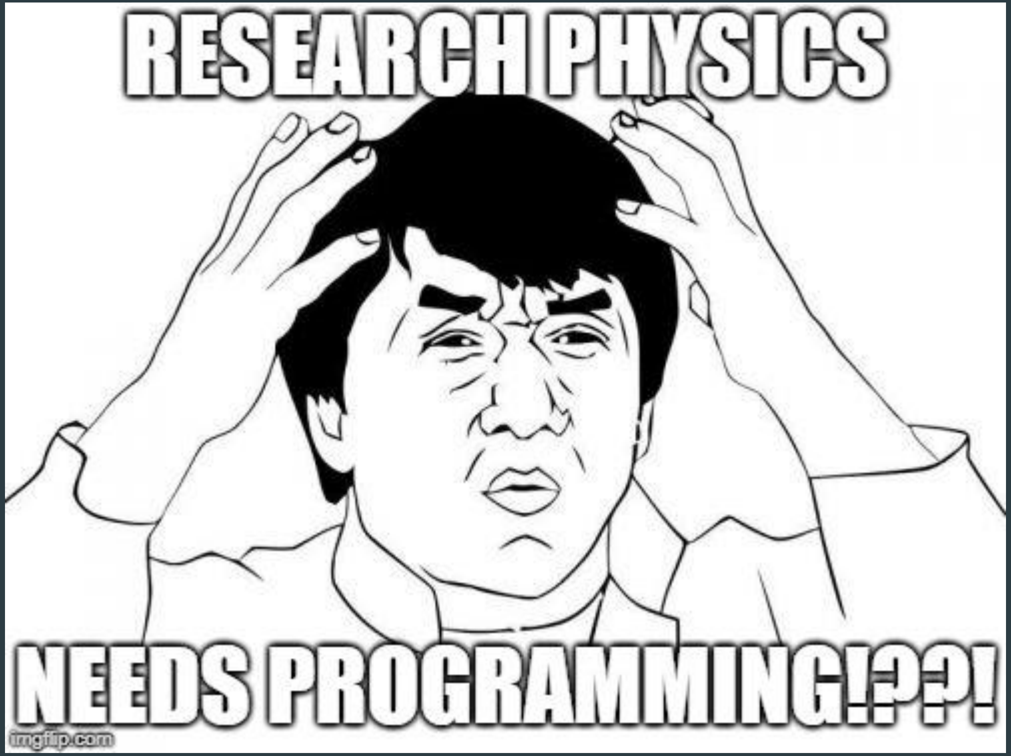
Technical Meeting: Software Development

The Zen of Python

Jupyter Notebooks

Responsibility should be unique

Programming is for gits



1. The Zen of Python (goo.gl/KubWVd)

- 聽 Beautiful is better than ugly
- 聽 Simple is better than Complex
- 聽 Flat is better than nested
- 聽 Readability counts

More important: PEP8

- Python’s standard library style conventions
  - Most, if not all, professional Pythons use it in their own code
- Key insight: Code is *read* more often than it is *written*
  - From experience: You can’t even read your own code down the road!
  - Now imagine inheriting code from and old PostDoc from Russia 5 years ago…
- Developers communicate (in code): use the same language (code style)!
- If others use your code – they need to be able to understand it!
  - For instance, to report bugs. Which we don’t make. Even so.

Jupyter Notebooks

- More ‘notebook’ style working
- Good for experimentation
- Good for sharing (e.g. discussion about Z)
- Can be rendered on GitHub (etc)

- Downside:
  - Usually: Unordered
  - Usually: Unclear persistence of variables
  - Obviously: You won’t make such mistakes

The Physicist Trap

- Physicists define their symbols in math: e.g. r, x, g_munu
- Trap: Call your variables the same
  - Confusion – others do not have your notes!
  - Conflicts! E.g. lambda is a defined keyword (anonymous function)
  - Common variables change!
- Instead, describe. Autocompletion is your friend!
  - x: spatial_distance
  - R: conformal_dimension
  - G_munu: gravitational_tensor

2. Responsibility should be unique

- A generic coding principle.
  - “A class should have only one reason to change”
  - “Gather together the things that change for the same reasons. Separate those things that change for different reasons”.
  - See e.g. doi 10.1145/361598.361623
- We’ve already thought of this, in many ways:
  - Flowchart separates e.g. importing mathematica from optimizing the matrix
- Continue to use it ☺
  - E.g. a calculation optimization rarely has to know of the ‘physical objects’ it optimizes.

3. Programming is for Gits

- But Josko, I don’t want to have so many files.
- But, I won’t be able to find the function in the right file
- But, what if we change things?
- What if we all change things?
- Wait, what did you change last week on Friday 15:21 when I wasn’t paying attention?
- Did you do that?
- Wait, what do we still have to do?
- Aaah!

Version Control Software

- Think of “save points” or “check points” in games. You’re able to go back to a previous point and continue.
- Changes are recorded, and can be undone.
- By keeping changes recorded, multiple changes by many people can be combined automatically.

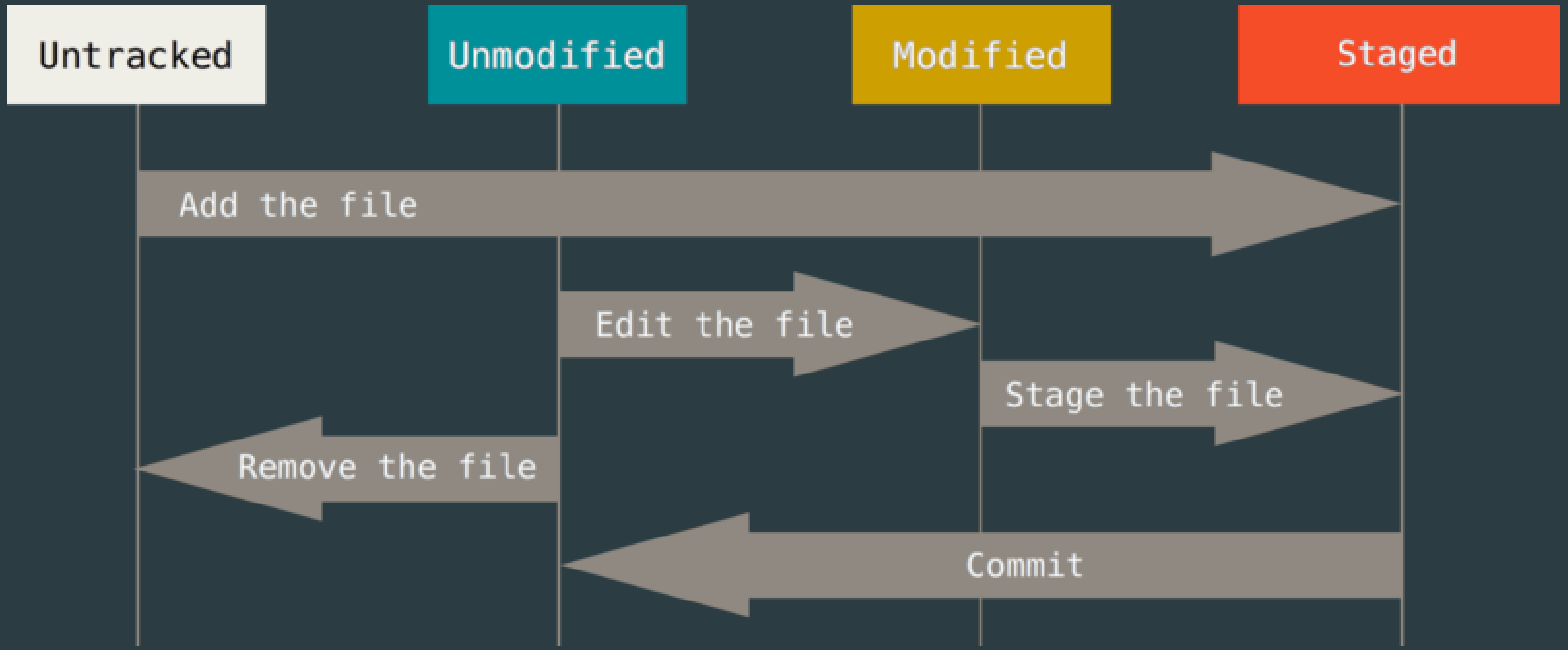
On the history of Git

- Long Ago, before I went to university, the Linux kernel used a proprietary software package for its development.
- Then, the management changed and the relationship between the Linux developer community and the commercial company broke. As I heard it, the new management went “haha you have to pay now” and Linux went “Fuck that”.
- Anyway, they made Git and it is now used by about 90-100% of developers.
- Its goals:
  - Speed
  - Simple Design
  - Strong support for non-linear development
  - Fully distributed
  - Able to handle large projects

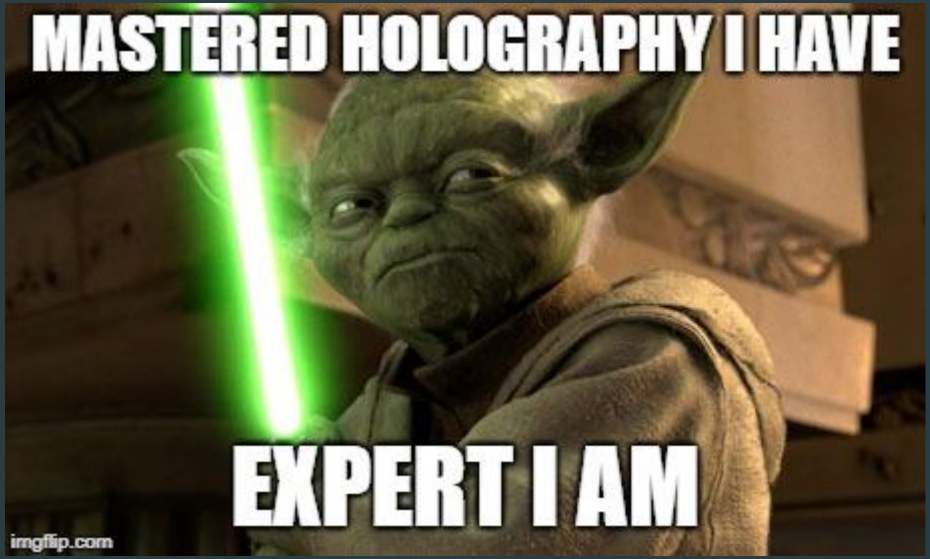
Using Git

- It’s a command line program.
- Or use a graphical user interface, like GitHub provides.
- Or don’t. Command lines are better anyway.
  - Seriously, the command line offers more functionality than the more limited user interfaces do. That’s life.

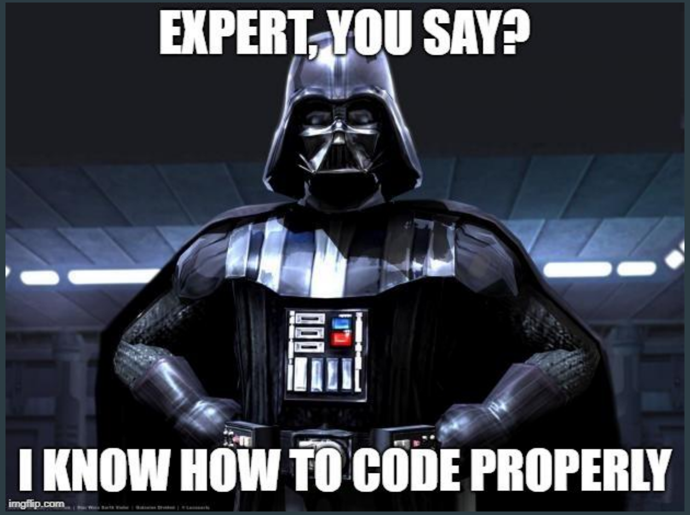
Git Basics: Recording Changes



Conclusion



- After mastery of
  - The Zen of Pythons
  - Singular Responsibility
  - Being a Git
- You’re able to become a Jedi Master. Alternatively, to make “good” code.



🡐 This was basically how I got a PhD spot