First Steps:
Install - Git - Python

2018-10-03

Scientific Programming with Python

Install

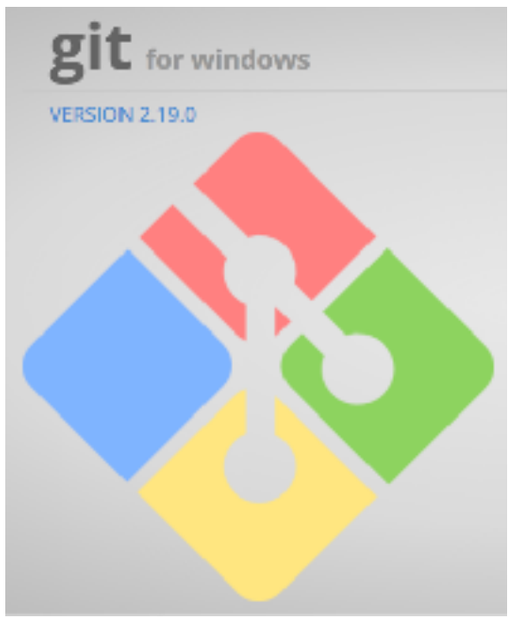
Install

1) https://www.anaconda.com/download/

ANACONDA. **Python 3.7 (64-bit)**

- **Python distribution** for scientists and data analysts
- **Packages** for data analysis and visualisation
  (numpy, scipy, matplotlib, hdf5, pandas, and many more)
- **Jupyter Lab**
- **Package management** system for installing more

2) https://gitforwindows.org/



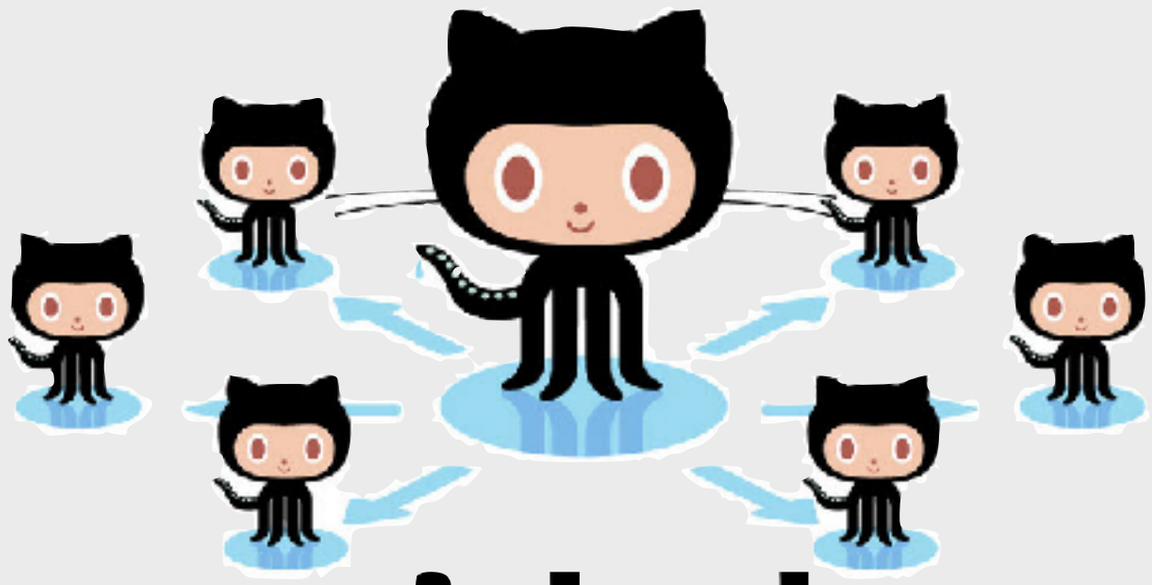
- **Version control** system
- Shell based: ***Git BASH***

Version Control With Git

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

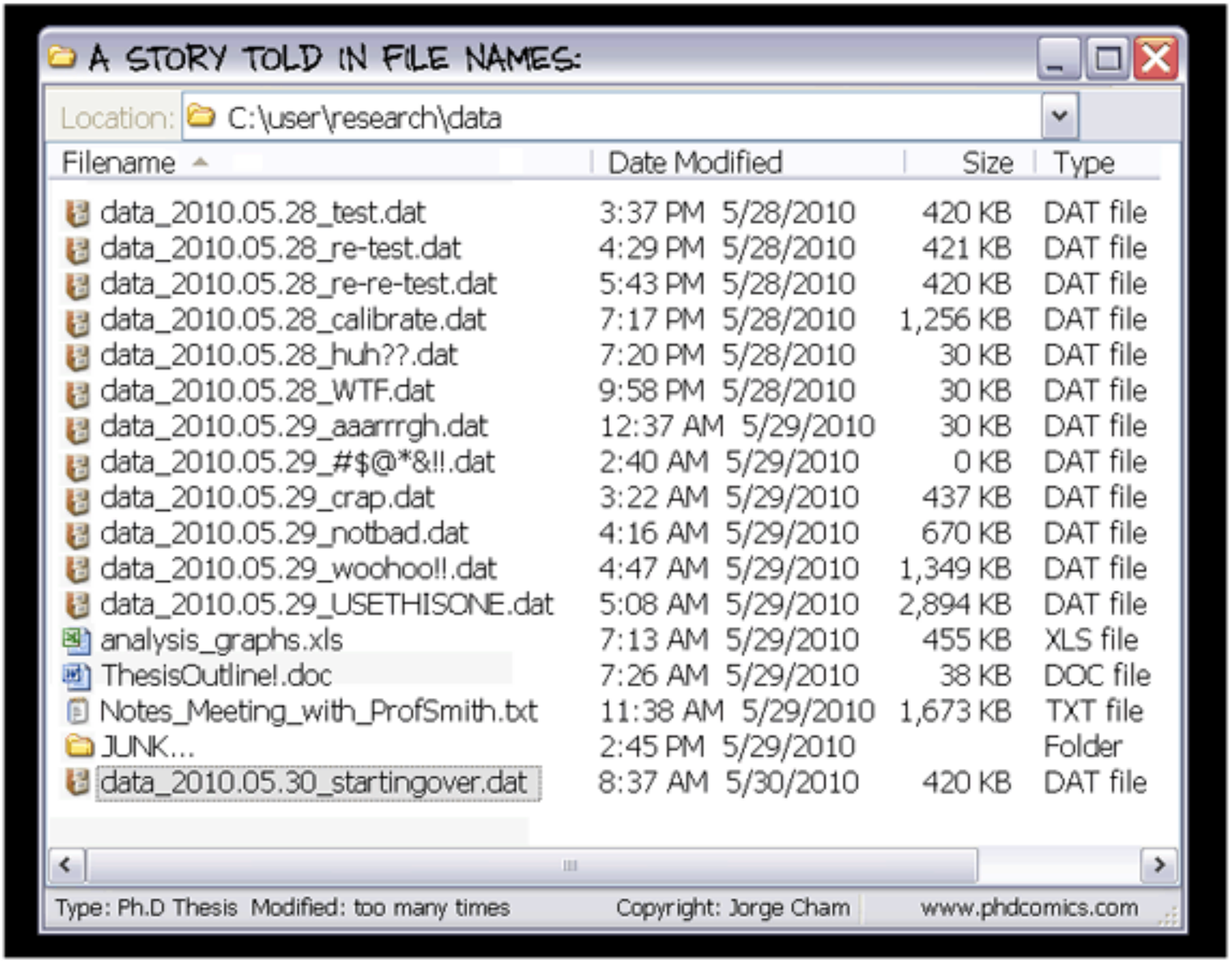
COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.



github
SOCIAL CODING

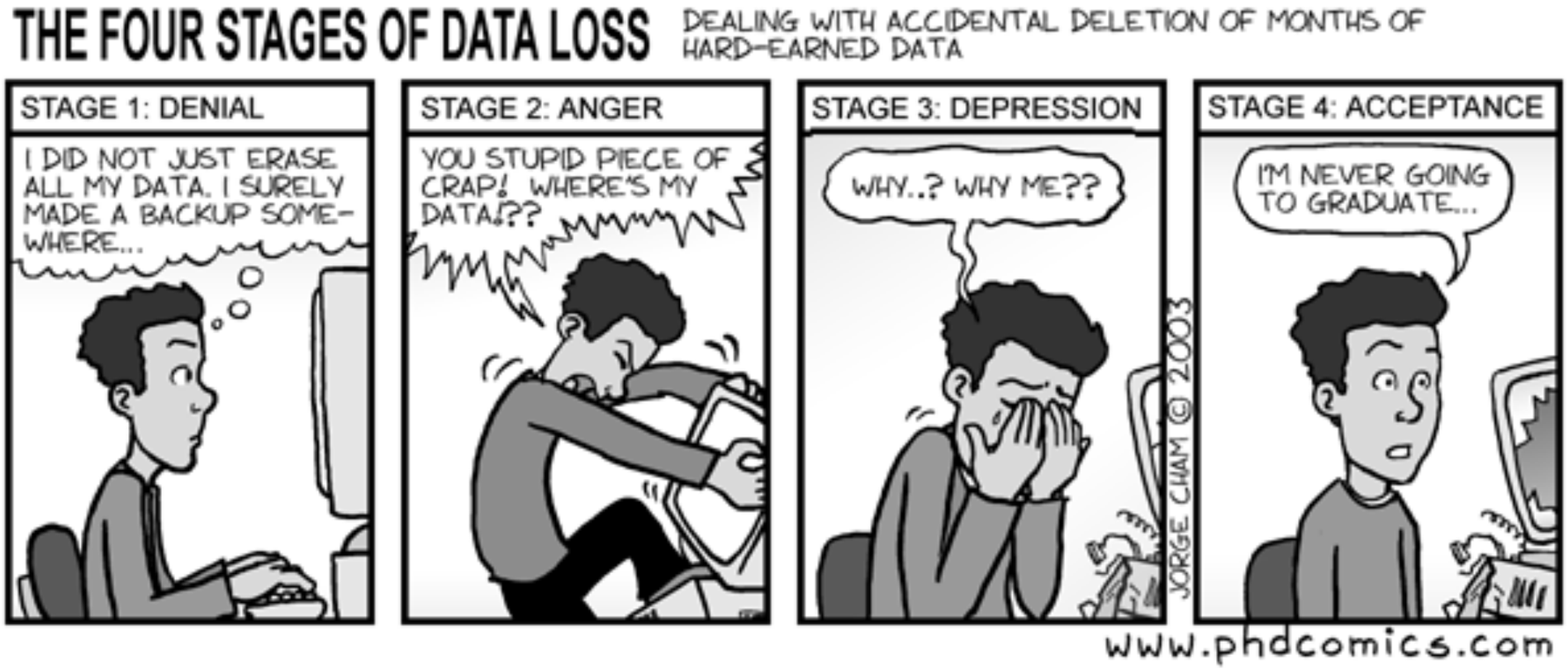
Why Do I Need Version Control?



by Jorge Cham
www.phdcomics.com

Why Do I Need Version Control?

- Your files are better organised
- You keep a history of all previous versions
- Your research is faster, more efficient and more reproducible
- Version control benefits collaborative work
- You always have a backup



How Do I Use Git?

Creating a new project

$ git init

Cloning an existing project

$ git clone https://github.com/…/project.git

Adding new files to be committed

$ git add README.md

Commit all new files

$ git commit -m “Useful message”

Updating the local copy (“pulling”)

$ git pull

Updating the remote (“pushing”)

$ git push

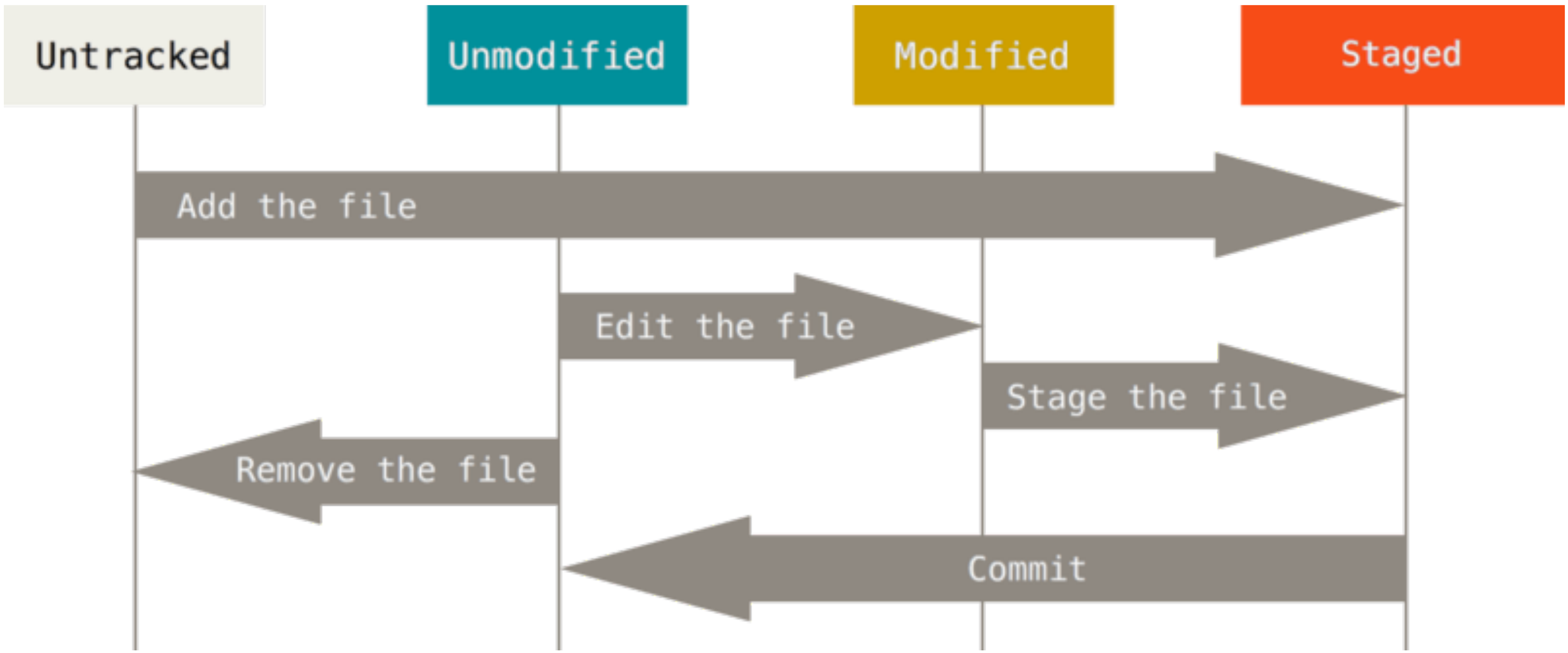
Current status of all files of repository

$ git status

Show the history (commit log)

$ git log

The Lifecycle Of The Status Of Your Files



Pro Git Boot, by Scott Chacon: http://git-scm.com/book

Setting Up Git

- Local configurations (only the current repository is affected)
  $ git config [options]
- Global configurations (only the user's configuration is modified)
  $ git config --global [options]
- System configurations (all users are affected)
  $ git config --system [options]
- Change your identity
  $ git config —-global user.name "Max Hantke"
  $ git config —-global user.email "max.hantke@gmail.com"
- Set your favourite editor (e.g. emacs or vim)
  $ git config —-global core.editor emacs
- Check your current settings
  $ git config —-list

Deleting, Moving, Cancelling, Resetting

- Deleting a tracked file
  $ git rm FILE
- Deleting a tracked file (but keeping an untracked copy)
  $ git rm --cached FILE
- Moving a file (renaming)
  $ git mv FILE TARGET
- Unstaging a file
  $ git reset HEAD FILE
- Undo modifications of unstaged files
  $ git checkout -- FILE1 FILE2
- Checkout a previous version
  $ git checkout HASH

Branching And Merging

Create a new branch "testing"

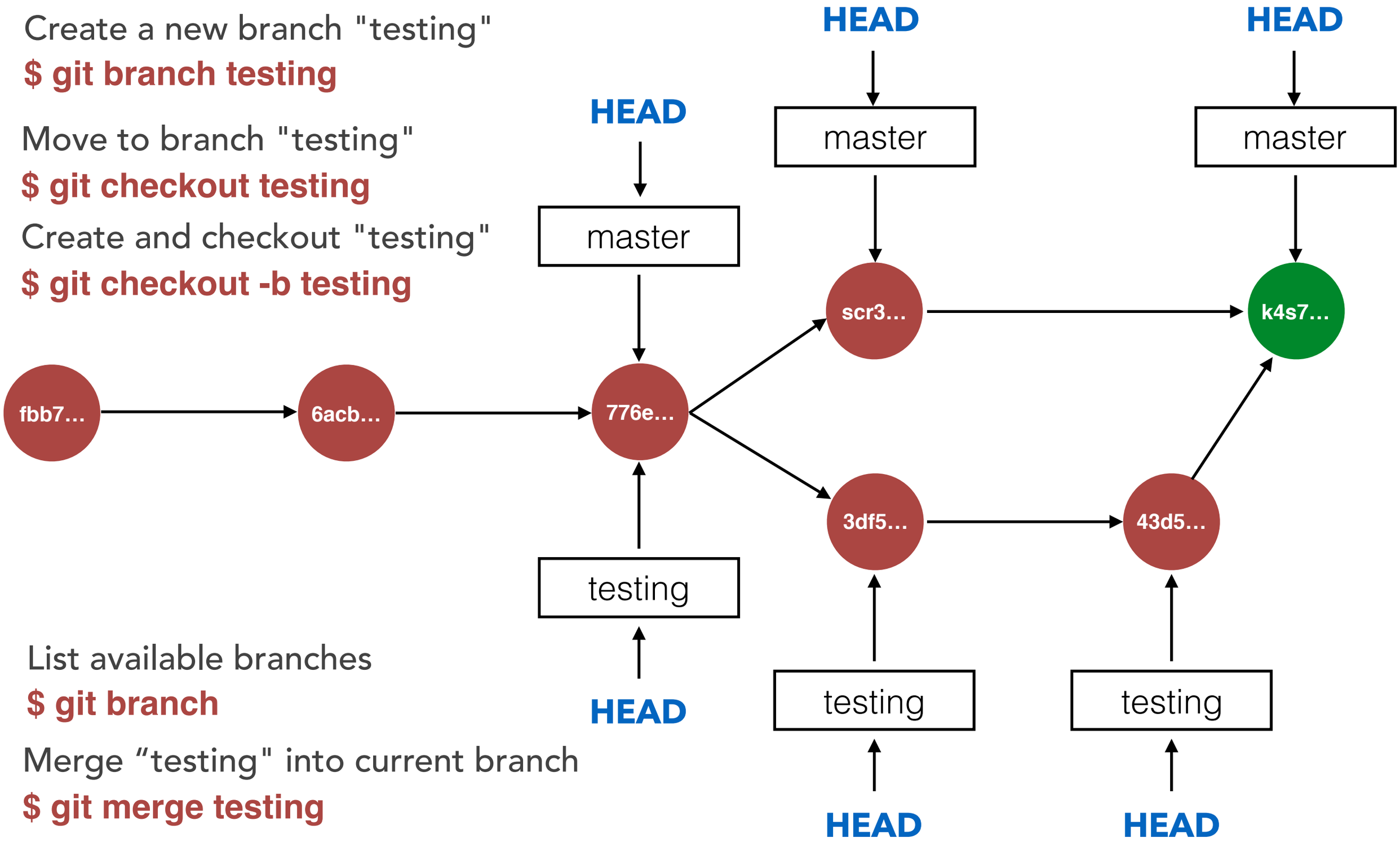
$ git branch testing

Move to branch "testing"

$ git checkout testing

Create and checkout "testing"

$ git checkout -b testing



List available branches

$ git branch

Merge “testing" into current branch

$ git merge testing

The Unix Shell

- Provides a command line interface (CLI) to the operating system
- Large variety of shells: **bash, tcsh, csh, ksh, zsh**
- Documentation can be found by typing **man <command>**, e.g. **man bash**.
- Popular command examples:
  cd: Change directory
  ls: List
  pwd: Print work directory
  git: …

A shell on Mac OS X, a Unix system

Python

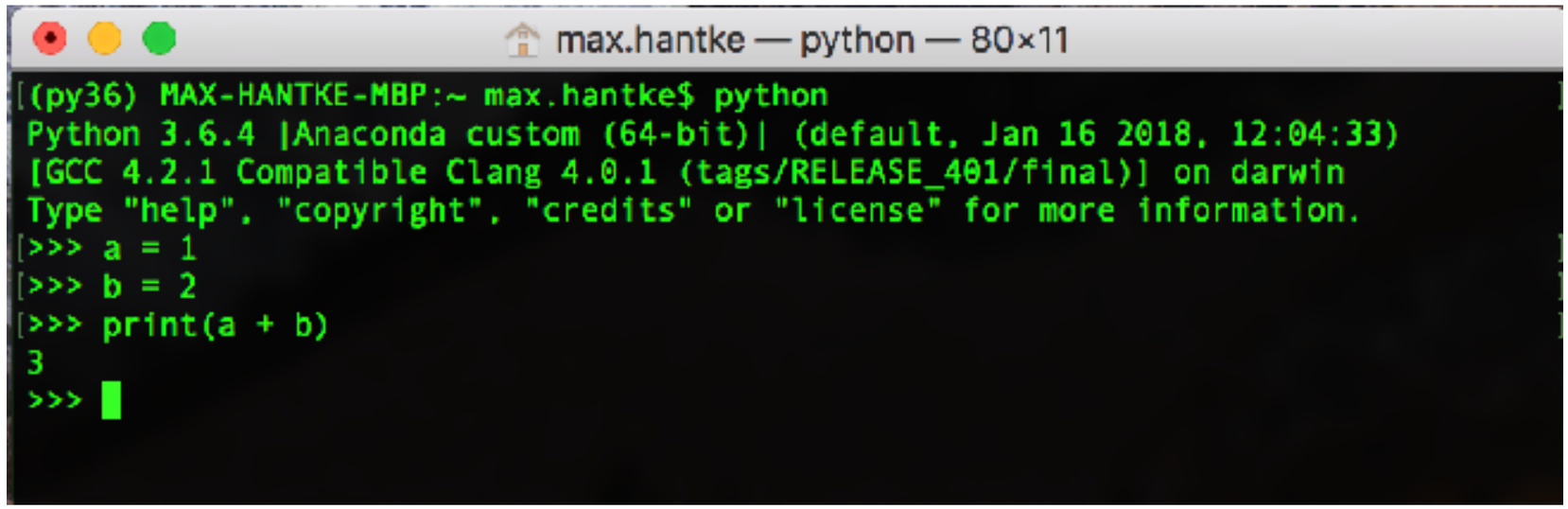
Why Python

- **Target**:
  - Language developed for scientists
  - By now general-purpose language
- **Difficulty**:
  - Good code readability

    (concept of “significant whitespaces”)
  - Well documented
  - Script language
- **Toolbox**: Huge amount of useful tools
- **Performance**:
  - Can be easily interfaced with code written in other languages
  - High-performance computing tools

Standard Python Interpreter

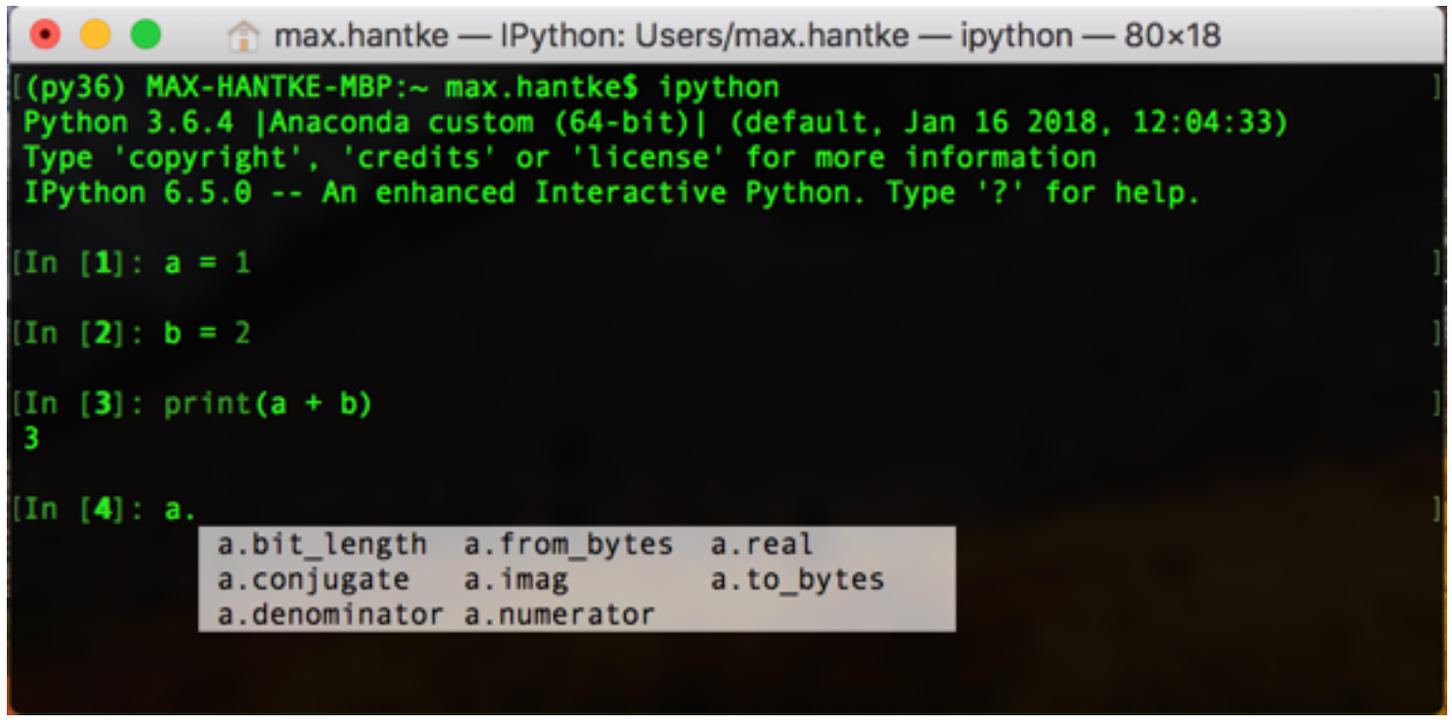
- The standard Python interpreter is **python**.
- For example, to run a script **my-program.py**:
  $ python my-program.py
- We can also start the interpreter by simply typing python at the command line, and interactively type Python code into the interpreter.



- The standard Python interpreter is not very convenient due to a number of limitations.

IPython

- IPython addresses the limitation of the standard python interpreter
- A work-horse for scientific use of python. It provides an interactive prompt to the python interpreter with a greatly improved user-friendliness.



- It includes:
  Command history, using the up and down arrows.
  Tab auto-completion.
  In-line editing of code.
  Object introspection, and docstring extraction.
  Good interaction with operating system shell.

Jupyter Lab

- Open-source web application
- Allows you to create and share documents that contain live code, equations, visualisations and explanatory text
- Based on the IPython shell, but provides a cell-based environment with great interactivity
- Similar interface capabilities to Matlab.
- Calculations can be organised and documented in a structured way.
- Jupyter Lab notebooks are usually run locally, from the same computer that run the browser.
- To start a new Jupiter notebook session, start the Anaconda application and click Jupiter Lab

Now:

1. Install Anaconda and