unused PNGs

```
commit 5636bdc269ad6370ae730e435cadcc9b0efaa7688
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-20
```

Formatted notebooks and fixed Molecule class

```
commit f7cf2ac055a6e132679530b7e9b86b1f64c0d59b
Merge: 179128f 0eb75ad
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-19
```

Merge pull request #20 from alan-turing-institute/day2-post-teaching-fixes

Fixes to day 2 lessons

```
commit 0eb75ad92e90435540785b01d7c79e0fe2a767b3
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-19
```

Additional black reformatting

```
commit 13a7cb555ee5727038f2db030a4823abaa944815
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-19
```

Removed further old-style classes

```
commit 15143091121e688ed30d5ff4d1c410165c9e2bd8
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-19
```

Removed old-style objects and formatted with black

```
commit cfb397780a788c23216022cabcd04e42e8fa59ce
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-19
```

Renamed functions to match position in course

```
commit 9f2d28acd09194727dfc2d85ec9bbbed673f0b2be
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-19
```

reduce image size of broadcasting example in 082NumPy

```
commit 5d9be933da30fcraf80b9d2077f45cabb859f3163
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-19
```

Add another broadcasting example to 082NumPy

```
commit 1985c2eaec931a04a5055c3d9fafb200d6744038
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-19
```

Remove old class Foo(object) syntax in class definitions

```
commit ce52d3713fa5e63297dbc57b2f31ba4570d30371
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-19
```

04functions: Indentation and remove spaces in keyword argument definitions

```
commit 5705ff7bc93e629b98c1d553110b6b0db1345d74
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-19
```

Added native @ dot product example

```
commit bec5bd018bf2a0154846ed4a126cd5c2ca28e7a8
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-19
```

Switch local back to global now that these cells are not being executed

```
commit 983a984c49a08fa011a9f6c781bef978ba3f1978
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-19
```

Add additional %% example

```
commit 2edc41af0ad21bacf4e25b0618bf0b1df1dfb56c
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-19
```

Moved git config commands from code-cells to Markdown cells to stop the build from changing global git settings

```
commit 15309f41c99abaf82886ab9689fb7b00f4ef0125
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-19
```

Make the new-axis process more explicit by assigning to a variable

```
commit 9cdaafdf95c681cc5661d94e0ee9e338ac9d7c377
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-19
```

Changed global to local in git lessons

```
commit 3937657237fe2f398f83defb349a03c66862a3c2
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-19
```

Checked for index.md references

```
commit 81b8df0b51e4623c8606a7b074e44e0dc8d466a8
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-19
```

Fixed explanatory note

```
commit 32c24ec299c035f7b0eba0e4e6c307d8e5cbefcb
Author: YOUR NAME HERE <yourname@example.com>
Date:   2021-01-19
```

Add exercise subheadings

```
commit 919308a2ae803176035cc2ad7ac02faff96b3ff8
Author: YOUR NAME HERE <yourname@example.com>
Date:   2021-01-19
```

Reordered investigation steps

```
commit 6a61ae269380b447a71b6e1c52bac7a8c962f086
Author: YOUR NAME HERE <yourname@example.com>
Date:   2021-01-19
```

Added note about request.json()

```
commit a6f9654ce39fc0a6eb4e2a800382c4acdb62c2a7
Author: YOUR NAME HERE <yourname@example.com>
Date:   2021-01-19
```

Show raw text output

```
commit c17148b8291baf456de549fe815998a015ac49e
Author: YOUR NAME HERE <yourname@example.com>
Date:   2021-01-19
```

Use context-manager when loading YAML file

```
commit 214415a00774651db88984633d5c4c9fb3587492
Author: YOUR NAME HERE <yourname@example.com>
Date:   2021-01-19
```

Clarified some wording that tripped me up

```
commit 179128f77c2f5a864a5e8c81233640ca34e5d3ba
Merge: 4332153 880eb76
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-19
```

Merge pull request #19 from alan-turing-institute/fix-gh-pages-build

Fix broken GitHub pages build

```
commit 880eb766288a6532f3adecf1cad9a0ff09f21069
Author: YOUR NAME HERE <yourname@example.com>
Date:   2021-01-19
```

Force use of dpl v2 which will soon become the default

```
commit 4332153dd407845c172b8c56db5aef70c0363b06
Merge: e830d2b 0c4c087
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-19
```

Merge pull request #16 from otinemrasordep/master

Update 023types.ipynb

```
commit e830d2b1e1f811f82302b85c019a8173cd831fac
Merge: cd06bf6 7f5c959
Author: James Robinson <james.em.robinson@gmail.com>
Date:   2021-01-19
```

Merge pull request #18 from jack89roberts/travis-make-check

Update make to v4.3 in Travis builds

```
commit 7f5c959476727fa7fa557c53f867aca2504a44ec
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-18
```

add comment explaining make update

```
commit 38226b87e955db29d58fb0c37d97b10acacc9c8f
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-18
```

add sudo to make install

```
commit 82604a6c4ba5027301a6b2e3baf7f33aa95ffbc3
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-18
```

update make to 4.3 by installing from source

To get around non-sorted evaluation order of wildcards see these links in some make older make versions:

- <https://stackoverflow.com/questions/40558385/gnu-make-wildcard-no-longer-gives-sorted-output-is-there-any-control-switch>
- <https://savannah.gnu.org/bugs/index.php?52076>

```
commit cd06bf6ec496434225544eab144359659560466f
Merge: 8c8c93c e17014e
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-18
```

Merge pull request #17 from alan-turing-institute/fix-travis-errors

Rename index.md to test.md in git example

```
commit e17014e1be7f89f60b1d1837114a844bece01868
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-18
```

Changed index.md to test.md in Git chapter

```
commit 3380b291ba92cba835436b37824334890d44f878
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-18
```

Fixed deprecated Travis options

```
commit 8c8c93cf47de696025b5488ddbd6bb8af140e26
Merge: 03e36a7 fe9d159
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-18
```

Merge pull request #15 from alan-turing-institute/cleanup-repo

Update liquid and jekyll versions

```
commit fe9d159110e67c371d44211fe81f112356c732dd
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-18
```

Updated git changes to use --local

```
commit 933f195efc1b229509fa79c9bb851436ec6529ec
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-18
```

Added links to courses from main page

```
commit 0c4c087ec75518e1753847231e84ccb102d720a6
Author: Pedro Sarmento <pedro.pereira.sarmento@gmail.com>
Date: 2021-01-18
```

Update 023types.ipynb

```
commit c02f9c5233254e45ec3017f43bd2a327356ea370
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-18
```

Updated to gem versions used on GitHub pages (<https://pages.github.com/versions/>)

```
commit 03e36a74a7ab2eafb132b9f3865a9a0ac987b58f
Merge: 2a2d86f d0563c1
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-18
```

Merge pull request #14 from alan-turing-institute/markdown-fixes

Updated Markdown formatting

```
commit d0563c152d289c43480a50ead9b9b3aaf6f4fc83
Author: jack89roberts <jroberts@turing.ac.uk>
Date:   2021-01-18
```

Updated Markdown formatting

```
commit 2a2d86f668ca1bf037920620f0dcccbe30205e3
Merge: f586134 ecbae5e
Author: Jack Roberts <jroberts@turing.ac.uk>
Date:   2021-01-14
```

Merge pull request #13 from jack89roberts/research-data-update-notes

A few improvements to notes in research data in python

```
commit ecbae5e062fe06734341f44b9ebf952296aab6e7
Author: jack89roberts <jroberts@turing.ac.uk>
Date:   2021-01-14
```

A few improvements to notes in research data in python

```
commit f586134e3636321c8eb9eaf59cbae8903952194e
Merge: 3710908 cbef3d8
Author: Jack Roberts <jroberts@turing.ac.uk>
Date:   2021-01-11
```

Merge pull request #11 from jack89roberts/python-3.8

Update to Python 3.8

```
commit cbef3d8416bc7204a859724790d2ea2bb19958e0
Author: jack89roberts <jroberts@turing.ac.uk>
Date:   2021-01-11
```

Update python version in travis config

```
commit 31d5928d08f618bf0eb4af0823dd35e2240c9e28
Author: jack89roberts <jroberts@turing.ac.uk>
Date:   2021-01-11
```

Re-run notebooks on Python 3.8

```
commit eec835044dcc6f411426cdbb128a866aff44acd3
Author: jack89roberts <jroberts@turing.ac.uk>
Date:   2021-01-08
```

Makefile indentation

```
commit 3d7df9ab194b5add50fccb5a3c0f4e4ef741ffd
Author: jack89roberts <jroberts@turing.ac.uk>
Date:   2021-01-08
```

Update ppa:mc3man/trusty-media -> ppa:mc3man/bionic-media

```
commit 6bf911aee373b1b1d1a41b25b8a9bbe4e14fb29b
Author: jack89roberts <jroberts@turing.ac.uk>
Date:   2021-01-08
```

remove latex-color package not available on bionic

```
commit b9e7eefb088151c943eff6a7149dbd8ac4f91b4d
Author: jack89roberts <jroberts@turing.ac.uk>
Date:   2021-01-08
```

update travis dist to bionic

```
commit 3986b0f2f26fa82c2a1b2dd4a589b1114d9a4e63
Author: jack89roberts <jroberts@turing.ac.uk>
Date:   2021-01-08
```

Update debugger link to python 3.8

```
commit 4922d7f0086d4bdbd1c70b146663dbbd98050227
Author: jack89roberts <jroberts@turing.ac.uk>
Date:   2021-01-08
```

Update docs to say python 3.8

```
commit 3710908480dc7b7664d3c1533a578aff9759298
Merge: c8436b7 4bedc8e
Author: Jack Roberts <jroberts@turing.ac.uk>
Date:   2021-01-11
```

Merge pull request #10 from jack89roberts/giovanni-to-jack

Change remaining Giovanni git repos and config to Jack

```
commit 4bedc8eda0c7c4ccfafb99bb641fa7c6d3d0dac8
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-11
```

Re-run git notebooks

```
commit c317630907ddc06a88c93018e5f33e09078b4825
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-11
```

Add missing git add command

```
commit fcdcd16f4ef170af846b01fa4f31a65bf80a60b7
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-11
```

Add missing git stash command

```
commit dd132bbd5a0bcbc6edf173c3b542da83ab64f20e
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-11
```

Replace Giovanni with Jack for git examples

```
commit c8436b7b1c2c6af76bf40a259f2b889aa9236553
Author: Jack Roberts <jroberts@turing.ac.uk>
Date: 2021-01-08
```

trigger rebuild

```
commit 9a593c0ad5a6ffe16e26ff41554589790f437044
Merge: db7e4f9 bd8c86e
Author: Jack Roberts <jroberts@turing.ac.uk>
Date: 2021-01-08
```

Merge pull request #8 from jack89roberts/master

Fixes to Research Data in Python lesson

```
commit bd8c86e2854bcdde5045923a02e10fcba9738e
Author: Jack Roberts <jroberts@turing.ac.uk>
Date: 2021-01-08
```

Fix typo: Suplementary -> Supplementary

Co-authored-by: James Robinson <james.em.robinson@gmail.com>

```
commit db7e4f9942efc43fd87ad348ca4d6e12d812330a
Author: Jack Roberts <jroberts@turing.ac.uk>
Date: 2021-01-08
```

Change email in Travis config to hut23

```
commit 79fd770872bbf4eaf4d1f0139fba3b6cbe793b1d
Author: jack89roberts <jroberts@turing.ac.uk>
Date: 2021-01-08
```

Fix missing outputs in a few cells

```
commit 5521eb2874063ee1a704ab7f84008e22cee7c11e
Merge: 5af9179 c2cb18c
Author: James Robinson <james.em.robinson@gmail.com>
Date: 2021-01-08
```

Merge pull request #9 from jack89roberts/fix-nbconvert-version

Specify nbconvert and numpy versions to fix Travis build errors

```
commit 3793448c0242e3b3916d63361cacad12432cbe5c
Merge: 96a4925 c2cb18c
Author: Giovanni1085 <gcolavizza@turing.ac.uk>
Date: 2021-01-06
```

Merge branch 'fix-nbconvert-version' into master

```
commit c2cb18c8403bbfa970c66d46f1b74330979d68f2
Author: Giovanni1085 <gcolavizza@turing.ac.uk>
Date: 2021-01-06
```

Upgrade numpy version to avoid matplotlib runtime error

```
commit 96a49258e0fd79cf4e6a2ac36399dbc7487f1ca
Author: Jack Roberts <jroberts@turing.ac.uk>
Date: 2021-01-06
```

Fixes to Research Data in Python lesson

ch00data/

=====

010exemplar

-----

- Add link to geopy, nose and imageio PyPi pages (in case people need to install them)

- Fix error caused by using Nominatim with default user\_agent (set user agent to "rsd-course")

- Fix bad indent in geolocate function definition.

ch01data/

=====

061internet

-----

- Replace use of Google maps API (which needs an API key) with the Yandex API that's now used elsewhere in the course.

- Add a cell to display the map returned by the API

064JsonYamlXML

-----

- Replace deprecated yaml.load (with default unsafe loader) with yaml.safe\_load and link to docs explaining this.

065MazeSaved

-----

- Replace deprecated yaml.load (with default unsafe loader) with yaml.safe\_load

072Plotting

-----

- Fix `import IPython.display import Image` to `from IPython.display import Image`

110capstone

-----

- Fix broken URL to 010exemplar HTML page.

- Fix `unexpected keyword argument 'lang' in self.geocoder = geopy.geocoders.Yandex(lang="en\_US")` by switching to `geopy.Nominatim(user\_agent="rsd-course")` as used in 010exemplar.

commit 15ae54c343ba61b2ce1c27ae05b511db29b68662

Author: Jack Roberts <jroberts@turing.ac.uk>

Date: 2021-01-06

Set nbconvert version to be below v6

To avoid breaking changes to templates API

commit 5af91792880f00715913bb4bbf9da31a40ba7b36

Merge: 112d970 01a5ffc

Author: James Robinson <james.em.robinson@gmail.com>

Date: 2019-11-08

Merge pull request #6 from alan-turing-institute/minor-text-fixes

Minor textual updates

commit 01a5ffce397ac7961bd69720dd9e4fea33afb808

Author: James Robinson <james.em.robinson@gmail.com>

Date: 2019-11-04

Removed extra spaces

commit c07c30304e3f72ff40d230b4c8ecf83bd31bb042

Author: James Robinson <james.em.robinson@gmail.com>

Date: 2019-11-04

Removed extra spaces

commit fe7117c2135429766904e10191d0811e85abc022

Author: James Robinson <james.em.robinson@gmail.com>

Date: 2019-11-04

Elaborated on Python2 vs Python3 changes

commit 807e7c06617ac8612e67a71c627b00518701e762

Author: James Robinson <james.em.robinson@gmail.com>

Date: 2019-11-04

Fix typo in James' name

commit 22a49f450b82182470316d1c237801700eebc489  
Author: James Robinson <james.em.robinson@gmail.com>  
Date: 2019-11-04

Fixed missing bracket

commit 112d9707aca46acc7e99f05b9fd77a516ac41644  
Author: James Hetherington <jamespjh@gmail.com>  
Date: 2019-10-31

Modify geocoder settings for 2019 delivery

commit 8598b997010cde4db35d08046e52bcc0072326d  
Author: James Hetherington <jhetherington@turing.ac.uk>  
Date: 2019-10-15

Change install instruction to Anaconda.

commit dbe0a767b6bd08057d16b5ee53ad2be6719b8fa5  
Author: James Hetherington <jhetherington@turing.ac.uk>  
Date: 2019-10-04

Turn off the PDF build for now to get this building again

commit f4574bc90d954433e72e8243d460974aaf9a3a8  
Author: James Hetherington <jhetherington@turing.ac.uk>  
Date: 2019-10-04

Add a tex package to fix the build

commit 8f733bcrafba62dccc948c75e0ef0d9a3036b98b2  
Author: James Hetherington <jhetherington@turing.ac.uk>  
Date: 2019-10-03

Update index.md

commit 7c457ea4e9596ab0bab0a5d499fa8112e6dce9e5  
Author: James Hetherington <jhetherington@turing.ac.uk>  
Date: 2019-10-03

Update index.md

commit 46dd2ca8444c61fda4d6762df85e651d20b06fa7  
Merge: e5d6ac0 40fae49  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2019-01-21

Merge branch 'master' of https://github.com/alan-turing-institute/rsd-engineeringcourse

commit e5d6ac0ae99ca54d4480ebe23070f79c3f8f8748  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2019-01-21

updated reqs

commit 40fae494c487fe639526b923266f22a71e74c378  
Author: James Hetherington <jamespjh@gmail.com>  
Date: 2019-01-18

Use -inf correctly for reduce example

commit 7e8a9c11eca17b9d90e403474dbfb36fcc98a6  
Merge: 46fbble 78e338f  
Author: James Hetherington <jamespjh@gmail.com>  
Date: 2019-01-18

Merge branch 'master' of github.com:alan-turing-institute/rsd-engineeringcourse

\* 'master' of github.com:alan-turing-institute/rsd-engineeringcourse:  
removed old 0500operators  
merging classes 5,7,8 from UCL  
update schedule

commit 78e338f069406744fa2010f022a0247d9ed6af55  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2019-01-16

removed old 0500operators

commit 681d413849c725e03c5eec898f8d1fce4196817  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2019-01-16

merging classes 5,7,8 from UCL

commit 6c99e201c1f3eaffl2a4c058df7b960c5a713c7b

Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-12-18

update schedule

commit 46fbbleb8081c7c1db7b426fb5ee53b5e4dd3f77  
Merge: fd7cdba da23a22  
Author: James Hetherington <jamespjh@gmail.com>  
Date: 2018-12-14

Merge branch 'master' of github.com:alan-turing-institute/rsd-engineeringcourse

\* 'master' of github.com:alan-turing-institute/rsd-engineeringcourse:  
fix to requirements  
pulled ch05 from UCL  
updated schedule

commit da23a224bc7c0818c0a17e588d2a20122a522255  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-11-24

fix to requirements

commit 421ff20a51ce664b2b15a16c580ee95f23d1d42d  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-11-24

pulled ch05 from UCL

commit 91f1f59f0473106b8e798eefec9d3cbf2ab30adb  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-11-16

updated schedule

commit fd7cdbae8f1c5830b5b5cf031b47fe9252e1bf33  
Merge: 6853baf 0abdab7  
Author: James Hetherington <jamespjh@gmail.com>  
Date: 2018-11-15

Merge branch 'master' of github.com:alan-turing-institute/rsd-engineeringcourse

\* 'master' of github.com:alan-turing-institute/rsd-engineeringcourse:  
requirements  
checked ch04  
Notes from local  
fix in stashing code to avoid keys getting public  
update to schedule  
removed ref to UCL class  
auth, messy auth  
auth, messy auth  
fixing auth still

commit 0abdab7ba3d777a2e2082532014b82cdfe07141b  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-11-07

requirements

commit ad5ea5d8a6f8553972d6a02ae98e84718b61e933  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-11-07

checked ch04

commit c6bd2ae0cf66ea782ff835bc7db921fb5199aeb3  
Author: Giovanni Colavizza <gcolavizza@turing.ac.uk>  
Date: 2018-11-05

Notes from local

commit eafebb65296652991871756b01d965a947fe00b6  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-11-05

fix in stashing code to avoid keys getting public

commit fd369891f390a0a42036e0a1c0fa48b3724a0626  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-11-05

update to schedule

commit 6853bafb567cf2eefc3458bea29110b82481e86  
Author: James Hetherington <jamespjh@gmail.com>  
Date: 2018-11-02

Fix classes bug thanks to @hhau

commit 9beb9c72db7dc60567ad0b83a56e05ef4314d533  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-11-01

removed ref to UCL class

commit 120555ed2de539bbea9d315e1b35093bd1006223  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-26

auth, messy auth

commit 0a40b95d055e5db5807c8201ae29616c936440e4  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-26

auth, messy auth

commit c69208629761cf5f0616ec412bb50dbcda44e9f9  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-26

fixing auth still

commit ccf8dd8e069456963e1d7af096d7807bec466cc6  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-26

git auth hopefully fixed

commit ccad62f76bed097377470055ccf460ab36e113ea  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-25

auth

commit a61e0b9145e85f585a6ae762a7481d7f862f7175  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-25

auth

commit 3118652f5a8250e2a5a17cd23c8bfecaad4568fe  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-25

auth

commit 345f358ac5c94f53eccef5503c0d944a34159609  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-25

auth

commit 6b8044dbf1de4cdbd1aeb7aafd639fdd50d0affd  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-25

auth

commit 1e33f2c39ab83e2df1526111580cff49ef295ece  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-25

auth

commit 5942caaa71d4939089dd4330529685584c7fd00  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-25

git auth

commit eb32fc34a3b25affc5da910d9fe46a25a98bb35b  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-25

git auth

commit 36d381b87444a615acaclb9a153ebf5692d1c4cd  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-25

git auth

commit 27ae7bec7d1933e1028b147b10808a28e6f0d65d

Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-25

fix ch02

commit 26e80a0e2758e91f35024f660c9ef83f4d572027  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-25

fix 02

commit 49d97ff9475ac282bef6130f4e7f0e7346cc80c5  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-25

updated and checked weeks 2 to 4

commit eac3f2f5fa0540be9925fcbb3cle822b0f554aa7  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-10

updated schedule

commit fc73f6185088bc5a091020ab0b019a8a8c5a87e2  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-04

lesson 1 checked

commit d39e881a309c291a1320a22bdac207330b3e9655  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-04

minor tweaks to Install

commit 7c8e5544781fc7553e22a940c9779cefc853882b  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-03

delete previous folders

commit dbc617a22b4f7fdec78ad7b6158bef2701445a93  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-03

merging last 2 classes

commit 77a9e9e1d5b1e43c1c804635e8c9d755ef4957d  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-03

new chapters

commit d6e35c79dff0d0bfb42e37e9dcla71c460752073  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-03

added last two classes and updated index

commit adae052b2f9437ba8101d8a36d530ed71bda9ff3  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-03

added assessment

commit 74f022433ae336b1cfb966425dfed84194aa06a9  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-02

jekyll version making troubles

commit d3209e82acedfaaca0ed6d17edad482e354d6c5e  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-02

jekyll version making troubles

commit 2208e4b63bacbcc8c3443fe0285c7725b7dd0167  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-02

added imageio to req

commit 48c441ad21f08331e1df9fee845c1911b3518743  
Author: Giovanni1085 <gcolavizza@turing.ac.uk>  
Date: 2018-10-02

```
dropping UCL crumbs

commit cbf54789adc335b72877ca99035262357d75fb09
Author: James Hetherington <jamespjh@gmail.com>
Date:   2018-10-02

    adding missing packages to req, moving to 3.6

commit 9c2f1156b006c84409bd1ae092e6984c75aeeeda
Author: James Hetherington <jamespjh@gmail.com>
Date:   2018-10-02

    update jekyll and liquid

commit 76c3d827820229197eaec448547d3c58f1d1336c
Author: James Hetherington <gcolavizza@turing.ac.uk>
Date:   2018-10-01

    add comment

commit 5314bb92c80045dcb81f876fc6d353c55ac10db6
Author: James Hetherington <gcolavizza@turing.ac.uk>
Date:   2018-10-01

    typo

commit 0086e45732d04de9677e0005a832f85455881bef
Author: James Hetherington <jamespjh@gmail.com>
Date:   2018-10-01

    first commit
```

## Antipatch

Notice how the mistake has stayed in the history.

There is a new commit which undoes the change: this is colloquially called an “antipatch”. This is nice: you have a record of the full story, including the mistake and its correction.

## Rewriting history

It is possible, in git, to remove the most recent change altogether, “rewriting history”. Let’s make another bad change, and see how to do this.

## A new lie

```
%%writefile test.md
Mountains and Hills in the UK
=====
Engerland is not very mountainous.
But has some tall hills, and maybe a
mountain or two depending on your definition.
```

```
Overwriting test.md
```

```
%%bash
cat test.md
```

```
Mountains and Hills in the UK
=====
Engerland is not very mountainous.
But has some tall hills, and maybe a
mountain or two depending on your definition.
```

```
%%bash
git diff
```

```
%%bash
git commit -am "Add a silly spelling"
```

```
On branch 47-reorganise-chapters
Your branch is ahead of 'origin/47-reorganise-chapters' by 1 commit.
  (use "git push" to publish your local commits)
```

```
nothing to commit, working tree clean
```

```
-----  
CalledProcessError                                     Traceback (most recent call last)  
/tmp/ipykernel_3335/3130984486.py in <module>  
----> 1 get_ipython().run_cell_magic('bash', '', 'git commit -am "Add a silly  
spelling"\n')  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/interactiveshell.py in run_cell_magic(self, magic_name, line,  
cell)  
 2404         with self.builtin_trap:  
 2405             args = (magic_arg_s, cell)  
-> 2406             result = fn(*args, **kwargs)  
 2407             return result  
 2408  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magics/script.py in named_script_magic(line, cell)  
 140         else:  
 141             line = script  
-> 142             return self.shebang(line, cell)  
 143  
 144         # write a basic docstring:  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/decorator.py in  
fun(*args, **kw)  
 230         if not kwsyntax:  
 231             args, kw = fix(args, kw, sig)  
-> 232             return caller(func, *(extras + args), **kw)  
 233     fun.__name__ = func.__name__  
 234     fun.__doc__ = func.__doc__  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magic.py in <lambda>(f, *a, **k)  
 185     # but it's overkill for just that one bit of state.  
 186     def magic_deco(arg):  
-> 187         call = lambda f, *a, **k: f(*a, **k)  
 188  
 189         if callable(arg):  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magics/script.py in shebang(self, line, cell)  
 243             sys.stderr.flush()  
 244             if args.raise_error and p.returncode!=0:  
-> 245                 raise CalledProcessError(p.returncode, cell, output=out,  
stderr=err)  
 246  
 247     def _run_script(self, p, cell, to_close):  
  
CalledProcessError: Command 'b'git commit -am "Add a silly spelling"\n'' returned non-  
zero exit status 1.
```

```
%%bash  
git log --date=short
```

```
commit 13cfc78608055a7e681195e13e47bea96934e648
Author: Turing Developer <developer@example.com>
Date:   2021-11-08

    Add a silly spelling

commit c489aababc4488d5556ef340257905ee18873125
Author: Turing Developer <developer@example.com>
Date:   2021-11-08

    Revert "Add a lie about a mountain"

    This reverts commit 6e8a302387007780675dbd5cb1823901d1a7b59b.

commit f79cacc17500651a228f9b5a1922c3b50ea723c3
Author: Turing Developer <developer@example.com>
Date:   2021-11-08

    Change title

commit 6e8a302387007780675dbd5cb1823901d1a7b59b
Author: Turing Developer <developer@example.com>
Date:   2021-11-08

    Add a lie about a mountain

commit 238eaff15e2769e0ef1d989f1a2e8be1873fa0ab
Author: Turing Developer <developer@example.com>
Date:   2021-11-08

    First commit of discourse on UK topography
```

## Using reset to rewrite history

```
%%bash
git reset HEAD^
```

```
Unstaged changes after reset:
M      test.md
```

```
%%bash
git log --date=short
```

```
commit c489aababc4488d5556ef340257905ee18873125
Author: Turing Developer <developer@example.com>
Date:   2021-11-08

    Revert "Add a lie about a mountain"

    This reverts commit 6e8a302387007780675dbd5cb1823901d1a7b59b.

commit f79cacc17500651a228f9b5a1922c3b50ea723c3
Author: Turing Developer <developer@example.com>
Date:   2021-11-08

    Change title

commit 6e8a302387007780675dbd5cb1823901d1a7b59b
Author: Turing Developer <developer@example.com>
Date:   2021-11-08

    Add a lie about a mountain

commit 238eaff15e2769e0ef1d989f1a2e8be1873fa0ab
Author: Turing Developer <developer@example.com>
Date:   2021-11-08

    First commit of discourse on UK topography
```

## Covering your tracks

The silly spelling is *no longer in the log*. This approach to fixing mistakes, “rewriting history” with `reset`, instead of adding an antipatch with `revert`, is dangerous, and we don’t recommend it. But you may want to do it for small silly mistakes, such as to correct a commit message.

## Resetting the working area

When `git reset` removes commits, it leaves your working directory unchanged – so you can keep the work in the bad change if you want.

```
%%bash  
cat test.md
```

```
Mountains and Hills in the UK  
=====  
Engerland is not very mountainous.  
But has some tall hills, and maybe a  
mountain or two depending on your definition.
```

If you want to lose the change from the working directory as well, you can do `git reset --hard`.

I'm going to get rid of the silly spelling, and I didn't do `--hard`, so I'll reset the file from the working directory to be the same as in the index:

```
%%bash  
git checkout test.md
```

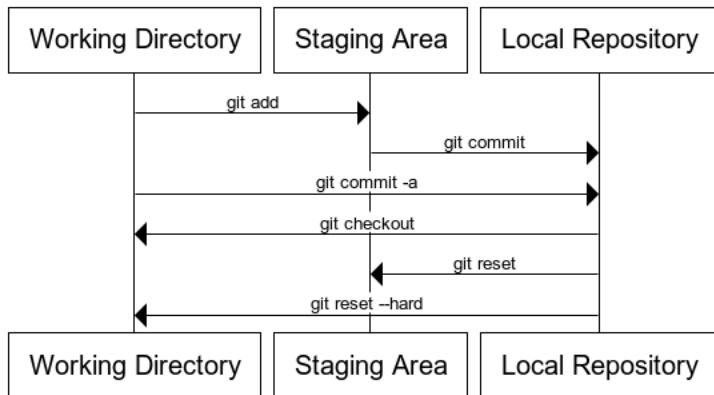
```
Updated 1 path from the index
```

```
%%bash  
cat test.md
```

```
Mountains and Hills in the UK  
=====  
England is not very mountainous.  
But has some tall hills, and maybe a mountain or two depending on your definition.
```

We can add this to our diagram:

```
message = """  
Working Directory -> Staging Area : git add  
Staging Area -> Local Repository : git commit  
Working Directory -> Local Repository : git commit -a  
Local Repository -> Working Directory : git checkout  
Local Repository -> Staging Area : git reset  
Local Repository -> Working Directory: git reset --hard  
"""  
  
from wsd import wsd  
  
%matplotlib inline  
wsd(message)
```



[www.websequencediagrams.com](http://www.websequencediagrams.com)

We can add it to Jim's story:

```

message = """
participant "Jim's repo" as R
participant "Jim's index" as I
participant Jim as J

note right of J: git revert HEAD^

J->R: Add new commit reversing change
R->I: update staging area to reverted version
I->J: update file to reverted version

note right of J: vim test.md
note right of J: git commit -am "Add another mistake"
J->I: Add mistake
I->R: Add mistake

note right of J: git reset HEAD^

J->R: Delete mistaken commit
R->I: Update staging area to reset commit

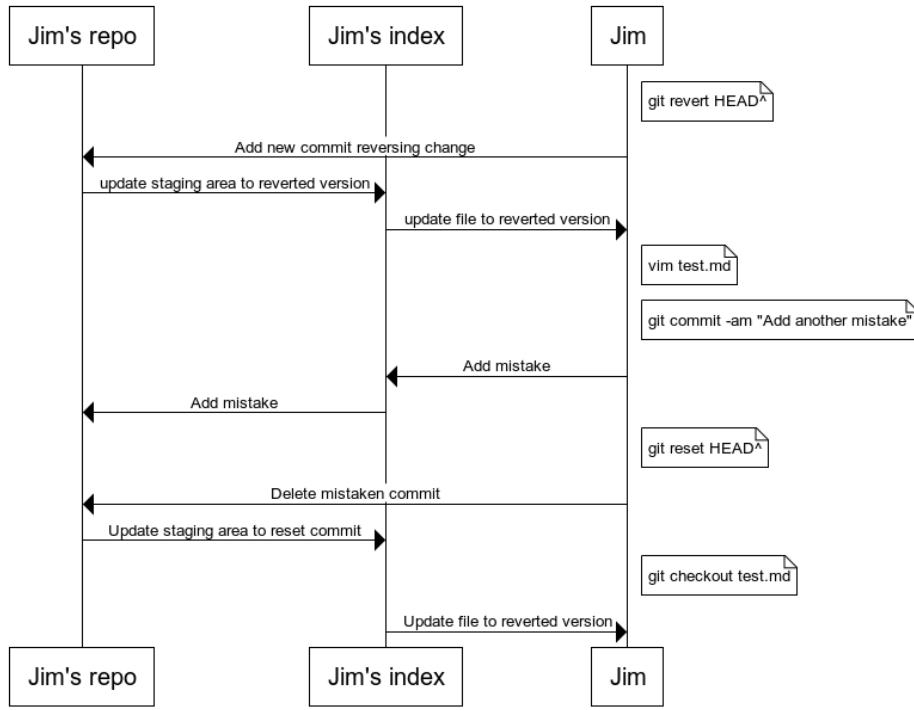
note right of J: git checkout test.md

I->J: Update file to reverted version

"""

wsd(message)

```



[www.websequencediagrams.com](http://www.websequencediagrams.com)

## Publishing

We're still in our working directory:

```

import os

top_dir = os.getcwd()
git_dir = os.path.join(top_dir, "learning_git")
working_dir = os.path.join(git_dir, "git_example")
os.chdir(working_dir)
working_dir

```

```
'/home/runner/work/rsd-engineeringcourse/rsd-
engineeringcourse/module04_version_control_with_git/learning_git/git_example'
```

## Sharing your work

So far, all our work has been on our own computer. But a big part of the point of version control is keeping your work safe, on remote servers. Another part is making it easy to share your work with the world. In this example, we'll be using the "GitHub" cloud repository to store and publish our work.

If you have not done so already, you should create an account on GitHub: go to <https://github.com/>, fill in a username and password, and click on "sign up for free".

## Creating a repository

Ok, let's create a repository to store our work. Hit "new repository" on the right of the github home screen, or click [here](#).

Fill in a short name, and a description. Choose a "public" repository. Don't choose to add a Readme.

## Paying for GitHub

For this course, you should use public repositories in your personal account for your example work: it's good to share! GitHub is free for open source, but in general, charges a fee if you want to keep your work private.

In the future, you might want to keep your work on GitHub private.

Students can get free private repositories on GitHub, by going to [GitHub Education](#) and filling in a form (look for the Student Developer Pack).

## Adding a new remote to your repository

Instructions will appear, once you've created the repository, as to how to add this new "remote" server to your repository. In this example we are using pre-authorised [Deploy Keys](#) to connect using the [SSH](#) method. If you prefer to use username and password/token, these instructions will be slightly different:

```
%%bash
git remote add origin git@github.com:alan-turing-institute/github-example.git
```

```
error: remote origin already exists.
```

```
-----  
CalledProcessError                                     Traceback (most recent call last)  
/tmp/ipykernel_3366/2402638072.py in <module>  
---> 1 get_ipython().run_cell_magic('bash', '', 'git remote add origin  
git@github.com:alan-turing-institute/github-example.git\n')  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/interactiveshell.py in run_cell_magic(self, magic_name, line,  
cell)  
    2404         with self.builtin_trap:  
    2405             args = (magic_arg_s, cell)  
-> 2406             result = fn(*args, **kwargs)  
    2407             return result  
    2408  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magics/script.py in named_script_magic(line, cell)  
    140         else:  
    141             line = script  
-> 142             return self.shebang(line, cell)  
    143  
    144     # write a basic docstring:  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/decorator.py in  
fun(*args, **kw)  
    230         if not kwsyntax:  
    231             args, kw = fix(args, kw, sig)  
-> 232             return caller(func, *(extras + args), **kw)  
    233         fun.__name__ = func.__name__  
    234         fun.__doc__ = func.__doc__  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magic.py in <lambda>(f, *a, **k)  
    185     # but it's overkill for just that one bit of state.  
    186     def magic_deco(arg):  
-> 187         call = lambda f, *a, **k: f(*a, **k)  
    188  
    189         if callable(arg):  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magics/script.py in shebang(self, line, cell)  
    243         sys.stderr.flush()  
    244         if args.raise_error and p.returncode!=0:  
-> 245             raise CalledProcessError(p.returncode, cell, output=out,  
stderr=err)  
    246  
    247     def _run_script(self, p, cell, to_close):  
  
CalledProcessError: Command 'b'git remote add origin git@github.com:alan-turing-  
institute/github-example.git\n' returned non-zero exit status 3.
```

Note that the `https` version of this instruction would be something like `git remote add origin`

`https://${YOUR_USERNAME}:${GITHUB_TOKEN}@github.com/alan-turing-institute/github-example.git`

```
%%bash  
git remote -v
```

```
origin  git@github.com:alan-turing-institute/github-example.git (fetch)  
origin  git@github.com:alan-turing-institute/github-example.git (push)
```

```
%%bash  
git push -uf origin main # Note we use the '-f' flag here to force an update
```

```
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

```
To github.com:alan-turing-institute/github-example.git  
+ d358baa...c489aab main -> main (forced update)
```

## Remotes

The first command sets up the server as a new `remote`, called `origin`.

Git, unlike some earlier version control systems is a “distributed” version control system, which means you can work with multiple remote servers.

Usually, commands that work with remotes allow you to specify the remote to use, but assume the `origin` remote if you don’t.

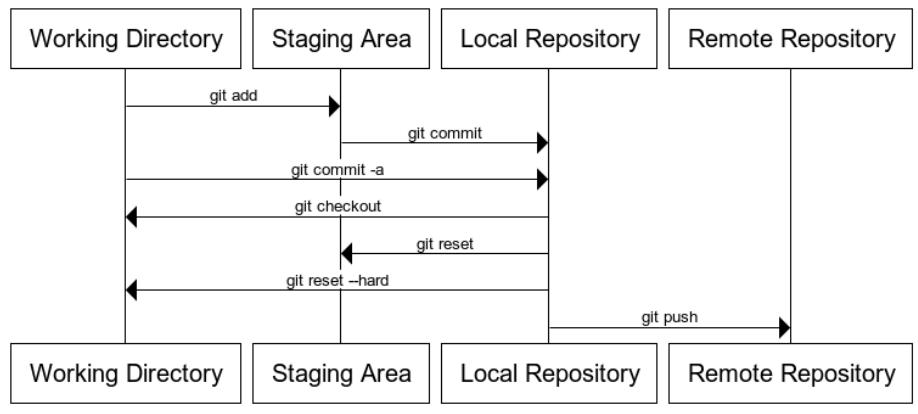
Here, `git push` will push your whole history onto the server, and now you’ll be able to see it on the internet! Refresh your web browser where the instructions were, and you’ll see your repository!

Let’s add these commands to our diagram:

```
message = """
Working Directory -> Staging Area : git add
Staging Area -> Local Repository : git commit
Working Directory -> Local Repository : git commit -a
Local Repository -> Working Directory : git checkout
Local Repository -> Staging Area : git reset
Local Repository -> Working Directory: git reset --hard
Local Repository -> Remote Repository : git push
"""

from wsd import wsd

%matplotlib inline
wsd(message)
```



[www.websequencediagrams.com](http://www.websequencediagrams.com)

## Playing with GitHub

Take a few moments to click around and work your way through the GitHub interface. Try clicking on ‘[test.md](#)’ to see the content of the file: notice how the markdown renders prettily.

Click on “commits” near the top of the screen, to see all the changes you’ve made. Click on the commit number next to the right of a change, to see what changes it includes: removals are shown in red, and additions in green.

## Working with multiple files

### Some new content

So far, we’ve only worked with one file. Let’s add another:

```
vim lakeland.md
```

```
%%writefile lakeland.md
Lakeland
=====

Cumbria has some pretty hills, and lakes too.
```

```
Writing lakeland.md
```

```
cat lakeland.md
```

```
Lakeland
=====
Cumbria has some pretty hills, and lakes too.
```

## Git will not by default commit your new file

```
%%bash
git commit -am "Try to add Lakeland" || echo "Commit failed"
```

```
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    __pycache__/
      lakeland.md
    wsd.py

nothing added to commit but untracked files present (use "git add" to track)
Commit failed
```

This failed, because we've not told git to track the new file yet.

## Tell git about the new file

```
%%bash
git add lakeland.md
git commit -am "Add lakeland"
```

```
[main d063119] Add lakeland
 1 file changed, 4 insertions(+)
 create mode 100644 lakeland.md
```

Ok, now we have added the change about Cumbria to the file. Let's publish it to the origin repository.

```
%%bash
git push
```

```
To github.com:alan-turing-institute/github-example.git
 c489aab..d063119  main -> main
```

Visit GitHub, and notice this change is on your repository on the server. We could have said `git push origin` to specify the remote to use, but origin is the default.

## Changing two files at once

What if we change both files?

```
%%writefile lakeland.md
Lakeland
=====

Cumbria has some pretty hills, and lakes too

Mountains:
* Helvellyn
```

```
Overwriting lakeland.md
```

```
%%writefile test.md
Mountains and Lakes in the UK
=====
Engerland is not very mountainous.
But has some tall hills, and maybe a
mountain or two depending on your definition.
```

Overwriting test.md

```
%%bash
git status
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   lakeland.md
      modified:   test.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    __pycache__/
    wsd.py

no changes added to commit (use "git add" and/or "git commit -a")
```

These changes should really be separate commits. We can do this with careful use of git add, to **stage** first one commit, then the other.

```
%%bash
git add test.md
git commit -m "Include lakes in the scope"
```

```
[main 6ff088f] Include lakes in the scope
 1 file changed, 4 insertions(+), 3 deletions(-)
```

Because we “staged” only [test.md](#), the changes to [lakeland.md](#) were not included in that commit.

```
%%bash
git commit -am "Add Helvellyn"
```

```
[main 82b4fa0] Add Helvellyn
 1 file changed, 4 insertions(+), 1 deletion(-)
```

```
%%bash
git log --oneline
```

```
82b4fa0 Add Helvellyn
6ff088f Include lakes in the scope
d063119 Add lakeland
c489aab Revert "Add a lie about a mountain"
f79cacc Change title
6e8a302 Add a lie about a mountain
238eaff First commit of discourse on UK topography
```

```
%%bash
git push
```

```
To github.com:alan-turing-institute/github-example.git
d063119..82b4fa0  main -> main
```

```

message = """
participant "Jim's remote" as M
participant "Jim's repo" as R
participant "Jim's index" as I
participant Jim as J

note right of J: vim test.md
note right of J: vim lakeland.md

note right of J: git add test.md
J->I: Add *only* the changes to test.md to the staging area

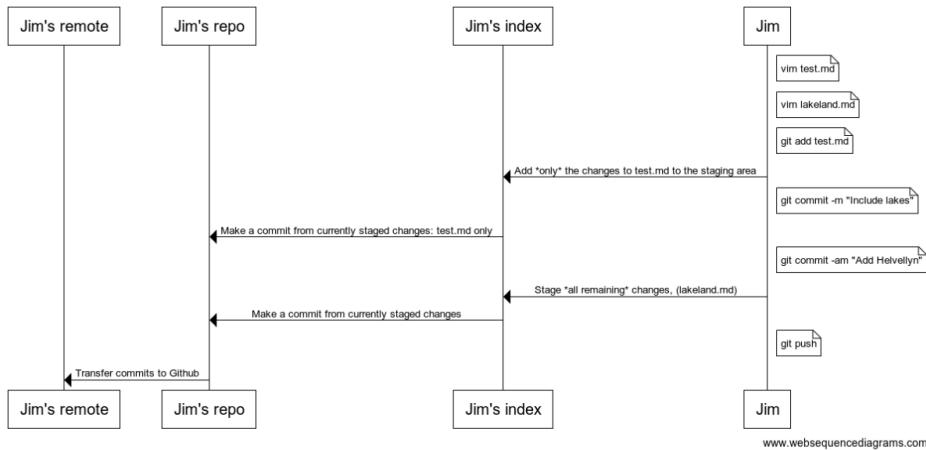
note right of J: git commit -m "Include lakes"
I->R: Make a commit from currently staged changes: test.md only

note right of J: git commit -am "Add Helvellyn"
J->I: Stage *all remaining* changes, (lakeland.md)
I->R: Make a commit from currently staged changes

note right of J: git push
R->M: Transfer commits to Github
"""

wsd(message)

```



## Collaboration

### Form a team

Now we're going to get to the most important question of all with Git and GitHub: working with others.

Organise into pairs. You're going to be working on the website of one of the two of you, together, so decide who is going to be the leader, and who the collaborator.

### Giving permission

The leader needs to let the collaborator have the right to make changes to his code.

In GitHub, go to [Settings](#) on the right, then [Collaborators & teams](#) on the left.

Add the user name of your collaborator to the box. They now have the right to push to your repository.

### Obtaining a colleague's code

Next, the collaborator needs to get a copy of the leader's code. For this example notebook, I'm going to be collaborating with myself, swapping between my two repositories. Make yourself a space to put it your work. (I will have two)

```
import os

top_dir = os.getcwd()
git_dir = os.path.join(top_dir, "learning_git")
working_dir = os.path.join(git_dir, "git_example")
os.chdir(git_dir)
```

```
%%bash
pwd
rm -rf github-example # cleanup after previous example
rm -rf partner_dir # cleanup after previous example
```

```
/home/runner/work/rsd-engineeringcourse/rsd-
engineeringcourse/module04_version_control_with_git/learning_git
```

Next, the collaborator needs to find out the URL of the repository: they should go to the leader's repository's GitHub page, and note the URL on the top of the screen.

As before, we're using **SSH** to connect - to do this you'll need to make sure the **ssh** button is pushed, and check that the URL begins with <git@github.com>.

Copy the URL into your clipboard by clicking on the icon to the right of the URL, and then:

```
%%bash
pwd
git clone git@github.com:alan-turing-institute/github-example.git partner_dir
```

```
/home/runner/work/rsd-engineeringcourse/rsd-
engineeringcourse/module04_version_control_with_git/learning_git
```

```
Cloning into 'partner_dir'...
Warning: Permanently added the RSA host key for IP address '140.82.113.3' to the list
of known hosts.
```

```
partner_dir = os.path.join(git_dir, "partner_dir")
os.chdir(partner_dir)
```

```
%%bash
pwd
ls
```

```
/home/runner/work/rsd-engineeringcourse/rsd-
engineeringcourse/module04_version_control_with_git/learning_git/partner_dir
Scotland.md
lakeland.md
test.md
```

Note that your partner's files are now present on your disk:

```
%%bash
cat lakeland.md
```

```
Lakeland
=====
Cumbria has some pretty hills, and lakes too
Mountains:
* Helvellyn
```

## Nonconflicting changes

Now, both of you should make some changes. To start with, make changes to *different* files. This will mean your work doesn't "conflict". Later, we'll see how to deal with changes to a shared file.

Both of you should commit, but not push, your changes to your respective files:

E.g., the leader:

```
os.chdir(working_dir)
```

```
%%writefile Wales.md
Mountains In Wales
=====
```

```
* Tryfan
* Yr Wyddfa
```

```
Writing Wales.md
```

```
%%bash
ls
```

```
Wales.md
test.md
```

```
%%bash
git add Wales.md
git commit -m "Add wales"
```

```
On branch 47-reorganise-chapters
Your branch is ahead of 'origin/47-reorganise-chapters' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

```
The following paths are ignored by one of your .gitignore files:
module04_version_control_with_git/learning_git
hint: Use -f if you really want to add them.
hint: Turn this message off by running
hint: "git config advice.addIgnoredFile false"
```

```
-----  

CalledProcessError                                     Traceback (most recent call last)
/tmp/ipykernel_3387/3825437270.py in <module>
--> 1 get_ipython().run_cell_magic('bash', '', 'git add Wales.md\ngit commit -m "Add wales"\n')

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/interactiveshell.py in run_cell_magic(self, magic_name, line,
cell)
2404         with self.builtin_trap:
2405             args = (magic_arg_s, cell)
-> 2406             result = fn(*args, **kwargs)
2407             return result
2408

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/magics/script.py in named_script_magic(line, cell)
140     else:
141         line = script
--> 142     return self.shebang(line, cell)
143
144     # write a basic docstring:
```

**/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/decorator.py in fun(\*args, \*\*kw)**

```
230         if not kwsyntax:
231             args, kw = fix(args, kw, sig)
--> 232         return caller(func, *(extras + args), **kw)
233     fun.__name__ = func.__name__
234     fun.__doc__ = func.__doc__
```

**/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/magic.py in <lambda>(f, \*a, \*\*k)**

```
185     # but it's overkill for just that one bit of state.
186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
188
189         if callable(arg):
```

**/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/magics/script.py in shebang(self, line, cell)**

```
243         sys.stderr.flush()
244         if args.raise_error and p.returncode!=0:
--> 245             raise CalledProcessError(p.returncode, cell, output=out,
stderr=err)
246
247     def _run_script(self, p, cell, to_close):
```

**CalledProcessError: Command 'b'git add Wales.md\ngit commit -m "Add wales"\n'' returned non-zero exit status 1.**

And the partner:

```
os.chdir(partner_dir)
```

```
%%writefile Scotland.md
Mountains In Scotland
=====
* Ben Eighe
* Cairngorm
```

```
Writing Scotland.md
```

```
%%bash
ls
```

```
Scotland.md  
lakeland.md  
test.md
```

```
%%bash  
git add Scotland.md  
git commit -m "Add Scotland"
```

```
[main 289afed] Add Scotland  
1 file changed, 5 insertions(+)  
create mode 100644 Scotland.md
```

One of you should now push with `git push`:

```
%%bash  
git push
```

```
To github.com:alan-turing-institute/github-example.git  
82b4fa0..289afed main -> main
```

## Rejected push

The other should then attempt to push, but should receive an error message:

```
os.chdir(working_dir)
```

```
%%bash  
git push || echo "Push failed"
```

```
Push failed
```

```
To github.com:alan-turing-institute/github-example.git  
! [rejected]      main -> main (fetch first)  
error: failed to push some refs to 'github.com:alan-turing-institute/github-  
example.git'  
hint: Updates were rejected because the remote contains work that you do  
hint: not have locally. This is usually caused by another repository pushing  
hint: to the same ref. You may want to first integrate the remote changes  
hint: (e.g., 'git pull ...') before pushing again.  
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Do as it suggests:

```
%%bash  
git pull
```

```
Merge made by the 'recursive' strategy.  
Scotland.md | 5 +++++  
1 file changed, 5 insertions(+)  
create mode 100644 Scotland.md
```

```
From github.com:alan-turing-institute/github-example  
82b4fa0..289afed main      -> origin/main  
* [new branch]    experiment -> origin/experiment
```

## Merge commits

A window may pop up with a suggested default commit message. This commit is special: it is a *merge* commit. It is a commit which combines your collaborator's work with your own.

Now, push again with `git push`. This time it works. If you look on GitHub, you'll now see that it contains both sets of changes.

```
%%bash  
git push
```

```
To github.com:alan-turing-institute/github-example.git  
 289afed..34f8bbb main -> main
```

The partner now needs to pull down that commit:

```
os.chdir(partner_dir)
```

```
%%bash  
git pull
```

```
Updating 289afed..34f8bbb  
Fast-forward  
 Wales.md | 5 +++++  
 1 file changed, 5 insertions(+)  
 create mode 100644 Wales.md
```

```
From github.com:alan-turing-institute/github-example  
 289afed..34f8bbb main      -> origin/main
```

```
%%bash  
ls
```

```
Scotland.md  
Wales.md  
LakeLand.md  
test.md
```

## Nonconflicted commits to the same file

Go through the whole process again, but this time, both of you should make changes to a single file, but make sure that you don't touch the same *line*. Again, the merge should work as before:

```
%%writefile Wales.md  
Mountains In Wales  
=====  
* Tryfan  
* Snowdon
```

```
Overwriting Wales.md
```

```
%%bash  
git diff
```

```
diff --git a/Wales.md b/Wales.md  
index f3e88b4..90f23ec 100644  
--- a/Wales.md  
+++ b/Wales.md  
@@ -2,4 +2,4 @@ Mountains In Wales  
=====  
 * Tryfan  
-* Yr Wyddfa  
+* Snowdon
```

```
%%bash  
git commit -am "Translating from the Welsh"
```

```
[main b8fb6c5] Translating from the Welsh  
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
%%bash  
git log --oneline
```

```
b8fb6c5 Translating from the Welsh  
34f8bbb Merge branch 'main' of github.com:alan-turing-institute/github-example  
289afed Add Scotland  
0c976b0 Add wales  
82b4fa0 Add Helvellyn  
6ff088f Include lakes in the scope  
d063119 Add lakeland  
c489aab Revert "Add a lie about a mountain"  
f79cacc Change title  
6e8a302 Add a lie about a mountain  
238eaff First commit of discourse on UK topography
```

```
os.chdir(working_dir)
```

```
%%writefile Wales.md  
Mountains In Wales  
=====  
* Pen y Fan  
* Tryfan  
* Yr Wyddfa
```

```
Overwriting Wales.md
```

```
%%bash  
git commit -am "Add a beacon"
```

```
[main 8cd684e] Add a beacon  
1 file changed, 1 insertion(+)
```

```
%%bash  
git log --oneline
```

```
8cd684e Add a beacon  
34f8bbb Merge branch 'main' of github.com:alan-turing-institute/github-example  
289afed Add Scotland  
0c976b0 Add wales  
82b4fa0 Add Helvellyn  
6ff088f Include lakes in the scope  
d063119 Add lakeland  
c489aab Revert "Add a lie about a mountain"  
f79cacc Change title  
6e8a302 Add a lie about a mountain  
238eaff First commit of discourse on UK topography
```

```
%%bash  
git push
```

```
To github.com:alan-turing-institute/github-example.git  
34f8bbb..8cd684e main -> main
```

Switching back to the other partner...

```
os.chdir(partner_dir)
```

```
%%bash  
git push || echo "Push failed"
```

```
Push failed
```

```
To github.com:alan-turing-institute/github-example.git
 ! [rejected]      main -> main (fetch first)
error: failed to push some refs to 'github.com:alan-turing-institute/github-
example.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
%%bash
git pull
```

```
Auto-merging Wales.md
Merge made by the 'recursive' strategy.
Wales.md | 1 +
1 file changed, 1 insertion(+)
```

```
From github.com:alan-turing-institute/github-example
34f8bbb..8cd684e  main      -> origin/main
```

```
%%bash
git push
```

```
To github.com:alan-turing-institute/github-example.git
 8cd684e..ecc3206  main -> main
```

```
%%bash
git log --oneline --graph
```

```
*   ecc3206 Merge branch 'main' of github.com:alan-turing-institute/github-example
|\ 
| * 8cd684e Add a beacon
* | b8fb6c5 Translating from the Welsh
|/
* 34f8bbb Merge branch 'main' of github.com:alan-turing-institute/github-example
|\ 
| * 289afed Add Scotland
* | 0c976b0 Add wales
|/
* 82b4fa0 Add Helvellyn
* 6ff088f Include lakes in the scope
* d063119 Add lakeland
* c489aab Revert "Add a lie about a mountain"
* f79cacc Change title
* 6e8a302 Add a lie about a mountain
* 238eaff First commit of discourse on UK topography
```

```
os.chdir(working_dir)
```

```
%%bash
git pull
```

```
Updating 8cd684e..ecc3206
Fast-forward
Wales.md | 2 ++
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
From github.com:alan-turing-institute/github-example
8cd684e..ecc3206  main      -> origin/main
```

```
%%bash
git log --graph --oneline
```

```

*   ecc3206 Merge branch 'main' of github.com:alan-turing-institute/github-example
|\
| * 8cd684e Add a beacon
* | b8fb6c5 Translating from the Welsh
|/
* 34f8bbb Merge branch 'main' of github.com:alan-turing-institute/github-example
|\
| * 289afed Add Scotland
* | 0c976b0 Add wales
|/
* 82b4fa0 Add Helvellyn
* 6ff088f Include lakes in the scope
* d063119 Add lakeland
* c489aab Revert "Add a lie about a mountain"
* f79cacc Change title
* 6e8a302 Add a lie about a mountain
* 238eaff First commit of discourse on UK topography

```

```

message = """
participant Sue as S
participant "Sue's repo" as SR
participant "Shared remote" as M
participant "Jim's repo" as JR
participant Jim as J

note left of S: git clone
M->SR: fetch commits
SR->S: working directory as at latest commit

note left of S: edit Scotland.md
note right of J: edit Wales.md

note left of S: git commit -am "Add scotland"
S->SR: create commit with Scotland file

note right of J: git commit -am "Add wales"
J->JR: create commit with Wales file

note left of S: git push
SR->M: update remote with changes

note right of J: git push
JR-->M: !Rejected change

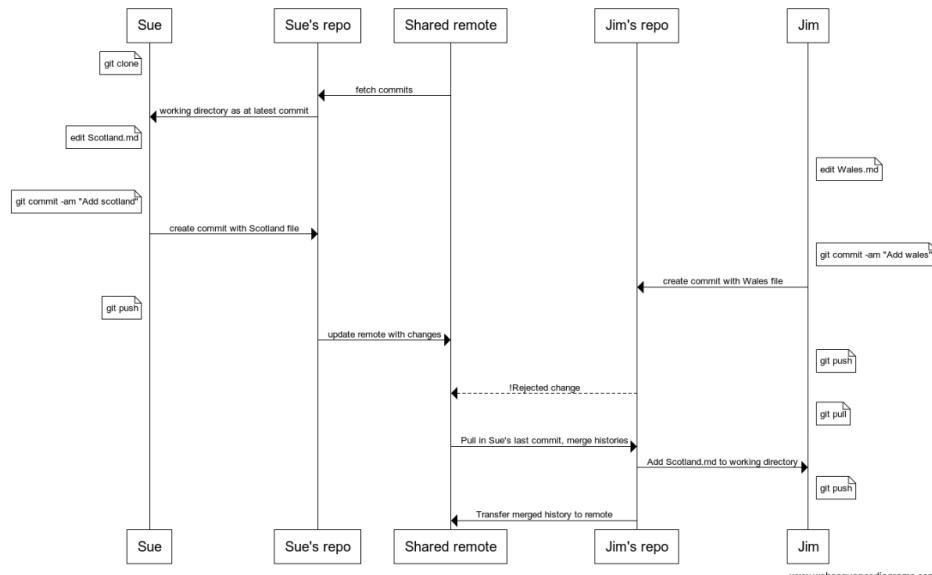
note right of J: git pull
M->JR: Pull in Sue's last commit, merge histories
JR->J: Add Scotland.md to working directory

note right of J: git push
JR->M: Transfer merged history to remote

"""
from wsdiagram import wsdiagram

%matplotlib inline
wsdiagram(message)

```



## Conflicting commits

Finally, go through the process again, but this time, make changes which touch the same line.

```
%%writefile Wales.md
Mountains In Wales
=====
* Pen y Fan
* Tryfan
* Snowdon
* Fan y Big
```

Overwriting Wales.md

```
%%bash
git commit -am "Add another Beacon"
git push
```

```
[main ed30178] Add another Beacon
 1 file changed, 1 insertion(+)
```

```
To github.com:alan-turing-institute/github-example.git
  ecc3206..ed30178 main -> main
```

```
os.chdir(partner_dir)
```

```
%%writefile Wales.md
Mountains In Wales
=====
* Pen y Fan
* Tryfan
* Snowdon
* Glyder Fawr
```

Overwriting Wales.md

```
%%bash
git commit -am "Add Glyder"
```

```
[main bc04a83] Add Glyder
 1 file changed, 1 insertion(+)
```

```
%%bash
git push || echo "Push failed"
```

Push failed

```
To github.com:alan-turing-institute/github-example.git
 ! [rejected]      main -> main (fetch first)
error: failed to push some refs to 'github.com:alan-turing-institute/github-
example.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

When you pull, instead of offering an automatic merge commit message, it says:

```
%%bash
git pull || echo "Pull failed"
```

```
Auto-merging Wales.md
CONFLICT (content): Merge conflict in Wales.md
Automatic merge failed; fix conflicts and then commit the result.
Pull failed
```

```
From github.com:alan-turing-institute/github-example
ecc3206..ed30178  main      -> origin/main
```

## Resolving conflicts

Git couldn't work out how to merge the two different sets of changes.

You now need to manually resolve the conflict.

It has marked the conflicted area:

```
%%bash
cat Wales.md
```

```
Mountains In Wales
=====
* Pen y Fan
* Tryfan
* Snowdon
<<<<< HEAD
* Glyder Fawr
=====
* Fan y Big
>>>> ed301786b17deffffe617ed8c7ded6591a7fb94f0
```

Manually edit the file, to combine the changes as seems sensible and get rid of the symbols:

```
%%writefile Wales.md
Mountains In Wales
=====
* Pen y Fan
* Tryfan
* Snowdon
* Fan y Big
* Glyder Fawr
```

```
Overwriting Wales.md
```

## Commit the resolved file

Now commit the merged result:

```
%%bash
git commit -a --no-edit # I added a No-edit for this non-interactive session. You can
edit the commit if you like.
```

```
[main 85b2797] Merge branch 'main' of github.com:alan-turing-institute/github-example
```

```
%%bash
git push
```

```
To github.com:alan-turing-institute/github-example.git
  ed30178..85b2797  main -> main
```

```
os.chdir(working_dir)
```

```
%%bash
git pull
```

```
Updating ed30178..85b2797
```

```
Fast-forward
```

```
Wales.md | 1 +  
1 file changed, 1 insertion(+)
```

```
From github.com:alan-turing-institute/github-example  
ed30178..85b2797 main -> origin/main
```

```
%%bash  
cat Wales.md
```

```
Mountains In Wales
```

```
=====
```

- \* Pen y Fan
- \* Tryfan
- \* Snowdon
- \* Fan y Big
- \* Glyder Fawr

```
%%bash  
git log --oneline --graph
```

```
* 85b2797 Merge branch 'main' of github.com:alan-turing-institute/github-example  
|\  
| * ed30178 Add another Beacon  
* | bc04a83 Add Glyder  
|/  
* ecc3206 Merge branch 'main' of github.com:alan-turing-institute/github-example  
|\  
| * 8cd684e Add a beacon  
* | b8fb6c5 Translating from the Welsh  
|/  
* 34f8bbb Merge branch 'main' of github.com:alan-turing-institute/github-example  
|\  
| * 289afed Add Scotland  
* | 0c976b0 Add wales  
|/  
* 82b4fa0 Add Helvellyn  
* 6ff088f Include lakes in the scope  
* d063119 Add lakeland  
* c489aab Revert "Add a lie about a mountain"  
* f79cacc Change title  
* 6e8a302 Add a lie about a mountain  
* 238eaff First commit of discourse on UK topography
```

## Distributed VCS in teams with conflicts

```

message = """
participant Sue as S
participant "Sue's repo" as SR
participant "Shared remote" as M
participant "Jim's repo" as JR
participant Jim as J

note left of S: edit the same line in wales.md
note right of J: edit the same line in wales.md

note left of S: git commit -am "update wales.md"
S->SR: add commit to local repo

note right of J: git commit -am "update wales.md"
J->JR: add commit to local repo

note left of S: git push
SR->M: transfer commit to remote

note right of J: git push
JR->M: !Rejected

note right of J: git pull
M->J: Make conflicted file with conflict markers

note right of J: edit file to resolve conflicts
note right of J: git add wales.md
note right of J: git commit
J->JR: Mark conflict as resolved

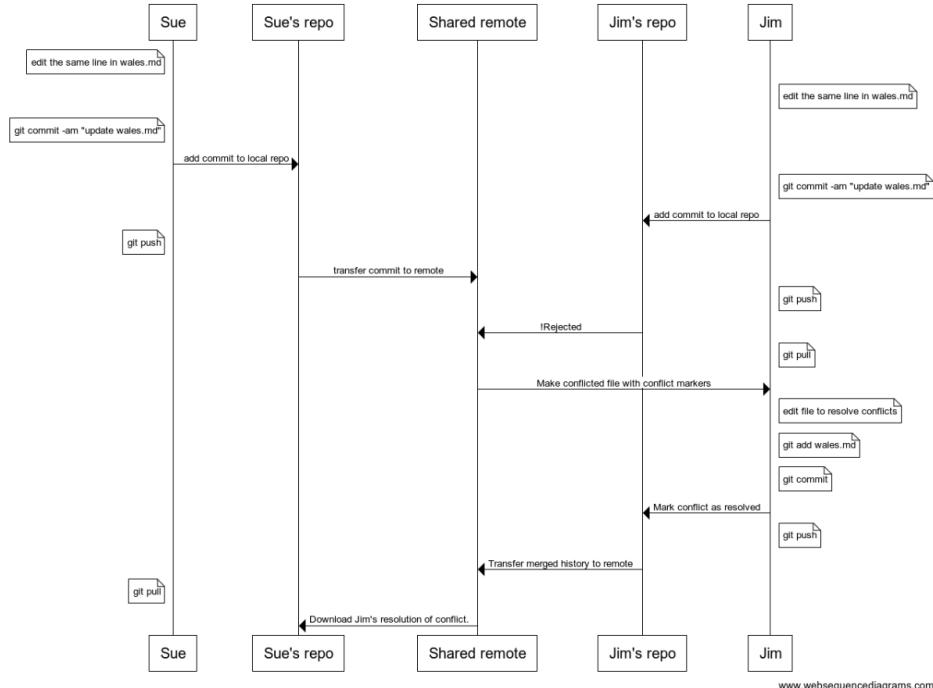
note right of J: git push
JR->M: Transfer merged history to remote

note left of S: git pull
M->SR: Download Jim's resolution of conflict.

"""

wsd(message)

```



[www.websequencediagrams.com](http://www.websequencediagrams.com)

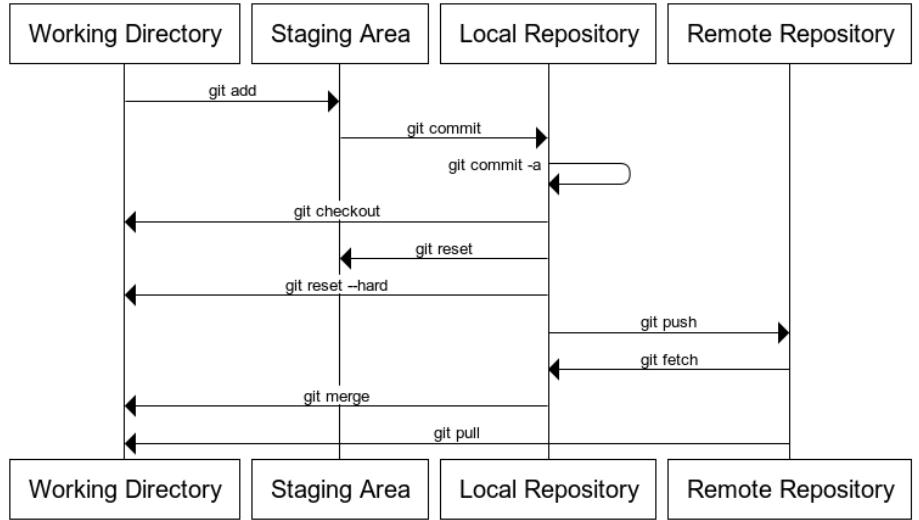
## The Levels of Git

```

message = """
Working Directory -> Staging Area : git add
Staging Area -> Local Repository : git commit
Local Repository -> Local Repository : git commit -a
Local Repository -> Working Directory : git checkout
Local Repository -> Staging Area : git reset
Local Repository -> Working Directory: git reset --hard
Local Repository -> Remote Repository : git push
Remote Repository -> Local Repository : git fetch
Local Repository -> Working Directory : git merge
Remote Repository -> Working Directory: git pull
"""

wsd(message)

```



www.websequencediagrams.com

## Editing directly on GitHub

Note that you can also make changes in the GitHub website itself. Visit one of your files, and hit "edit".

Make a change in the edit window, and add an appropriate commit message.

That change now appears on the website, but not in your local copy. (Verify this).

Now pull, and check the change is now present on your local version.

## GitHub as a social network

In addition to being a repository for code, and a way to publish code, GitHub is a social network.

You can follow the public work of other coders: go to the profile of your collaborator in your browser, and hit the "follow" button.

[Here's mine](#) : if you want to you can follow me.

Using GitHub to build up a good public profile of software projects you've worked on is great for your CV!

## Fork and Pull

### Different ways of collaborating

We have just seen how we can work with others on GitHub: we add them as collaborators on our repositories and give them permissions to push changes.

Let's talk now about some other type of collaboration.

Imagine you are a user of an Open Source project like Numpy and find a bug in one of their methods.

You can inspect and clone Numpy's code in GitHub <https://github.com/numpy/numpy>, play around a bit and find how to fix the bug.

Numpy has done so much for you asking nothing in return, that you really want to contribute back by fixing the bug for them.

You make all of the changes but you can't push it back to Numpy's repository because you don't have permissions.

The right way to do this is **forking Numpy's repository**.

## Forking a repository on GitHub

By forking a repository, all you do is make a copy of it in your GitHub account, where you will have write permissions as well.

If you fork Numpy's repository, you will find a new repository in your GitHub account that is an exact copy of Numpy. You can then clone it to your computer, work locally on fixing the bug and push the changes to your *fork* of Numpy.

Once you are happy with the changes, GitHub also offers you a way to notify Numpy's developers of this changes so that they can include them in the official Numpy repository via starting a **Pull Request**.

## Pull Request

You can create a Pull Request and select those changes that you think can be useful for fixing Numpy's bug.

Numpy's developers will review your code and make comments and suggestions on your fix. Then, you can commit more improvements in the pull request for them to review and so on.

Once Numpy's developers are happy with your changes, they'll accept your Pull Request and merge the changes into their original repository, for everyone to use.

## Practical example - Team up!

We will be working in the same repository with one of you being the leader and the other being the collaborator.

Collaborators need to go to the leader's GitHub profile and find the repository we created for that lesson. Mine is in <https://github.com/jamespjh/github-example>

### 1. Fork repository

You will see on the top right of the page a **Fork** button with an accompanying number indicating how many GitHub users have forked that repository.

Collaborators need to navigate to the leader's repository and click the **Fork** button.

Collaborators: note how GitHub has redirected you to your own GitHub page and you are now looking at an exact copy of the team leader's repository.

### 2. Clone your forked repo

Collaborators: go to your terminal and clone the newly created fork.

```
git clone git@github.com:jamespjh/github-example.git
```

### 3. Create a feature branch

It's a good practice to create a new branch that'll contain the changes we want. We'll learn more about branches later on. For now, just think of this as a separate area where our changes will be kept not to interfere with other people's work.

```
git checkout -b southwest
```

### 4. Make, commit and push changes to new branch

For example, let's create a new file called `SouthWest.md` and edit it to add this text:

```
* Exmoor  
* Dartmoor  
* Bodmin Moor
```

Save it, and push this changes to your fork's new branch:

```
git add SouthWest.md  
git commit -m "The South West is also hilly."  
git push origin southwest
```

### 5. Create Pull Request

Go back to the collaborator's GitHub site and reload the fork. GitHub has noticed there is a new branch and is presenting us with a green button to [Compare & pull request](#). Fantastic! Click that button.

Fill in the form with additional information about your change, as you consider necessary to make the team leader understand what this is all about.

Take some time to inspect the commits and the changes you are submitting for review. When you are ready, click on the [Create Pull Request](#) button.

Now, the leader needs to go to their GitHub site. They have been notified there is a pull request in their repo awaiting revision.

### 6. Feedback from team leader

Leaders can see the list of pull requests in the vertical menu of the repo, on the right hand side of the screen. Select the pull request the collaborator has done, and inspect the changes.

There are three tabs: in one you can start a conversation with the collaborator about their changes, and in the others you can have a look at the commits and changes made.

Go to the tab labeled as "Files Changed". When you hover over the changes, a small [+](#) button appears. Select one line you want to make a comment on. For example, the line that contains "Exmoor".

GitHub allows you to add a comment about that specific part of the change. Your collaborator has forgotten to add a title at the beginning of the file right before "Exmoor", so tell them so in the form presented after clicking the [+](#) button.

### 7. Fixes by collaborator

Collaborators will be notified of this comment by email and also in their profiles page. Click the link accompanying this notification to read the comment from the team leader.

Go back to your local repository, make the changes suggested and push them to the new branch.

Add this at the beginning of your file:

```
Hills in the South West:  
=====
```

Then push the change to your fork:

```
git add .  
git commit -m "Titles added as requested."  
git push origin southwest
```

This change will automatically be added to the pull request you started.

## 8. Leader accepts pull request

The team leader will be notified of the new changes that can be reviewed in the same fashion as earlier.

Let's assume the team leader is now happy with the changes.

Leaders can see in the "Conversation" tab of the pull request a green button labelled **Merge pull request**. Click it and confirm the decision.

The collaborator's pull request has been accepted and appears now in the original repository owned by the team leader.

Fork and Pull Request done!

## Some Considerations

- Fork and Pull Request are things happening only on the repository's server side (GitHub in our case). Consequently, you can't do things like `git fork` or `git pull-request` from the local copy of a repository.
- You don't always need to fork repositories with the intention of contributing. You can fork a library you use, install it manually on your computer, and add more functionality or customise the existing one, so that it is more useful for you and your team.
- Numpy's example is only illustrative. Normally, Open Source projects have in their documentation (sometimes in the form of a wiki) a set of instructions you need to follow if you want to contribute to their software.
- Pull Requests can also be done for merging branches in a non-forked repository. It's typically used in teams to merge code from a branch into the master branch and ask team colleagues for code reviews before merging.
- It's a good practice before starting a fork and a pull request to have a look at existing forks and pull requests. On GitHub, you can find the list of pull requests on the horizontal menu on the top of the page. Try to also find the network graph displaying all existing forks of a repo, like this example in the NumpyDoc repo: <https://github.com/numpy/numpydoc/network>

## Git Theory

### The revision Graph

Revisions form a **GRAPH**

```
import os  
  
top_dir = os.getcwd()  
git_dir = os.path.join(top_dir, "learning_git")  
working_dir = os.path.join(git_dir, "git_example")  
os.chdir(working_dir)  
  
%%bash  
git log --graph --oneline
```



```
* b3b0bad Revert ":bug: Reset git merge"
* 73ba9b1 TMP test build
* 1318b35 :bug: Reset git merge
* 300f122 :bug: Fixed GitHub building bugs
* d5cb1f2 :bug: Fixed notebook links
* 341ab0e :arrow_up: Updated requirements
* 8a446fc :art: Clarified prerequisites and self-assesment. Reformatted plain
Markdown files.
* 4280ac0 :art: Renamed notebooks into standard format. Split modules 1&2 into three.
* 2f27115 :art: Refactored module names
* f1d8042 Merge pull request #45 from alan-turing-institute/44-fix-gh-pages
|\
| * 39ff4f3 :bug: Add a second .nojekyll directive which seems to be required
| * 4ed15ee :bug: Fix the PDF download location since '/' points to https://alan-
turing-institute.github.io/ rather than the repo-specific document root
| * 2323d59 :bug: Fix the GitHub Pages Jekyll build
|/
* 0d14a20 Merge pull request #41 from alan-turing-institute/jupyter-book
|\
| * d194e6b :construction_worker: Switched to building on master and pushing to gh-
pages
| * c014315 :memo: Updated README
| * acd935b :coffin: Removed autogenerated DiffusionExample code
| * e21b995 :coffin: Removed energy_example
| * 1575e17 :coffin: Removed auto-generated diffusion outputs
| * 2bffc43 :coffin: Removed auto-generated saskatchewan outputs
| * a19d45e :memo: Simplified introductory pages
| * 5817309 :sparkles: Added PDF building
| * d515f3f :coffin: Removed greetings documentation
| * c2582d6 :memo: Fixed Markdown header levels
| * aae3056 :memo: Marked cells which raise exceptions
| * 58302c2 :memo: Added prerequisites section
| * 655cff5 :coffin: Removed mp4 output for boids
| * 7d2f2d0 :recycle: Move setup commands into Makefile
| * 62afabb :coffin: Removed unnecessary files
| * 16ff5d4 :lipstick: Added Turing logo (see #1). Added sections.
| * 5cdee87 :recycle: Added a redirecting index page
| * a60ef6e :construction_worker: Switch to GitHub Actions for documentation building
| * 059c089 :memo: Updated to jupyter-book build
| * 60136f7 :lipstick: Ran all notebooks with Python 3.7.9
| * bc100f6 :rotating_light: Ran black over the codebase
| * 8ddad41 :pushpin: Pinned all dependencies
| * cfaf328 :sparkles: Added basic jupyter-book skeleton
| * 1268477 :coffin: Remove unnecessary autogenerated files
|/
* d0d90be Merge pull request #35 from alan-turing-institute/ch05-fixes
|\
| * 8acbaa2 Add a section on Pseudocode Programming Process, restore the Robert Martin
quote with a caveat
| * 1caf0db Remove quote from Robert Martin about comments always being failures, add
section on who comments are for, and replace 'bad comments' with 'possibly
unnecessarily/extraneous comments'
| * bf71f7f fix a couple of yuml calls
| * d179b7b add SVG for yuml function
| * 2ea9632 fix 'type' -> 'kind' in Bird constructor
* | a822844 Merge pull request #40 from alan-turing-institute/day10-updates
|\
| * | 0ba9b88 Updated template syntax
| * | 25cc5f0 Updated https links
| * | b5dda65 Moved element to atom in chapter 11
| * | 38d3116 Changed name of 'element' element to atom. This avoids confusion with XML
elements.
| * | bb6b2ea Added XPath explanation
| * | d4cc722 Standardised mko formatting. Added explanations for some of the XPath
constructs.
| * | 6e274e5 Expanded XSLT comments
|/
* | edce93b Merge pull request #39 from alan-turing-institute/day9-updates
|\
| * | 67dfc36 Print limited amounts of output files
| * | 46490d1 Apply @edaub's suggestions from code review
| * | aeedaa92 Toned down some of the text (eg. 'easy', 'stupid') and tidied up the NF
explanations
| * | 9862534 Added Yacc explanation, expanded system to allow Element * Molecule
| * | 81af404 Added explanation of how Yacc parsing works
| * | 69f1d76 Added link to the Python grammar
| * | 46ffb6f Fixed lexer regex and added example of parsing failure
| * | 1b7eba3 Added explanations of file format types. Added demonstration of binary
outputs.
| * | 17a5ea9 Added normal form explanation. Fixed incorrect chemical formulae!
|/
* | 2f5e131 Merge pull request #28 from alan-turing-institute/day4-post-teaching-
fixes
|\
| | 18a3aa0 Merge branch 'master' into day4-post-teaching-fixes
```

```
| \\  
| /  
|/  
* | 7409bf5 Merge pull request #34 from alan-turing-institute/ch04-fixes  
| \\  
| * | 0700cb4 Prefer pip install over setup.py install  
| /  
* | d12a6c3 Merge pull request #31 from alan-turing-institute/check-ch04-lessons  
| \\  
| * | 99ff41d Re-run of ch04 notebooks  
| /  
* | 20965ba Merge pull request #26 from alan-turing-institute/missing_buckets  
| \\  
| * | 0e2e633 Add missing brackets in MagicMock construction.  
* | | c19f5f8 Merge pull request #30 from jack89roberts/master  
| \\  
| /  
|/  
* | 1b8e3df Restore 06Documentation.ipynb notebook  
| /  
| * 4c7d857 present tests before implementation solution  
| * 2dfd8f8 some more comments on first module entries of %%bash  
| * b2849c4 move numpy import to outside function  
| * 3b038b8 add %%cmd comments to chp2  
| * e0d96a2 add comment on %%cmd when first introducing %%bash for git module  
| * 6565051 add windows alternative to bash commands in testing  
|/  
* 4033d00 Merge pull request #22 from alan-turing-institute/day3-post-teaching-fixes  
| \  
| * c5bcb3a Removed CalledProcessError for known cell failures  
| * 9131c3d Updated default names in all notebooks  
| * 5ed9c46 Added default name of 'main' for initial branch  
| * ac9cf96 Move git config commands from code to Markdown as they fail during the HTML  
build process  
| * 6034ee0 Fixed cell that was incorrectly labelled as code rather than Markdown  
| * 2ad76f3 another git checkout edit  
| * 3f179a2 correct more git checkout wsd entries  
| * 4c8c40b correct git checkout in wsdiagram  
* | 5db9209 Merge pull request #24 from alan-turing-institute/fix-testing-module  
| \  
| * | b84e972 Updated ch03 tests  
| /  
* | 8ac0d2d Merge pull request #23 from alan-turing-institute/packaging-lesson-fixes  
| \  
| * | 1ead20c Simplified gitignore  
| * | 15997a5 Restored docs/output folder  
| * | 00d3ab0 Moved all docs into doc/ directory  
| * | 42f11b7 Include all ch04 artifacts except eggs and doctrees  
| * | 0a8d8d6 Updated documentation notebook  
| * | d086525 Regenerated ch04 files  
| * | 231a00e Renamed and renumbered notebooks  
| /  
* | 3d087eb Merge pull request #21 from alan-turing-institute/file-format-fixes  
| \  
| /  
|/  
| * 7fada36 Simplified logic  
| * 1c2fdd3 Removed unused PNGs  
| * 5636bdc Formatted notebooks and fixed Molecule class  
|/  
* f7cf2ac Merge pull request #20 from alan-turing-institute/day2-post-teaching-fixes  
| \  
| * 0eb75ad Additional black reformatting  
| * 13a7cb5 Removed further old-style classes  
| * 1514309 Removed old-style objects and formatted with black  
| * cfb3977 Renamed functions to match position in course  
| * 9f2d28a reduce image size of broadcasting example in 082NumPy  
| * 5d9be93 Add another broadcasting example to 082NumPy  
| * 1985c2e Remove old class Foo(object) syntax in class definitions  
| * ce52d37 04functions: Indentation and remove spaces in keyword argument definitions  
| * 5705ff7 Added native @ dot product example  
| * bec5bd0 Switch local back to global now that these cells are not being executed  
| * 983a984 Add additional %% example  
| * 2edc41a Moved git config commands from code-cells to Markdown cells to stop the  
build from changing global git settings  
| * 15309f4 Make the new-axis process more explicit by assigning to a variable  
| * 9cdaaf9 Changed global to local in git lessons  
| * 3937657 Checked for index.md references  
| * 81b8df0 Fixed explanatory note  
| * 32c24ec Add exercise subheadings  
| * 919308a Reordered investigation steps  
| * 6a61ae2 Added note about request.json()  
| * a6f9654 Show raw text output  
| * c17148b Use context-manager when loading YAML file  
| * 214415a Clarified some wording that tripped me up  
|/  
* 179128f Merge pull request #19 from alan-turing-institute/fix-gh-pages-build
```

```
|\\
| * 880eb76 Force use of dpl v2 which will soon become the default
|/
* 4332153 Merge pull request #16 from otnemrasordep/master
|\\
| * 0c4c087 Update 023types.ipynb
* | e830d2b Merge pull request #18 from jack89roberts/travis-make-check
|\\ \
| * | 7f5c959 add comment explaining make update
| * | 38226b8 add sudo to make install
| * | 82604a6 update make to 4.3 by installing from source
|// /
* | cd06bf6 Merge pull request #17 from alan-turing-institute/fix-travis-errors
|\\ \
| * | e17014e Changed index.md to test.md in Git chapter
| * | 3380b29 Fixed deprecated Travis options
|// /
* | 8c8c93c Merge pull request #15 from alan-turing-institute/cleanup-repo
|\\ \
| * | fe9d159 Updated git changes to use --local
| * | 933f195 Added links to courses from main page
| * | c02f9c5 Updated to gem versions used on GitHub pages
(https://pages.github.com/versions/)
|// /
* | 03e36a7 Merge pull request #14 from alan-turing-institute/markdown-fixes
|\\ \
| | /
|/
| * d0563c1 Updated Markdown formatting
* | 2a2d86f Merge pull request #13 from jack89roberts/research-data-update-notes
|\\ \
| | /
|/
| * ecbae5e A few improvements to notes in research data in python
|/
* f586134 Merge pull request #11 from jack89roberts/python-3.8
|\\
| * cbef3d8 Update python version in travis config
| * 31d5928 Re-run notebooks on Python 3.8
| * eec8350 Makefile indentation
| * 3d7df9a Update ppa:mc3man/trusty-media -> ppa:mc3man/bionic-media
| * 6bf911a remove latex-color package not available on bionic
| * b9e7eeb update travis dist to bionic
| * 3986b0f Update debugger link to python 3.8
| * 4922d7f Update docs to say python 3.8
|/
* 3710908 Merge pull request #10 from jack89roberts/giovanni-to-jack
|\\
| * 4bedc8e Re-run git notebooks
| * c317630 Add missing git add command
| * fcddcd16 Add missing git stash command
| * dd132bb Replace Giovanni with Jack for git examples
|/
* c8436b7 trigger rebuild
* 9a593c0 Merge pull request #8 from jack89roberts/master
|\\
| * bd8c86e Fix typo: Supplementary -> Supplementary
| * 79fd770 Fix missing outputs in a few cells
| * 3793448 Merge branch 'fix-nbconvert-version' into master
| | \
| * | 96a4925 Fixes to Research Data in Python lesson
* | | db7e4f9 Change email in Travis config to hut23
* | | 5521eb2 Merge pull request #9 from jack89roberts/fix-nbconvert-version
|\\ \
| | /
|/
| | /
| * c2cb18c Upgrade numpy version to avoid matplotlib runtime error
| * 15ae54c Set nbconvert version to be below v6
|/
* 5af9179 Merge pull request #6 from alan-turing-institute/minor-text-fixes
|\\
| * 01a5ffc Removed extra spaces
| * c07c303 Removed extra spaces
| * fe7117c Elaborated on Python2 vs Python3 changes
| * 807e7c0 Fix typo in James' name
| * 22a49f4 Fixed missing bracket
|/
* 112d970 Modify geocoder settings for 2019 delivery
* 8598b99 Change install instruction to Anaconda.
* dbe0a76 Turn off the PDF build for now to get this building again
* f4574bc Add a tex package to fix the build
* 8f733bc Update index.md
* 7c457ea Update index.md
* 46dd2ca Merge branch 'master' of https://github.com/alan-turing-institute/rsd-engineeringcourse
|\\
```

```

| * 40fae49 Use -inf correctly for reduce example
| * 7e8a9c1 Merge branch 'master' of github.com:alan-turing-institute/rsd-
engineeringcourse
| \\
| * \ 46fbbe1 Merge branch 'master' of github.com:alan-turing-institute/rsd-
engineeringcourse
| \\
| * \ \ fd7cd8a Merge branch 'master' of github.com:alan-turing-institute/rsd-
engineeringcourse
| \\
| * | | 6853baf Fix classes bug thanks to @hhau
* | | | e5d6ac0 updated reqs
| |
| |
* | | 78e338f removed old 0500operators
* | | 681d413 merging classes 5,7,8 from UCL
* | | 6c99e20 update schedule
| |
| |
* | | da23a22 fix to requirements
* | | 421ff20 pulled ch05 from UCL
* | | 91f1f59 updated schedule
| |
| |
* | 0abdab7 requirements
* | ad5ea5d checked ch04
* | c6bd2ae Notes from local
* | eafebb6 fix in stashing code to avoid keys getting public
* | fd36989 update to schedule
* | 9beb9c7 removed ref to UCL class
* | 120555e auth, messy auth
* | 0a40b95 auth, messy auth
* | c692086 fixing auth still
|
* ccf8dd8 git auth hopefully fixed
* ccad62f auth
* a61e0b9 auth
* 3118652 auth
* 345f358 auth
* 6b8044d auth
* 1e33f2c auth
* 5942cae git auth
* eb32fc3 git auth
* 36d381b git auth
* 27ae7be fix ch02
* 26e80a0 fix 02
* 49d97ff updated and checked weeks 2 to 4
* eac3f2f updated schedule
* fc73f61 lesson 1 checked
* d39e881 minor tweaks to Install
* 7c8e554 delete previous folders
* dbc617a merging last 2 classes
* 77a9e9e new chapters
* d6e35c7 added last two classes and updated index
* adae052 added assessment
* 74f0224 jekyll version making troubles
* d3209e8 jekyll version making troubles
* 2208e4b added imageio to req
* 48c441a dropping UCL crumbs
* cbf5478 adding missing packages to req, moving to 3.6
* 9c2f115 update jekyll and liquid
* 76c3d82 add comment
* 5314bb9 typo
* 0086e45 first commit

```

## Git concepts

- Each revision has a parent that it is based on
- These revisions form a graph
- Each revision has a unique hash code
  - In Sue's copy, revision 43 is ab3578d6
  - Jim might think that is revision 38, but it's still ab3579d6
- Branches, tags, and HEAD are *labels* pointing at revisions
- Some operations (like fast forward merges) just move labels.

## The levels of Git

There are four **Separate** levels a change can reach in git:

- The Working Copy
- The **index** (aka **staging area**)
- The local repository
- The remote repository

Understanding all the things `git reset` can do requires a good grasp of git theory.

- `git reset <commit> <filename>`: Reset index and working version of that file to the version in a given commit
- `git reset --soft <commit>`: Move local repository branch label to that commit, leave working dir and index unchanged
- `git reset <commit>`: Move local repository and index to commit ("–mixed")
- `git reset --hard <commit>`: Move local repository, index, and working directory copy to that state

## Branches

Branches are incredibly important to why `git` is cool and powerful.

They are an easy and cheap way of making a second version of your software, which you work on in parallel, and pull in your changes when you are ready.

```
import os

top_dir = os.getcwd()
git_dir = os.path.join(top_dir, "learning_git")
working_dir = os.path.join(git_dir, "git_example")
os.chdir(working_dir)
```

```
%%bash
git branch # Tell me what branches exist
```

```
* 47-reorganise-chapters
```

```
%%bash
git checkout -b experiment # Make a new branch
```

```
Switched to a new branch 'experiment'
```

```
%%bash
git branch
```

```
47-reorganise-chapters
* experiment
```

```
%%writefile Wales.md
Mountains In Wales
=====
* Pen y Fan
* Tryfan
* Snowdon
* Glyder Fawr
* Fan y Big
* Cadair Idris
```

```
Overwriting Wales.md
```

```
%%bash
git commit -am "Add Cadair Idris"
```

```
On branch experiment
nothing to commit, working tree clean
```

```
-----  
CalledProcessError                                                 Traceback (most recent call last)  
/tmp/ipykernel_3468/1177132796.py in <module>  
----> 1 get_ipython().run_cell_magic('bash', '', 'git commit -am "Add Cadair Idris"\n')  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/interactiveshell.py in run_cell_magic(self, magic_name, line,  
cell)  
    2404         with self.builtin_trap:  
    2405             args = (magic_arg_s, cell)  
-> 2406             result = fn(*args, **kwargs)  
    2407             return result  
    2408  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magics/script.py in named_script_magic(line, cell)  
    140         else:  
    141             line = script  
--> 142             return self.shebang(line, cell)  
    143  
    144     # write a basic docstring:  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/decorator.py in  
fun(*args, **kw)  
    230         if not kwsyntax:  
    231             args, kw = fix(args, kw, sig)  
--> 232             return caller(func, *(extras + args), **kw)  
    233     fun.__name__ = func.__name__  
    234     fun.__doc__ = func.__doc__  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magic.py in <lambda>(f, *a, **k)  
    185     # but it's overkill for just that one bit of state.  
    186     def magic_deco(arg):  
--> 187         call = lambda f, *a, **k: f(*a, **k)  
    188  
    189         if callable(arg):  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magics/script.py in shebang(self, line, cell)  
    243         sys.stderr.flush()  
    244         if args.raise_error and p.returncode!=0:  
--> 245             raise CalledProcessError(p.returncode, cell, output=out,  
stderr=err)  
    246  
    247     def _run_script(self, p, cell, to_close):  
  
CalledProcessError: Command 'b'git commit -am "Add Cadair Idris"\n' returned non-zero  
exit status 1.
```

```
%%bash  
git checkout main # Switch to an existing branch
```

```
Your branch is up to date with 'origin/main'.
```

```
Switched to branch 'main'
```

```
%%bash  
cat Wales.md
```

```
Mountains In Wales  
=====
```

- \* Pen y Fan
- \* Tryfan
- \* Snowdon
- \* Fan y Big
- \* Glyder Fawr

```
%%bash  
git checkout experiment
```

```
Switched to branch 'experiment'
```

```
cat Wales.md
```

```
Mountains In Wales  
=====
```

- \* Pen y Fan
- \* Tryfan
- \* Snowdon
- \* Glyder Fawr
- \* Fan y Big
- \* Cadair Idris

## Publishing branches

To let the server know there's a new branch use:

```
%%bash  
git push -u origin experiment
```

```
Branch 'experiment' set up to track remote branch 'experiment' from 'origin'.
```

```
remote:  
remote: Create a pull request for 'experiment' on GitHub by visiting:  
remote:     https://github.com/alan-turing-institute/github-  
example/pull/new/experiment  
remote:  
To github.com:alan-turing-institute/github-example.git  
 * [new branch]   experiment -> experiment
```

We use `--set-upstream origin` (Abbreviation `-u`) to tell git that this branch should be pushed to and pulled from origin per default.

If you are following along, you should be able to see your branch in the list of branches in GitHub.

Once you've used `git push -u` once, you can push new changes to the branch with just a `git push`.

If others checkout your repository, they will be able to do `git checkout experiment` to see your branch content, and collaborate with you **in the branch**.

```
%%bash  
git branch -r
```

```
origin/experiment  
origin/main
```

Local branches can be, but do not have to be, connected to remote branches. They are said to “track” remote branches. `push -u` sets up the tracking relationship. You can see the remote branch for each of your local branches if you ask for “verbose” output from `git branch`:

```
%%bash  
git branch -vv
```

```
* experiment a4378e5 [origin/experiment] Add Cadair Idris  
  main      85b2797 [origin/main] Merge branch 'main' of github.com:alan-turing-  
institute/github-example
```

## Find out what is on a branch

In addition to using `git diff` to compare to the state of a branch, you can use `git log` to look at lists of commits which are in a branch and haven't been merged yet.

```
%%bash  
git log main..experiment
```

```
commit a4378e56a723f2ba662262f94108a74be5f896f0
Author: Turing Developer <developer@example.com>
Date:   Mon Nov 8 14:06:02 2021 +0000
```

```
Add Cadair Idris
```

Git uses various symbols to refer to sets of commits. The double dot `A..B` means “ancestor of B and not ancestor of A”

So in a purely linear sequence, it does what you'd expect.

```
%%bash
git log --graph --oneline HEAD~9..HEAD~5
```

```
* 34f8bbb Merge branch 'main' of github.com:alan-turing-institute/github-example
|\
| * 289afed Add Scotland
* | 0c976b0 Add wales
|/
* 82b4fa0 Add Helvellyn
* 6ff088f Include lakes in the scope
```

But in cases where a history has branches, the definition in terms of ancestors is important.

```
%%bash
git log --graph --oneline HEAD~5..HEAD
```

```
* a4378e5 Add Cadair Idris
* 85b2797 Merge branch 'main' of github.com:alan-turing-institute/github-example
|\
| * ed30178 Add another Beacon
* | bc04a83 Add Glyder
|/
* ecc3206 Merge branch 'main' of github.com:alan-turing-institute/github-example
|\
| * 8cd684e Add a beacon
* b8fb6c5 Translating from the Welsh
```

If there are changes on both sides, like this:

```
%%bash
git checkout main
```

```
Your branch is up to date with 'origin/main'.
```

```
Switched to branch 'main'
```

```
%%writefile Scotland.md
Mountains In Scotland
=====
* Ben Eigne
* Cairngorm
* Aonach Eagach
```

```
Overwriting Scotland.md
```

```
%%bash
git diff Scotland.md
```

```
diff --git a/Scotland.md b/Scotland.md
index 9613dda..bf5c643 100644
--- a/Scotland.md
+++ b/Scotland.md
@@ -3,3 +3,4 @@ Mountains In Scotland
 * Ben Eigne
 * Cairngorm
+* Aonach Eagach
```

```
%%bash
git commit -am "Commit Aonach onto main branch"
```

```
[main 537950c] Commit Aonach onto main branch
 1 file changed, 1 insertion(+)
```

Then this notation is useful to show the content of what's on what branch:

```
%%bash
git log --left-right --oneline main...experiment
```

```
< 537950c Commit Aonach onto main branch
> a4378e5 Add Cadair Idris
```

Three dots means “everything which is not a common ancestor” of the two commits, i.e. the differences between them.

## Merging branches

We can merge branches, and just as we would pull in remote changes, there may or may not be conflicts.

```
%%bash
git branch
git merge experiment
```

```
experiment
* main
Merge made by the 'recursive' strategy.
 Wales.md | 3 ++
 1 file changed, 2 insertions(+), 1 deletion(-)
```

```
%%bash
git log --graph --oneline HEAD~3..HEAD
```

```
* 2365c66 Merge branch 'experiment'
|\ 
| * a4378e5 Add Cadair Idris
* | 537950c Commit Aonach onto main branch
|/
* 85b2797 Merge branch 'main' of github.com:alan-turing-institute/github-example
* ed30178 Add another Beacon
```

## Cleaning up after a branch

```
%%bash
git branch
```

```
experiment
* main
```

```
%%bash
git branch -d experiment
```

```
Deleted branch experiment (was a4378e5).
```

```
%%bash
git branch
```

```
* main
```

```
%%bash
git branch --remote
```

```
origin/experiment  
origin/main
```

```
%%bash  
git push --delete origin experiment  
# Remove remote branch. Note that you can also use the GitHub interface to do this.
```

```
To github.com:alan-turing-institute/github-example.git  
- [deleted] experiment
```

```
%%bash  
git branch --remote
```

```
origin/main
```

## A good branch strategy

- A **production** branch: code used for active work
- A **develop** branch: for general new code
- **feature** branches: for specific new ideas
- **release** branches: when you share code with others
  - Useful for isolated bug fixes

## Grab changes from a branch

Make some changes on one branch, switch back to another, and use:

```
git checkout <branch> <path>
```

to quickly grab a file from one branch into another. This will create a copy of the file as it exists in **<branch>** into your current branch, overwriting it if it already existed. For example, if you have been experimenting in a new branch but want to undo all your changes to a particular file (that is, restore the file to its version in the **main** branch), you can do that with:

```
git checkout main test_file
```

Using **git checkout** with a path takes the content of files. To grab the content of a specific *commit* from another branch, and apply it as a patch to your branch, use:

```
git cherry-pick <commit>
```

## Git Stash

Before you can **git pull**, you need to have committed any changes you have made. If you find you want to pull, but you're not ready to commit, you have to temporarily "put aside" your uncommitted changes. For this, you can use the **git stash** command, like in the following example:

```
import os  
  
top_dir = os.getcwd()  
git_dir = os.path.join(top_dir, "learning_git")  
working_dir = os.path.join(git_dir, "git_example")  
os.chdir(working_dir)
```

```
%%writefile Wales.md
```

```
Mountains In Wales
```

```
=====
```

- \* Pen y Fan
- \* Tryfan
- \* Snowdon
- \* Glyder Fawr
- \* Fan y Big
- \* Cadair Idris
- \* Penygader

```
Overwriting Wales.md
```

```
%%bash
```

```
git stash
```

```
No local changes to save
```

```
%%bash
```

```
git pull
```

```
There is no tracking information for the current branch.  
Please specify which branch you want to merge with.  
See git-pull(1) for details.
```

```
git pull <remote> <branchn>
```

```
If you wish to set tracking information for this branch you can do so with:
```

```
git branch --set-upstream-to=origin/<branchn> experiment
```

```
-----  
CalledProcessError                                  Traceback (most recent call last)  
/tmp/ipykernel_3498/3022008257.py in <module>  
--> 1 get_ipython().run_cell_magic('bash', '', 'git pull\n')  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/interactiveshell.py in run_cell_magic(self, magic_name, line,  
cell)  
    2404         with self.builtin_trap:  
    2405             args = (magic_arg_s, cell)  
-> 2406             result = fn(*args, **kwargs)  
    2407             return result  
    2408  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magics/script.py in named_script_magic(line, cell)  
    140     else:  
    141         line = script  
--> 142     return self.shebang(line, cell)  
    143  
    144     # write a basic docstring:  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/decorator.py in  
fun(*args, **kw)  
    230         if not kwsyntax:  
    231             args, kw = fix(args, kw, sig)  
--> 232         return caller(func, *(extras + args), **kw)  
    233     fun.__name__ = func.__name__  
    234     fun.__doc__ = func.__doc__  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magic.py in <lambda>(f, *a, **k)  
    185     # but it's overkill for just that one bit of state.  
    186     def magic_deco(arg):  
--> 187         call = lambda f, *a, **k: f(*a, **k)  
    188  
    189     if callable(arg):  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/magics/script.py in shebang(self, line, cell)  
    243         sys.stderr.flush()  
    244         if args.raise_error and p.returncode!=0:  
--> 245             raise CalledProcessError(p.returncode, cell, output=out,  
stderr=err)  
    246  
    247     def _run_script(self, p, cell, to_close):  
  
CalledProcessError: Command 'b'git pull\n'' returned non-zero exit status 1.
```

By stashing your work first, your repository becomes clean, allowing you to pull. To restore your changes, use `git stash apply`.

```
%%bash  
git stash apply
```

```
On branch main  
Your branch is up to date with 'origin/main'.  
  
Changes not staged for commit:  
(use "git add <file>..." to update what will be committed)  
(use "git restore <file>..." to discard changes in working directory)  
      modified:   Wales.md  
  
Untracked files:  
(use "git add <file>..." to include in what will be committed)  
      __pycache__/  
      wsd.py  
  
no changes added to commit (use "git add" and/or "git commit -a")
```

The “Stash” is a way of temporarily saving your working area, and can help out in a pinch.

# Tagging

Tags are easy to read labels for revisions, and can be used anywhere we would name a commit.

Produce real results *only* with tagged revisions.

NB: we delete previous tags with the same name remotely and locally first, to avoid duplicates.

```
git tag -a v1.0 -m "Release 1.0"  
git push --tags
```

You can also use tag names in the place of commit hashes, such as to list the history between particular commits:

```
git log v1.0... --graph --oneline
```

If .. is used without a following commit name, HEAD is assumed.

## Working with generated files: gitignore

We often end up with files that are generated by our program. It is bad practice to keep these in Git; just keep the sources.

Examples include .o and .x files for compiled languages, .pyc files in Python.

In our example, we might want to make our .md files into a PDF with pandoc:

```
%%writefile Makefile  
  
MDS=$(wildcard *.md)  
PDFS=$(MDS:.md=.pdf)  
  
default: $(PDFS)  
  
.pdf: %.md  
    pandoc $< -o $@
```

Writing Makefile

```
%%bash  
make
```

```
pandoc Scotland.md -o Scotland.pdf  
pandoc Wales.md -o Wales.pdf  
pandoc lakeland.md -o lakeland.pdf  
pandoc test.md -o test.pdf
```

We now have a bunch of output .pdf files corresponding to each Markdown file.

But we don't want those to show up in git:

```
%%bash  
git status
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   Wales.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Makefile
    Scotland.pdf
    Wales.pdf
    __pycache__/
    lakeland.pdf
    test.pdf
    wsd.py

no changes added to commit (use "git add" and/or "git commit -a")
```

Use .gitignore files to tell Git not to pay attention to files with certain paths:

```
%%writefile .gitignore
*.pdf
```

```
Writing .gitignore
```

```
%%bash
git status
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   Wales.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    Makefile
    __pycache__/
    wsd.py

no changes added to commit (use "git add" and/or "git commit -a")
```

```
%%bash
git add Makefile
git add .gitignore
git commit -am "Add a makefile and ignore generated files"
git push
```

```
[main c8ba483] Add a makefile and ignore generated files
 3 files changed, 12 insertions(+), 1 deletion(-)
 create mode 100644 .gitignore
 create mode 100644 Makefile
```

```
To github.com:alan-turing-institute/github-example.git
 537950c..c8ba483  main -> main
```

## Git clean

Sometimes you end up creating various files that you do not want to include in version control. An easy way of deleting them (if that is what you want) is the `git clean` command, which will remove the files that git is not tracking.

```
%%bash
git clean -fX
```

```
Removing Scotland.pdf
Removing Wales.pdf
Removing lakeland.pdf
Removing test.pdf
```

```
%%bash
ls
```

```
Makefile
Scotland.md
Wales.md
__pycache__
lakeland.md
test.md
wsd.py
```

- With -f: don't prompt
- with -d: remove directories
- with -x: Also remote .gitignored files
- with -X: Only remove .gitignore files

## Hunks

### Git Hunks

A “Hunk” is one git change. This changeset has three hunks:

```
+import matplotlib
+import numpy as np

from matplotlib import pylab
from matplotlib.backends.backend_pdf import PdfPages

+def increment_or_add(key,hash,weight=1):
+    if key not in hash:
+        hash[key]=0
+    hash[key]+=weight
+
data_path=os.path.join(os.path.dirname(
    os.path.abspath(__file__)),
-regenerate=False
+regenerate=True
```

### Interactive add

`git add` and `git reset` can be used to stage/unstage a whole file, but you can use interactive mode to stage by hunk, choosing yes or no for each hunk.

```
git add -p myfile.py
```

```
+import matplotlib
+import numpy as np
#Stage this hunk [y,n,a,d/,j,J,g,e,?]?
```

## GitHub pages

### Yaml Frontmatter

GitHub will publish repositories containing markdown as web pages, automatically.

You'll need to add this content:

```
---
```

A pair of lines with three dashes, to the top of each markdown file. This is how GitHub knows which markdown files to make into web pages. [Here's why](#) for the curious.

```
%%writefile test.md
---
title: Github Pages Example
---
Mountains and Lakes in the UK
=====
Engerland is not very mountainous.
But has some tall hills, and maybe a mountain or two depending on your definition.
```

Overwriting test.md

```
%%bash
git commit -am "Add github pages YAML frontmatter"
```

```
[main 12ee6ad] Add github pages YAML frontmatter
 1 file changed, 7 insertions(+), 4 deletions(-)
```

## The gh-pages branch

GitHub creates github pages when you use a special named branch. By default this is **gh-pages** although you can change it to something else if you prefer. This is best used to create documentation for a program you write, but you can use it for anything.

```
os.chdir(working_dir)
```

```
%%bash
git checkout -b gh-pages
git push -uf origin gh-pages
```

```
Branch 'gh-pages' set up to track remote branch 'gh-pages' from 'origin'.
```

```
Switched to a new branch 'gh-pages'
remote:
remote: Create a pull request for 'gh-pages' on GitHub by visiting:
remote:     https://github.com/alan-turing-institute/github-example/pull/new/gh-pages
remote:
To github.com:alan-turing-institute/github-example.git
 * [new branch]      gh-pages -> gh-pages
```

The first time you do this, GitHub takes a few minutes to generate your pages.

The website will appear at <http://username.github.io/repositoryname>, for example:

<http://alan-turing-institute.github.io/github-example/>

## Layout for GitHub pages

You can use GitHub pages to make HTML layouts, here's an [example of how to do it](#), and [how it looks](#). We won't go into the detail of this now, but after the class, you might want to try this.

```
%%bash
# Cleanup by removing the gh-pages branch
git checkout main
git push
git branch -d gh-pages
git push --delete origin gh-pages
git branch --remote
```

```
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)
Deleted branch gh-pages (was 12ee6ad).
origin/main
```

```
Switched to branch 'main'
To github.com:alan-turing-institute/github-example.git
  c8ba483..12ee6ad  main -> main
To github.com:alan-turing-institute/github-example.git
  - [deleted]        gh-pages
```

## Working with multiple remotes

### Distributed versus centralised

Older version control systems (cvs, svn) were “centralised”; the history was kept only on a server, and all commits required an internet.

#### Centralised

Server has history

#### Distributed

Every user has full history

Your computer has one snapshot

Many local branches

To access history, need internet

History always available

You commit to remote server

Users synchronise histories

cvs, subversion(svn)

git, mercurial (hg), bazaar (bzr)

With modern distributed systems, we can add a second remote. This might be a personal *fork* on github:

```
import os

top_dir = os.getcwd()
git_dir = os.path.join(top_dir, "learning_git")
working_dir = os.path.join(git_dir, "git_example")
os.chdir(working_dir)
```

```
%%bash
git checkout main
git remote add jack89roberts git@github.com:jack89roberts/github-example.git
git fetch jack89roberts
```

```
error: pathspec 'main' did not match any file(s) known to git
From github.com:jack89roberts/github-example
 * [new branch]      main      -> jack89roberts/main
 * [new branch]      master     -> jack89roberts/master
```

Check your remote branches:

```
%%bash
git remote -v
```

```
jack89roberts  git@github.com:jack89roberts/github-example.git (fetch)
jack89roberts  git@github.com:jack89roberts/github-example.git (push)
origin  https://github.com/alan-turing-institute/rsd-engineeringcourse (fetch)
origin  https://github.com/alan-turing-institute/rsd-engineeringcourse (push)
```

and ensure that the newly-added remote is up-to-date

```
%%bash
git fetch jack89roberts
```

```
%%writefile Pennines.md

Mountains In the Pennines
=====
* Cross Fell
* Whernside
```

```
Writing Pennines.md
```

```
%%bash
git add Pennines.md
git commit -am "Add Whernside"
```

```
On branch experiment
nothing to commit, working tree clean
```

```
The following paths are ignored by one of your .gitignore files:
module04_version_control_with_git/learning_git
hint: Use -f if you really want to add them.
hint: Turn this message off by running
hint: "git config advice.addIgnoredFile false"
```

```
-----  
CalledProcessError                                     Traceback (most recent call last)
/tmp/ipykernel_3529/1299943852.py in <module>
--> 1 get_ipython().run_cell_magic('bash', '', 'git add Pennines.md\ngit commit -am
"Add Whernside"\n')

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/interactiveshell.py in run_cell_magic(self, magic_name, line,
cell)
    2404         with self.builtin_trap:
    2405             args = (magic_arg_s, cell)
-> 2406             result = fn(*args, **kwargs)
    2407             return result
    2408

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/magics/script.py in named_script_magic(line, cell)
    140     else:
    141         line = script
--> 142         return self.shebang(line, cell)
    143
    144     # write a basic docstring:

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/decorator.py in
fun(*args, **kw)
    230         if not kwsyntax:
    231             args, kw = fix(args, kw, sig)
--> 232         return caller(func, *(extras + args), **kw)
    233     fun.__name__ = func.__name__
    234     fun.__doc__ = func.__doc__

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/magic.py in <Lambda>(f, *, **k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/magics/script.py in shebang(self, line, cell)
    243             sys.stderr.flush()
    244             if args.raise_error and p.returncode!=0:
--> 245                 raise CalledProcessError(p.returncode, cell, output=out,
stderr=err)
    246
    247     def _run_script(self, p, cell, to_close):
```

```
CalledProcessError: Command 'b'git add Pennines.md\ngit commit -am "Add Whernside"\n'
returned non-zero exit status 1.
```

We can specify which remote to push to by name:

```
%%bash
git push -uf jack89roberts main || echo "Push failed"
```

Push failed

ERROR: Permission to jack89roberts/github-example.git denied to turingdeveloper.  
fatal: Could not read from remote repository.

Please make sure you have the correct access rights  
and the repository exists.

... but note that you need to have the correct permissions to do so.

## Referencing remotes

You can always refer to commits on a remote like this:

```
%%bash
git fetch
git log --oneline --left-right jack89roberts/main...origin/main
```

```
> 12ee6ad Add github pages YAML frontmatter
> c8ba483 Add a makefile and ignore generated files
> 537950c Commit Aonach onto main branch
> 85b2797 Merge branch 'main' of github.com:alan-turing-institute/github-example
> bc04a83 Add Glyder
> ed30178 Add another Beacon
> ecc3206 Merge branch 'main' of github.com:alan-turing-institute/github-example
> 8cd684e Add a beacon
> b8fb6c5 Translating from the Welsh
> 34f8bbb Merge branch 'main' of github.com:alan-turing-institute/github-example
> 289afed Add Scotland
> 0c976b0 Add wales
> 82b4fa0 Add Helvellyn
> 6ff088f Include lakes in the scope
> d063119 Add lakeland
> c489aab Revert "Add a lie about a mountain"
> f79acc Change title
> 6e8a302 Add a lie about a mountain
> 238eaff First commit of discourse on UK topography
< 31ea056 Add Whernside
< 009f998 Add github pages YAML frontmatter
< 2f9bcc8 Add a makefile and ignore generated files
< ae539cc Merge branch 'experiment' into main
< 492fec5 Commit Aonach onto main branch
< felc71d Add Cadair Idris
< 338d4d6 Merge branch 'main' of https://github.com/alan-turing-institute/github-example into main
< 07c4fea Add Glyder
< c405c4d Add another Beacon
< f8f20a6 Merge branch 'main' of https://github.com/alan-turing-institute/github-example into main
< 1f69c3f Translating from the Welsh
< b2b4fa3 Add a beacon
< c1897d4 Merge branch 'main' of https://github.com/alan-turing-institute/github-example into main
< 0e96c25 Add wales
< 0de6b80 Add Scotland
< 959e142 Add Helvellyn
< 600ffel Include lakes in the scope
< c7454a7 Add lakeland
< 5342922 Revert "Add a lie about a mountain"
< f65fd0b Change title
< 8c467a3 Add a lie about a mountain
< 1f92929 First commit of discourse on UK topography
```

To see the differences between remotes, for example.

To see what files you have changed that aren't updated on a particular remote, for example:

```
%%bash
git diff --name-only origin/main
```

When you reference remotes like this, you're working with a cached copy of the last time you interacted with the remote. You can do `git fetch` to update local data with the remotes without actually pulling. You can also get useful information about whether tracking branches are ahead or behind the remote branches they track:

```
%%bash
git branch -vv
```

```
* main 7444bd0 [origin/main: ahead 1] Add Whernside
```

## Hosting Servers

### Hosting a local server

- Any repository can be a remote for pulls
- Can pull/push over shared folders or ssh
- Pushing to someone's working copy is dangerous
- Use `git init --bare` to make a copy for pushing
- You don't need to create a "server" as such, any 'bare' git repo will do.

```
bare_dir = os.path.join(git_dir, "bare_repo")
os.chdir(bare_dir)
```

```
%%bash
mkdir -p bare_repo
cd bare_repo
git init --bare --initial-branch=main
```

```
Initialized empty Git repository in /home/turingdev/research-software/rse-course/module04_version_control_with_git/learning_git/bare_repo/
```

```
os.chdir(working_dir)
```

```
%%bash
git remote add local_bare ../bare_repo
git push -u local_bare main
```

```
Branch 'main' set up to track remote branch 'main' from 'local_bare'.
```

```
To ../bare_repo
 * [new branch]      main -> main
```

Check your remote branches:

```
%%bash
git remote -v
```

```
jack89roberts  git@github.com:jack89roberts/github-example.git (fetch)
jack89roberts  git@github.com:jack89roberts/github-example.git (push)
local_bare     ../bare_repo (fetch)
local_bare     ../bare_repo (push)
origin        git@github.com:alan-turing-institute/github-example.git (fetch)
origin        git@github.com:alan-turing-institute/github-example.git (push)
```

You can now work with this local repository, just as with any other git server. If you have a colleague on a shared file system, you can use this approach to collaborate through that file system.

## Home-made SSH servers

Classroom exercise: Try creating a server for yourself using a machine you can SSH to:

```
ssh <mymachine>
mkdir mygitserver
cd mygitserver
git init --bare
exit
git remote add <somename> ssh://user@host/mygitserver
git push -u <somename> master
```

## SSH keys and GitHub

Classroom exercise: If you haven't already, you should set things up so that you don't have to keep typing in your password whenever you interact with GitHub via the command line.

You can do this with an "ssh keypair". You may have created a keypair in the Software Carpentry shell training. Go to the [ssh settings page](#) on GitHub and upload your public key by copying the content from your computer. (Probably at .ssh/id\_rsa.pub)

If you have difficulties, the instructions for this are [on the GitHub website](#).

## Rebasing

### Rebase vs merge

A git *merge* is only one of two ways to get someone else's work into yours. The other is called a rebase.

In a merge, a revision is added, which brings the branches together. Both histories are retained. In a rebase, git tries to work out

What would you need to have done, to make your changes, if your colleague had already made theirs?

Git will invent some new revisions, and the result will be a repository with an apparently linear history. This can be useful if you want a cleaner, non-branching history, but it has the risk of creating inconsistencies, since you are, in a way, "rewriting" history.

### An example rebase

We've built a repository to help visualise the difference between a merge and a rebase, at

[https://github.com/UCL-RITS/wocky\\_rebase/blob/master/wocky.md](https://github.com/UCL-RITS/wocky_rebase/blob/master/wocky.md).

The initial state of both collaborators is a text file, [wocky.md](#):

```
It was clear and cold,  
and the slimy monsters
```

On the master branch, a second commit ('Dancing') has been added:

```
It was clear and cold,  
and the slimy monsters  
danced and spun in the waves
```

On the "Carollian" branch, a commit has been added translating the initial state into Lewis Caroll's language:

```
'Twas brillig,  
and the slithy toves
```

So the logs look like this:

```
git log --oneline --graph master
```

```
* 2a74d89 Dancing
* 6a4834d Initial state
```

```
git log --oneline --graph carollian
```

```
* 2232bf3 Translate into Caroll's language
* 6a4834d Initial state
```

If we now **merge** carollian into master, the final state will include both changes:

```
'Twas brillig,
and the slithy toves
danced and spun in the waves
```

But the graph shows a divergence and then a convergence:

```
git log --oneline --graph
```

```
* b41f869 Merge branch 'carollian' into master_merge_carollian
|\ 
| * 2232bf3 Translate into Caroll's language
* | 2a74d89 Dancing
|/
* 6a4834d Initial state
```

But if we **rebase**, the final content of the file is still the same, but the graph is different:

```
git log --oneline --graph master_rebase_carollian
```

```
* df618e0 Dancing
* 2232bf3 Translate into Caroll's language
* 6a4834d Initial state
```

We have essentially created a new history, in which our changes come after the ones in the carollian branch.

Note that, in this case, the hash for our "Dancing" commit has changed (from **2a74d89** to **df618e0**)!

To trigger the rebase, we did:

```
git checkout master
git rebase carollian
```

If this had been a remote, we would merge it with:

```
git pull --rebase
```

## Fast Forwards

If we want to continue with the translation, and now want to merge the rebased branch into the carollian branch, we get:

```
git checkout carollian
git merge master
```

```
Updating 2232bf3..df618e0
Fast-forward
 wacky.md | 1 +
 1 file changed, 1 insertion(+)
```

The master branch was already **rebased on** the carolian branch, so this merge was just a question of updating *metadata* (moving the label for the carolian branch so that it points to the same commit master does): a “fast forward”.

## Rebasing pros and cons

Some people like the clean, apparently linear history that rebase provides.

But *rebase rewrites history*.

If you've already pushed, or anyone else has got your changes, things will get screwed up.

If you know your changes are still secret, it might be better to rebase to keep the history clean. If in doubt, just merge.

## Squashing

A second way to use the `git rebase` command is to rebase your work on top of one of *your own* earlier commits, in interactive mode ([-i](#)). A common use of this is to “squash” several commits that should really be one, i.e. combine them into a single commit that contains all their changes:

```
git log
```

```
ea15 Some good work
ll54 Fix another typo
de73 Fix a typo
ab11 A great piece of work
cd27 Initial commit
```

## Using rebase to squash

If we type

```
git rebase -i ab11 #OR HEAD^~
```

an edit window pops up with:

```
pick cd27 Initial commit
pick ab11 A great piece of work
pick de73 Fix a typo
pick ll54 Fix another typo
pick ea15 Some good work

# Rebase 60709da..30e0ccb onto 60709da
#
# Commands:
#   p, pick = use commit
#   e, edit = use commit, but stop for amending
#   s, squash = use commit, but meld into previous commit
```

We can rewrite select commits to be merged, so that the history is neater before we push. This is a great idea if you have lots of trivial typo commits.

```
pick cd27 Initial commit
pick ab11 A great piece of work
squash de73 Fix a typo
squash ll54 Fix another typo
pick ea15 Some good work
```

save the interactive rebase config file, and rebase will build a new history:

```
git log
```

```
de82 Some good work
fc52 A great piece of work
cd27 Initial commit
```

Note the commit hash codes for 'Some good work' and 'A great piece of work' have changed, as the change they represent has changed.

## Debugging With Git Bisect

You can use

```
git bisect
```

to find out which commit caused a bug.

### An example repository

In a nice open source example, I found an arbitrary exemplar on github

```
import os

top_dir = os.getcwd()
git_dir = os.path.join(top_dir, "learning_git")
os.chdir(git_dir)
```

```
%%bash
rm -rf bisectdemo
git clone https://github.com/shawnsi/bisectdemo.git
```

```
Cloning into 'bisectdemo'...
```

```
bisect_dir = os.path.join(git_dir, "bisectdemo")
os.chdir(bisect_dir)
```

```
%%bash
python squares.py 2 # 4
```

```
4
```

This has been set up to break itself at a random commit, and leave you to use bisect to work out where it has broken:

```
%%bash
./breakme.sh > break_output
```

```
error: branch 'buggy' not found.
Switched to a new branch 'buggy'
```

Which will make a bunch of commits, of which one is broken, and leave you in the broken final state

```
python squares.py 2 # Error message
```

```
File "/tmp/ipykernel_3590/777131122.py", line 1
  python squares.py 2 # Error message
  ^
SyntaxError: invalid syntax
```

## Bisecting manually

```
%%bash
git bisect start
git bisect bad # We know the current state is broken
git checkout master
git bisect good # We know the master branch state is OK
```

```
Your branch is up to date with 'origin/master'.
Bisecting: 500 revisions left to test after this (roughly 9 steps)
[617d546cffec4d31775029924e8d5a2b3c5b198] Comment 499
```

```
Switched to branch 'master'
```

Bisect needs one known good and one known bad commit to get started

## Solving Manually

```
python squares.py 2 # 4
git bisect good
python squares.py 2 # 4
git bisect good
python squares.py 2 # 4
git bisect good
python squares.py 2 # Crash
git bisect bad
python squares.py 2 # Crash
git bisect bad
python squares.py 2 # Crash
git bisect bad
python squares.py 2 #Crash
git bisect bad
python squares.py 2 # 4
git bisect good
python squares.py 2 # 4
git bisect good
python squares.py 2 # 4
git bisect good
```

And eventually:

```
git bisect good
  Bisecting: 0 revisions left to test after this (roughly 0 steps)

python squares.py 2
  4

git bisect good
2777975a2334c2396ccb9faf98ab149824ec465b is the first bad commit
commit 2777975a2334c2396ccb9faf98ab149824ec465b
Author: Shawn Siefkas <shawn.siefkas@meredith.com>
Date:   Thu Nov 14 09:23:55 2013 -0600

  Breaking argument type
```

```
git bisect end
```

## Solving automatically

If we have an appropriate unit test, we can do all this automatically:

```
%%bash
git bisect start
git bisect bad HEAD # We know the current state is broken
git bisect good master # We know master is good
git bisect run python squares.py 2
```

```

Bisecting: 500 revisions left to test after this (roughly 9 steps)
[617d546cffec4d31775029924e8d5a2b3c5b198] Comment 499
running python squares.py 2
Bisecting: 249 revisions left to test after this (roughly 8 steps)
[c1c31d3c97c80a78a6a317734b9c653184f3af1b] Comment 249
running python squares.py 2
Bisecting: 124 revisions left to test after this (roughly 7 steps)
[b3f70d5b29ca4681f2d3475576402c3530eedbfd] Comment 124
running python squares.py 2
Bisecting: 62 revisions left to test after this (roughly 6 steps)
[39d50c04a8eddb853207058386d2bf8970625787] Comment 62
running python squares.py 2
4
Bisecting: 31 revisions left to test after this (roughly 5 steps)
[e481331452bb625c514fe2c21c9226fe156c4907] Comment 92
running python squares.py 2
Bisecting: 15 revisions left to test after this (roughly 4 steps)
[16308183d3f7972e963f5769a4df60c0869cc734] Comment 77
running python squares.py 2
4
Bisecting: 7 revisions left to test after this (roughly 3 steps)
[0ebf49eadc7b844929c1989f8001e12708992cd7] Comment 84
running python squares.py 2
Bisecting: 3 revisions left to test after this (roughly 2 steps)
[5c592e2e719cb94a515fb066c700da602d8c7d4c] Comment 81
running python squares.py 2
4
Bisecting: 1 revision left to test after this (roughly 1 step)
[05af67f4061b44906095ef422cd4ef0c605f8d1a] Comment 82
running python squares.py 2
Bisecting: 0 revisions left to test after this (roughly 0 steps)
[95644e2a318ce708f1ad9a337fad1eb817ad2fb5] Breaking argument type
running python squares.py 2
95644e2a318ce708f1ad9a337fad1eb817ad2fb5 is the first bad commit
commit 95644e2a318ce708f1ad9a337fad1eb817ad2fb5
Author: Shawn Siefkas <shawn.siefkas@meredith.com>
Date: Thu Nov 14 09:23:55 2013 -0600

    Breaking argument type

squares.py | 2 ++
1 file changed, 1 insertion(+), 1 deletion(-)
bisection run success

```

```

Previous HEAD position was 617d546 Comment 499
Switched to branch 'buggy'
Traceback (most recent call last):
  File "squares.py", line 9, in <module>
    print(integer**2)
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
Traceback (most recent call last):
  File "squares.py", line 9, in <module>
    print(integer**2)
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
Traceback (most recent call last):
  File "squares.py", line 9, in <module>
    print(integer**2)
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
Traceback (most recent call last):
  File "squares.py", line 9, in <module>
    print(integer**2)
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
Traceback (most recent call last):
  File "squares.py", line 9, in <module>
    print(integer**2)
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
Traceback (most recent call last):
  File "squares.py", line 9, in <module>
    print(integer**2)
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
Traceback (most recent call last):
  File "squares.py", line 9, in <module>
    print(integer**2)
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'

```

Boom!

## Assessment

We have provided the following exercises to let you self-assess your performance on some problems typical of those that are encountered in research software engineering.

- Packaging Greengraph
- Refactoring the Bad Boids

## An Adventure In Packaging: An exercise in research software engineering.

In this exercise, you will convert the already provided solution to the programming challenge defined in this Jupyter notebook, into a proper Python package.

The code to actually solve the problem is already given, but as roughly sketched out code in a notebook.

Your job will be to convert the code into a formally structured package, with unit tests, a command line interface, and demonstrating your ability to use `git` version control.

First, we set out the problem we are solving, and it's informal solution. Next, we specify in detail the target for your tidy solution. Finally, to assist you in creating a good solution, we state the marks scheme we will use.

## Treasure Hunting for Beginners: an AI testbed

We are going to look at a simple game, a modified version of one with a [long history](#). Games of this kind have been used as test-beds for development of artificial intelligence.

A *dungeon* is a network of connected *rooms*. One or more rooms contain *treasure*. Your character, the *adventurer*, moves between rooms, looking for the treasure. A *troll* is also in the dungeon. The troll moves between rooms at random. If the troll catches the adventurer, you lose. If you find treasure before being eaten, you win. (In this simple version, we do not consider the need to leave the dungeon.)

The starting rooms for the adventurer and troll are given in the definition of the dungeon.

The way the adventurer moves is called a *strategy*. Different strategies are more or less likely to succeed.

We will consider only one strategy this time - the adventurer will also move at random.

We want to calculate the probability that this strategy will be successful for a given dungeon.

We will use a "monte carlo" approach - simply executing the random strategy many times, and counting the proportion of times the adventurer wins.

Our data structure for a dungeon will be somewhat familiar from the Maze example:

```
dungeon1 = {
    "treasure": [1], # Room 1 contains treasure
    "adventurer": 0, # The adventurer starts in room 0
    "troll": 2, # The troll starts in room 2
    "network": [
        [1], # Room zero connects to room 1
        [0, 2], # Room one connects to rooms 0 and 2
        [1],
    ], # Room 2 connects to room 1
}
```

So this example shows a 3-room linear corridor: with the adventurer at one end, the troll at the other, and the treasure in the middle.

With the adventurer following a random walk strategy, we can define a function to update a dungeon:

```
import random

def random_move(network, current_loc):
    targets = network[current_loc]
    return random.choice(targets)
```

```
def update_dungeon(dungeon):
    dungeon["adventurer"] = random_move(dungeon["network"], dungeon["adventurer"])
    dungeon["troll"] = random_move(dungeon["network"], dungeon["troll"])
```

```
update_dungeon(dungeon1)
dungeon1
```

```
{'treasure': [1], 'adventurer': 1, 'troll': 1, 'network': [[1], [0, 2], [1]]}
```

We can also define a function to test if the adventurer has won, died, or if the game continues:

```
def outcome(dungeon):
    if dungeon["adventurer"] == dungeon["troll"]:
        return -1
    if dungeon["adventurer"] in dungeon["treasure"]:
        return 1
    return 0
```

```
outcome(dungeon1)
```

```
-1
```

So we can loop, to determine the outcome of an adventurer in a dungeon:

```
import copy

def run_to_result(dungeon):
    dungeon = copy.deepcopy(dungeon)
    max_steps = 1000
    for _ in range(max_steps):
        result = outcome(dungeon)
        if result != 0:
            return result
        update_dungeon(dungeon)
    # don't run forever, return 0 (e.g. if there is no treasure and the troll can't
    # reach the adventurer)
    return result
```

```
dungeon2 = {
    "treasure": [1], # Room 1 contains treasure
    "adventurer": 0, # The adventurer starts in room 0
    "troll": 2, # The troll starts in room 2
    "network": [
        [1], # Room zero connects to room 1
        [0, 2], # Room one connects to rooms 0 and 2
        [1, 3], # Room 2 connects to room 1 and 3
        [2],
    ], # Room 3 connects to room 2
}
```

```
run_to_result(dungeon2)
```

```
1
```

Note that we might get a different result sometimes, depending on how the adventurer moves, so we need to run multiple times to get our probability:

```

def success_chance(dungeon):
    trials = 10000
    successes = 0
    for _ in range(trials):
        outcome = run_to_result(dungeon)
        if outcome == 1:
            successes += 1
    success_fraction = successes / trials
    return success_fraction

```

```
success_chance(dungeon2)
```

0.4983

Make sure you understand why this number should be a half, given a large value for `trials`.

```

dungeon3 = {
    "treasure": [2], # Room 2 contains treasure
    "adventurer": 0, # The adventurer starts in room 0
    "troll": 4, # The troll starts in room 4
    "network": [
        [1], # Room zero connects to room 1
        [0, 2], # Room one connects to rooms 0 and 2
        [1, 3], # Room 2 connects to room 1 and 3
        [2, 4], # Room 3 connects to room 2 and 4
        [3],
    ], # Room 4 connects to room 3
}

```

```
success_chance(dungeon3)
```

0.4005

[Not for credit] Do you understand why this number should be 0.4? Hint: The first move is always the same. In the next state, a quarter of the time, you win. 3/8 of the time, you end up back where you were before. The rest of the time, you lose (eventually). You can sum the series:  $\left(\frac{1}{4}(1+\frac{3}{8}+(\frac{3}{8})^2+\dots)\right)=\frac{2}{5}$ .

## Packaging the Treasure: your exercise

If you are following the guidelines from earlier lectures, you will use a single top-level folder with a sensible name. This top level folder should contain all the parts of your solution.

Inside your top level folder, you should create a `setup.py` file to make the code installable. You should also create some other files, per the lectures, that should be present in all research software packages. (Hint, there are three of these.)

Your tidied-up version of the solution code should be in a sub-folder called `adventure` which will be the python package itself. It will contain an `__init__.py` file, and the code itself should be in a file called `dungeon.py`. This should define a class `Dungeon`: instead of a data structure and associated functions, you must refactor this into a class and methods.

Thus, if you run python in your top-level folder, you should be able to `from adventure.dungeon import Dungeon`.

You must create a command-line entry point, called `hunt`. This should use the `entry_points` facility in `setup.py`, to point toward a module designed for use as the entry point, in `adventure/command.py`. This should use the `Argparse` library. When invoked with `hunt mydungeon.yml --samples 500` the command must print on standard output the probability of finding the treasure in the specified dungeon, using the random walk strategy, after the specified number of test runs.

The `dungeon.yml` file should be a yml file containing a structure representing the dungeon state. Use the same structure as the sample code above, even though you'll be building a `Dungeon` object from this structure rather than using it directly.

You must create unit tests which cover a number of examples. These should be defined in `adventure/tests/test_dungeon.py`. Don't forget to add an `init.py` file to that folder too, so that at the top of the test file you can "`from ..dungeon import Dungeon`." If your unit tests use a fixture file to DRY up tests, this must be called `adventure/tests/fixtures.yml`. For example, this could contain a yaml array of many dungeon structures.

You should `git init` as soon as you create the top-level folder, and `git commit` your work regularly as the exercise progresses.

## Suggested Marking Scheme

If you want to self-assess your solution you can consider using the marking scheme below.

- Code in `dungeon.py`, implementing the random walk strategy: **(5 marks)**
  - Which works. **(1 mark)**
  - Cleanly laid out and formatted - PEP8. **(1 mark)**
  - Defining the class `Dungeon` with a valid object oriented structure. **(1 mark)**
  - Breaking down the solution sensibly into subunits. **(1 mark)**
  - Structured so that it could be used as a base for other strategies. **(1 mark)**
- Command line entry point: **(4 marks)**
  - Accepting a dungeon definition text file as input. **(1 mark)**
  - With an optional parameter to control sample size. **(1 mark)**
  - Which prints the result to standard out. **(1 mark)**
  - Which correctly uses the `Argparse` library. **(1 mark)**
  - Which is itself cleanly laid out and formatted. **(1 mark)**
- `setup.py` file: **(5 marks)**
  - Which could be used to `pip install` the project. **(1 mark)**
  - With appropriate metadata, including version number and author. **(1 mark)**
  - Which packages code (but not tests), correctly. **(1 mark)**
  - Which specifies library dependencies. **(1 mark)**
  - Which points to the entry point function. **(1 mark)**
- Three other metadata files: **(3 marks)**
  - Hint: Who did it, how to reference it, who can copy it.
- Unit tests: **(5 marks)**
  - Which test some obvious cases. **(1 mark)**
  - Which correctly handle approximate results within an appropriate tolerance. **(1 mark)**
  - Which test how the code fails when invoked incorrectly. **(1 mark)**
  - Which use a fixture file or other approach to avoid overly repetitive test code. **(1 mark)**
  - Which are themselves cleanly laid out code. **(1 mark)**
- Version control: **(2 marks)**
  - Sensible commit sizes. **(1 mark)**
  - Appropriate commit comments. **(1 mark)**

Total: **25 marks**

## Packaging Greengraph

### Summary

In an appendix, taken from [here](#), are classes which generate a graph of the proportion of green pixels in a series of satellite images between two points.

<https://alan-turing-institute.github.io/rsd-engineeringcourse/html/ch00python/index.html>

Your task is to take this code, and do the work needed to make it into a proper package which could be released, meeting minimum software engineering standards

- **package** this code, into a git repository, suitable to be installed with `pip`
- create an appropriate command line entry point, so that the code can be invoked with `greengraph --from London --to Oxford --steps 10 --out graph.png` or a similar interface
- Implement automated tests for each element of the code
- Add appropriate standard supplementary files to the code, describing license, citation and typical usage

## Assignment Presentation

For this coursework assignment, you are expected to submit a short report and your code. The purpose of the report is to answer the non-coding questions below, to present your results and provide a brief description of your design choices and implementation. The report need not be very long or overly detailed, but should provide a succinct record of your coursework. The report must have a cover sheet stating your name, your student number, and the code of the module (MPHYG001).

## Submission

TBD

You should submit your report and all of your source code so that an independent person can run the code. The code and report must be submitted as a single zip or tgz archive of a folder which contains **git** version control information for your project. Your report should be included as a PDF file, report.pdf, in the root folder of your archive. There is no need to include your source code in your report, but you can refer to it and if necessary reproduce lines if it helps to explain your solution.

## Marks Scheme

- Code broken up into appropriate files, and arranged into an appropriate folder structure [2 marks]
- Git version control used, with a series of sensible commit messages [2 marks]
- Command line entry point, using appropriate library to parse arguments [3 marks]
- Packaging for `pip` installation, with `setup.py` file with appropriate content [5 marks]
- Automated tests for each method and class (2 marks), with appropriate fixtures defined (1 mark) and mocks used to avoid tests interacting with internet (2 marks). [5 marks total]
- Supplementary files to define license, usage, and citation. [3 marks]
- A text report which:
  - Documents the usage of your entry point [1 mark]
  - Discusses problems encountered in completing your work [1 mark]
  - Discusses in your own words the advantages and costs involved in preparing work for release, the use of package managers like pip and package indexes like PyPI [2 marks]
  - Discusses further steps you would need to take to build a community of users for a project [1 mark]

[25 marks total]

## Appendix

```

import numpy as np
import geopy
from StringIO import StringIO
from matplotlib import image as img

class Greengraph:
    def __init__(self, start, end):
        self.start = start
        self.end = end
        self.geocoder = geopy.geocoders.Nominatim(user_agent="rsd-course")

    def geolocate(self, place):
        return self.geocoder.geocode(place, exactly_one=False)[0][1]

    def location_sequence(self, start, end, steps):
        lats = np.linspace(start[0], end[0], steps)
        longs = np.linspace(start[1], end[1], steps)
        return np.vstack([lats, longs]).transpose()

    def green_between(self, steps):
        return [
            Map(*location).count_green()
            for location in self.location_sequence(
                self.geolocate(self.start), self.geolocate(self.end), steps
            )
        ]
    ]

class Map:
    def __init__(
        self, lat, long, satellite=True, zoom=10, size=(400, 400), sensor=False
    ):
        base = "https://static-maps.yandex.ru/1.x/?"

        params = dict(
            z=zoom,
            size=str(size[0]) + "," + str(size[1]),
            ll=str(long) + "," + str(lat),
            l="sat" if satellite else "map",
            lang="en_US",
        )

        self.image = requests.get(
            base, params=params
        ).content # Fetch our PNG image data
        content = BytesIO(self.image)
        self.pixels = img.imread(content) # Parse our PNG image as a numpy array

    def green(self, threshold):
        # Use NumPy to build an element-by-element logical array
        greener_than_red = self.pixels[:, :, 1] > threshold * self.pixels[:, :, 0]
        greener_than_blue = self.pixels[:, :, 1] > threshold * self.pixels[:, :, 2]
        green = np.logical_and(greener_than_red, greener_than_blue)
        return green

    def count_green(self, threshold=1.1):
        return np.sum(self.green(threshold))

    def show_green(self, threshold=1.1):
        green = self.green(threshold)
        out = green[:, :, np.newaxis] * array([0, 1, 0])[np.newaxis, np.newaxis, :]
        buffer = BytesIO()
        result = img.imwrite(buffer, out, format="png")
        return buffer.getvalue()

mygraph=Greengraph('New York', 'Chicago')
data = mygraph.green_between(20)
plt.plot(data)

```

## Refactoring Trees: An exercise in Research Software Engineering

In this exercise, you will convert badly written code, provided here, into better-written code.

You will do this not through simply writing better code, but by taking a refactoring approach, as discussed in the lectures.

As such, your use of `git` version control, to make a commit after each step of the refactoring, with a commit message which indicates the refactoring you took, will be critical to success.

You will also be asked to look at the performance of your code, and to make changes which improve the speed of the code.

The script as supplied has its parameters hand-coded within the code. You will be expected, in your refactoring, to make these available as command line parameters to be supplied when the code is invoked.

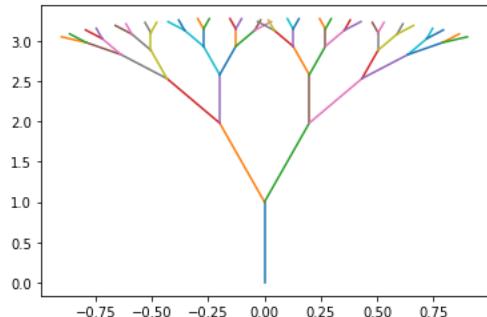
## Some terrible code

Here's our terrible code:

```
%matplotlib inline

from math import sin, cos
from matplotlib import pyplot as plt

s = 1
d = [[0, 1, 0]]
plt.plot([0, 0], [0, 1])
for i in range(5):
    n = []
    for j in range(len(d)):
        n.append(
            [
                d[j][0] + s * sin(d[j][2] - 0.2),
                d[j][1] + s * cos(d[j][2] - 0.2),
                d[j][2] - 0.2,
            ]
        )
    n.append(
        [
            d[j][0] + s * sin(d[j][2] + 0.2),
            d[j][1] + s * cos(d[j][2] + 0.2),
            d[j][2] + 0.2,
        ]
    )
    plt.plot([d[j][0], n[-2][0]], [d[j][1], n[-2][1]])
    plt.plot([d[j][0], n[-1][0]], [d[j][1], n[-1][1]])
    d = n
    s *= 0.6
plt.savefig("tree.png")
```



## Suggested Marking Scheme

If you want to self-assess your solution you can consider using the marking scheme below.

### Part one: Refactoring (15 marks)

- Copy the code above into a file [tree.py](#), invoke it with python [tree.py](#), and verify it creates an image tree.png which looks like that above.
- Initialise your git repository with the raw state of the code. **(1 mark)**
- Identify a number of simple refactorings which can be used to improve the code, *reducing repetition* and *improving readability*. Implement these one by one, with a git commit each time.
  - **1 mark** for each refactoring, **1 mark** for each git commit, at least five such: **10 marks total**.
- Do NOT introduce NumPy or other performance improvements yet (see below).

- Identify which variables in the code would, more sensibly, be able to be input parameters, and use Argparse to manage these.
  - **4 marks:** 1 for each of four arguments identified.

## Part two: performance programming (10 marks)

- For the code as refactored, prepare a figure which plots the time to produce the tree, versus number of iteration steps completed. Your code to produce this figure should run as a script, which you should call `perf_plot.py`, invoking a function imported from `tree.py`. The script should produce a figure called `perf_plot.png`. Comment on your findings in a text file, called `comments.md`. For your performance measurements you should turn off the actual plotting, and run only the mathematical calculation using an appropriate flag. **5 marks:**
  - Time to run code identified. **(1 mark)**
  - Figure created. **(1 mark)**
  - Figure correctly formatted. **(1 mark)**
  - Figure auto-generated from script. **(1 mark)**
  - Performance law identified. **(1 mark)**
- The code above makes use of `append()` which is not appropriate for NumPy. Create a new solution (in a file called `tree_np.py`) which makes use of NumPy. Compare the performance (again, excluding the plotting from your measurements), and discuss in `comments.md`. **5 marks:**
  - Array-operations used to subtract the change angle from all angles in a single minus sign. **(1 mark)**
  - Array-operations used to take the sine of all angles using `np.sin`. **(1 mark)**
  - Array-operations used to move on all the positions with a single vector displacement addition. **(1 mark)**
  - Numpy solution uses `hstack` or similar to create new arrays with twice the length, by composing the left-turned array with the right-turned array. **(1 mark)**
  - Performance comparison recorded. **(1 mark)**

## Refactoring the Bad Boids

### Summary

In an appendix, taken from [here](#), is a very poor implementation of the Boids flocking example from the course.

Your task is to take this code, and build from it a clean implementation of the flocking code, finishing with an appropriate object oriented design, using the **refactoring** approach to gradually and safely build your better solution from the supplied poor solution. Simply submitting a good, clean solution will not complete this assignment; you are being assessed on the step-by-step refactoring process, on the basis of individual git commits. You should develop unit tests for your code as units (functions, classes, methods and modules) emerge, and wrap the code in an appropriate command line entry point providing an appropriate configuration system for simulation setup.

### Assignment Presentation

For this coursework assignment, you are expected to submit a short report and your code. The purpose of the report is to answer the non-coding questions below, to present your results and provide a brief description of your design choices and implementation. The report need not be very long or overly detailed, but should provide a succinct record of your coursework.

### Submission

You should submit your report and all of your source code so that an independent person can run the code. The code and report must be submitted as a single `zip` or `tgz` archive of a folder which contains `git` version control information for your project. Your report should be included as a PDF file, `report.pdf`, in the root folder of your archive. There is no need to include your source code in your report, but you can refer to it and if necessary reproduce lines if it helps to explain your solution.

## Marking Scheme

- Final state of code is well broken down into functions, classes, methods and modules [3 marks]
- Final state of code is readable with good variable, function, class and method names and necessary comments [2 marks]
- Git version control used, with a series of sensible commit messages [2 marks]
- Command line entry point and configuration file, using appropriate libraries [3 marks]
- Packaging for pip installation, with setup.py file with appropriate content [2 marks]
- Automated tests for each method and class. [5 marks]
- Supplementary files to define license, usage, and citation. [3 marks]
- A text report which:
  - Lists by name the code smells identified refactorings used, making reference to the git commit log [2 marks]
  - Includes a UML diagram of the final class structure [1 mark]
  - Discusses in your own words the advantages of a refactoring approach to improving code [1 mark]
  - Discusses problems encountered during the project [1 mark]

[25 marks total]

## Appendix

```
from matplotlib import pyplot as plt
from matplotlib import animation
import random

# Deliberately terrible code for teaching purposes

boids_x=[random.uniform(-450,50.0) for x in range(50)]
boids_y=[random.uniform(300.0,600.0) for x in range(50)]
boid_x_velocities=[random.uniform(0,10.0) for x in range(50)]
boid_y_velocities=[random.uniform(-20.0,20.0) for x in range(50)]
boids=(boids_x,boids_y,boid_x_velocities,boid_y_velocities)

def update_boids(boids):
    xs,ys,xvs,yvs=boids
    # Fly towards the middle
    for i in range(50):
        for j in range(50):
            xvs[i]=xvs[i]+(xs[j]-xs[i])*0.01/len(xs)
    for i in range(50):
        for j in range(50):
            yvs[i]=yvs[i]+(ys[j]-ys[i])*0.01/len(xs)
    # Fly away from nearby boids
    for i in range(50):
        for j in range(50):
            if (xs[j]-xs[i])**2 + (ys[j]-ys[i])**2 < 100:
                xvs[i]=xvs[i]+(xs[i]-xs[j])
                yvs[i]=yvs[i]+(ys[i]-ys[j])
    # Try to match speed with nearby boids
    for i in range(50):
        for j in range(50):
            if (xs[j]-xs[i])**2 + (ys[j]-ys[i])**2 < 10000:
                xvs[i]=xvs[i]+(xvs[j]-xvs[i])*0.125/len(xs)
                yvs[i]=yvs[i]+(yvs[j]-yvs[i])*0.125/len(xs)
    # Move according to velocities
    for i in range(50):
        xs[i]=xs[i]+xvs[i]
        ys[i]=ys[i]+yvs[i]

figure=plt.figure()
axes=plt.axes(xlim=(-500,1500), ylim=(-500,1500))
scatter=axes.scatter(boids[0],boids[1])

def animate(frame):
    update_boids(boids)
    scatter.set_offsets(zip(boids[0],boids[1]))

anim = animation.FuncAnimation(figure, animate,
                               frames=50, interval=50)

if __name__ == "__main__":
    plt.show()
```

