# Software Engineering for Data Scientists

For PRIVATE USE ONLY, not for open publication

Training 2022

Ian Ozsvald

@IanOzsvald – ianozsvald.com

# Introductions

PLURALSIGHT

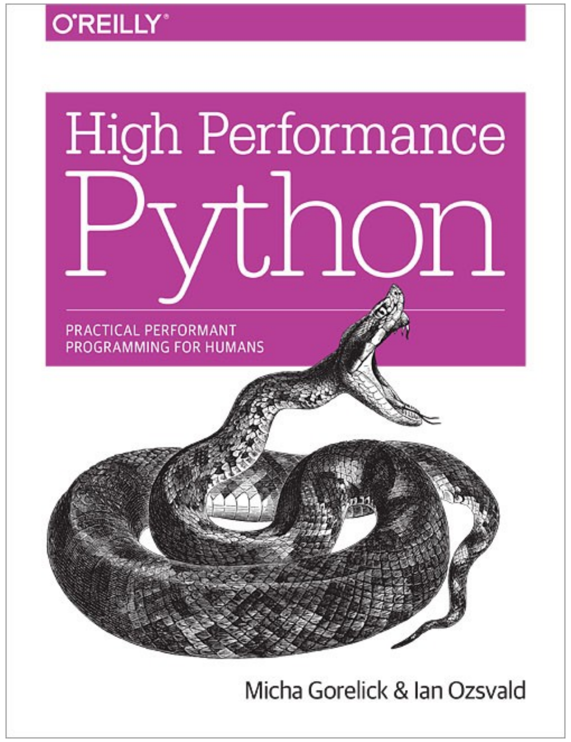
Hotels.com™

Mor Consulting

QBE

https://ianozsvald.com/about-me/

PyData


O'REILLY

High Performance Python

PRACTICAL PERFORMANT PROGRAMMING FOR HUMANS

Micha Gorelick & Ian Ozsvald

## Goals for the course

- Improve your confidence & skills with new process
- Take a bad scenario into a good scenario
  - Use new tools and process
  - Discuss why these are good ideas
- Keep discussing how to get these techniques back to the office

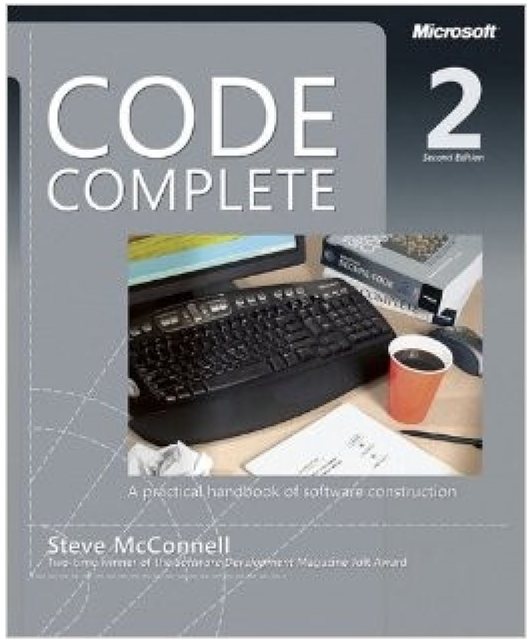
## Slack

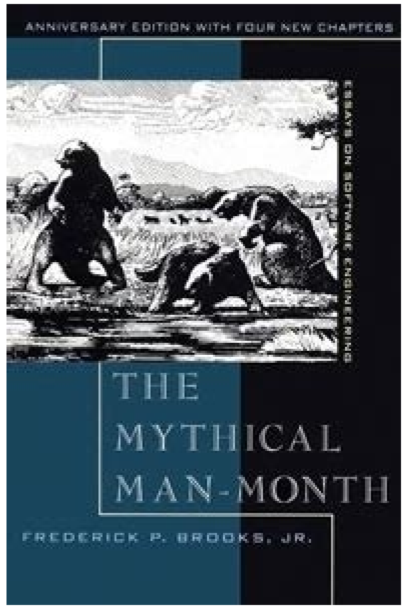
- Join it
- Intention – share & collaborate
- When? Now and future classes – you can stick around
- Share nothing private!

## Introduce yourselves

- Why are you here?
  - Your name and organisation
  - Why are you here? What’s the pain?
  - PyData member? Open Src contributor?

# Some of my experiences




ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS
ESSAYS ON SOFTWARE ENGINEERING
THE MYTHICAL MAN-MONTH
FREDERICK P. BROOKS, JR.

- "Learned it the hard way"
- Started with C++
- Python process was "consensus driven" for years
- Data science process is now "consensus driven"

## Your experiences?

- What SW Eng pains have you experienced?
- Worst bug? Most obscure code? Weirdest issue debugged? Worst Pandas problem?
- Ian’s war story (insurance)
- Photos → Slack (IAN)

## Can we get a photo please

- I like photos, it’ll go on my blog
- Feel free to object if you’d rather not be in it (totally cool)

## Goals for the course

- Improve your confidence & skills with new process
- Engineer a good scenario from a bad start
  - Use new tools and process
  - Discuss why these are good ideas
- **Keep discussing how to get these techniques back to the office**

## First – check your machines are setup

- Start Jupyter
- Go into “session1”
- “check_installed_versions.ipynb”
- All ok?

## What’s a code review?

- Does anyone know what it is?
- Why would we do it?
- Are you comfortable having someone else critique your work?

## Scenario

- Less experienced colleague wants to get into research
- They've built a home temp/humidity analysis system
- If they demo it, maybe they get to join a new team
- How can we help them do well?
- *Their goal – learn how they lose water and heat at home*

# Good code...

- Looks clean (**PEP8?**) & makes sense (bonus – has tests)
- Avoid confusion and repetition
- Has clear and easy to read diagrams (DSci specific)
- All the text is accurate and descriptive
- Nothing feels “broken” or “half-baked” or “risky”

## Doing a code review (practical)

- "session1" – take a look at the folder & the eda3 notebook – Ian to assign where you'll start
- What's wrong in here? Make a list, note worst offenders
- What's your best advice to the author to reduce harm, mistakes or confusion to keep their velocity high?

# Let’s fix our code

- Recap – “Code Reviews.pdf” document, see last page
- Fixing Notebooks safely
  - Copy the Notebook
  - Rerun the copy – did it work?
  - Let’s make it reproducible

## Automatic diagnostics

- “flake8” is a *linter* – it spots boring mistakes reliably
- “black” is a *formatter* – it boringly rewrites your code to PEP8 (so you don’t have to memorise the rules)
- “nbqa” runs tools like these on a Notebook, otherwise we’d be stuck with text scripts only
- “jupytext” lets us use an IDE on our Notebooks

## First clean-up done

- We've *refactored* our code
- What recommendations might you have for the colleague to carry on their clean-up?
- How much money would you bet that the calculations are correct?

## Testing – why bother?

- How confident are we that our code is right? What might change in the future?
- If we change/refactor our code – confident?
- If colleagues look at our code – should they be confident? Are we wasting their time?
- War story – Knight Capital, 400M AUM, 45mins->bust

## Testing – how?

- Write functions with known inputs and expected outputs
- “pytest” is a standard tool, builds on built-in “unittest”
- Test Driven Development vs Test After Development
- *Having tests is the key (bonus – how many is “enough”?)*

## Testing – tools?

- A “unit test” is a testing function that tests 1 unit of code – often a function, sometimes several functions
- *pytest* is the de facto unit test choice in Python
  - It extends the older *unittest* module
- Functional/black box tests might test larger chunks of code e.g. end-to-end processes (not units)

## Let’s write a test, then solve tests

- Look in “tests/”
- We’ll build up
- Critical – “def test_…”
- Critical – some “assert” statements
- Now solve those test challenges...

```python
import pytest

def double(x):
    """Double x"""
    return x * 2

def test_double():
    # expected, double, answer could be called whatever you want
    # the only critical bit is the function name starting "test_"
    # plus some assert statements
    expected = 20
    assert double(10) == 20, "Expecting x*2"
```

## Next steps

- With tests and clean code is our scenario-colleague more likely to impress during their interview? Why?
- *How will you use this in your own code?* What’s valuable or “not clearly of value”?
- Discuss the Primes homework + “Code Reviews.pdf”
- Could you do a mutual code review before next time?