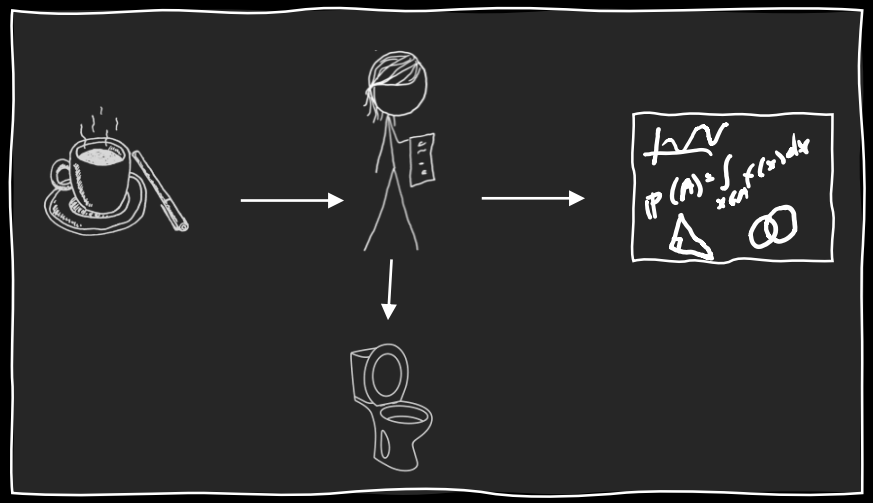
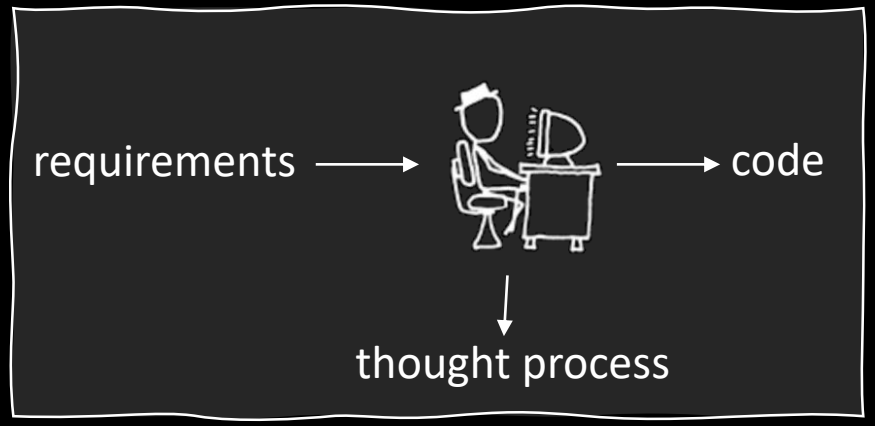
IA Scientific Computing

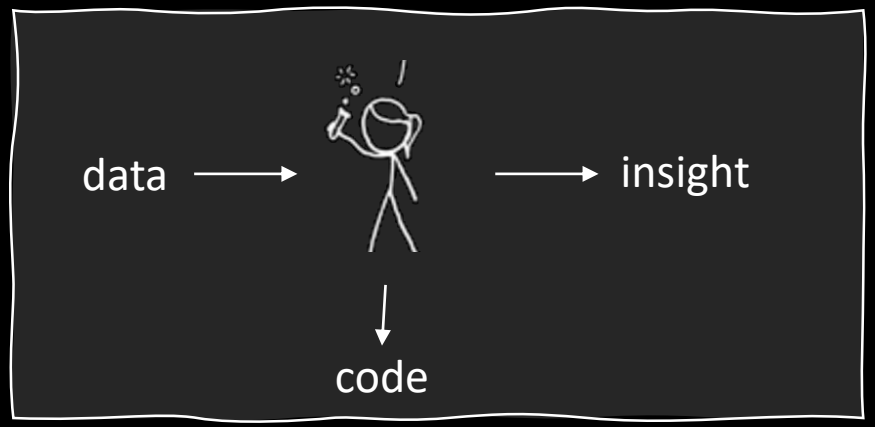
Dr Damon Wischik



"A mathematician is a machine for turning coffee into theorems" – Erdős / Rényi

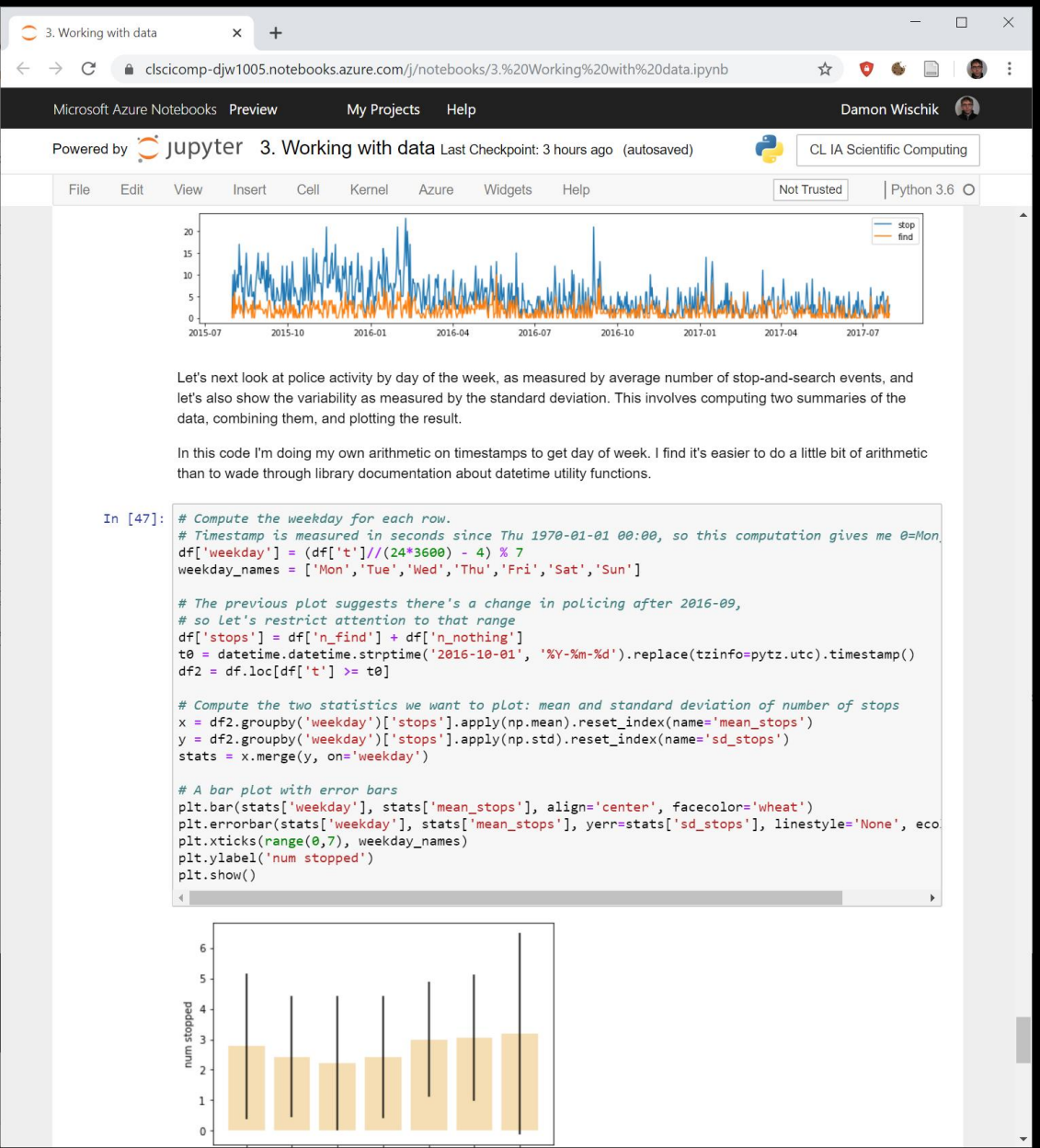


A software engineer



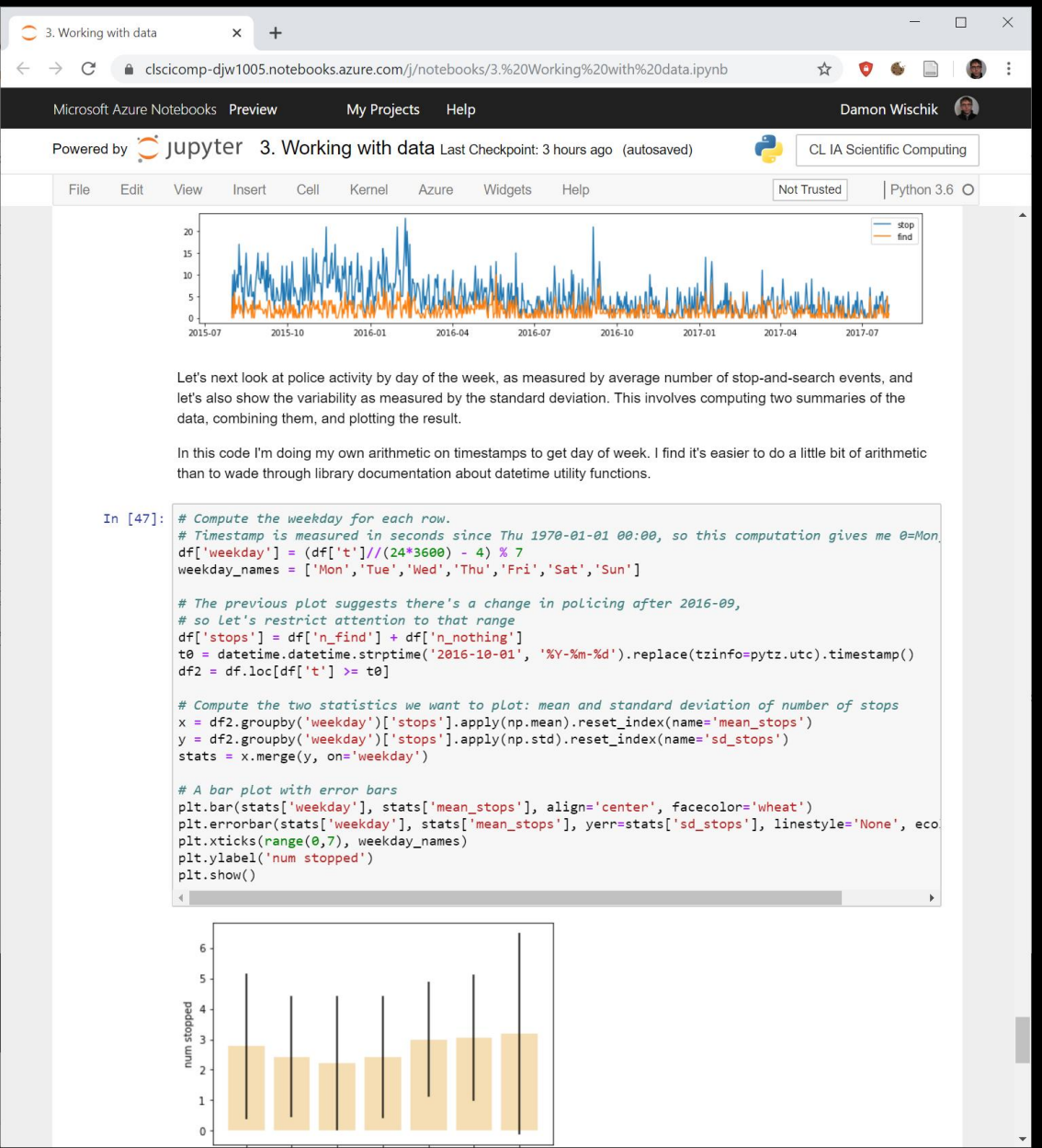
A scientist / machine learner / modeller

What is scientific computing?



Jupyter notebooks
Python
numpy + pandas + matplotlib

How to write bad code



first I ran this cell

and now this cell is producing strange answers

Then this cell, I think

What does good code look like?

What Not to Do

- Your ML has doubtless been one big file where you threw together all the functions and value declarations
- Lots of C programs look like this :-(
- We *could* emulate this in OOP by having one class and throwing everything into it
- We can do (much) better

OOP Concepts

- OOP provides the programmer with a number of important concepts:
  - Modularity
  - Code Re-Use
  - Encapsulation
  - Inheritance (lecture 5)
  - Polymorphism (lecture 6)
- Let's look at these more closely...

Modularity and Code Re-Use

- You've long been taught to break down complex problems into more tractable sub-problems.
- Each class represents a sub-unit of code that (if written well) can be developed, tested and updated independently from the rest of the code.
- Indeed, two classes that achieve the same thing (but perhaps do it in different ways) can be swapped in the code
- Properly developed classes can be used in other programs without modification.

How to make effective use of notebooks

"Every line of code is written without reason, maintained out of weakness, and orphaned by chance."

Jean-Paul Sartre

"Look at each line of your code and ask yourself: 'does this spark joy?' If not, don't keep it."

Marie Kondo

For this course, you should create your own notebooks from scratch.

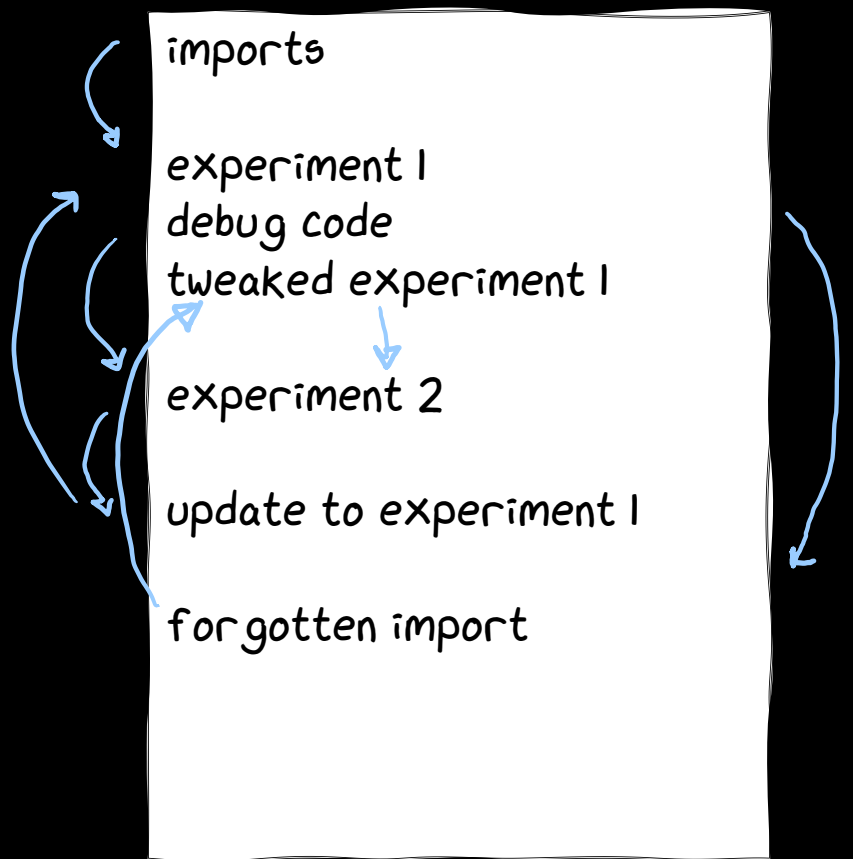
- A notebook is not a logfile.
  It's a document, which you edit and reshape.
- A notebook is not a source file.
  Text cells are more important than code cells.
- A notebook is for exploring and explaining.
  One-off tasks.
  Many of the questions are one- or two-liners.

https://programmingisterrible.com/post/139222674273/write-code-that-is-easy-to-delete-not-easy-to extend
"Effective Jupyter notebooks" https://news.ycombinator.com/item?id=19860955
"JupyterCon: I don't like Notebooks" https://news.ycombinator.com/item?id=17856700

Don’t set traps for yourself with disorganized notebooks

- You should accumulate code during a work session, but trim it at the end of each session.
- Your reader should be able to understand your work by reading top-to-bottom.
  Your code should all work if run top-to-bottom

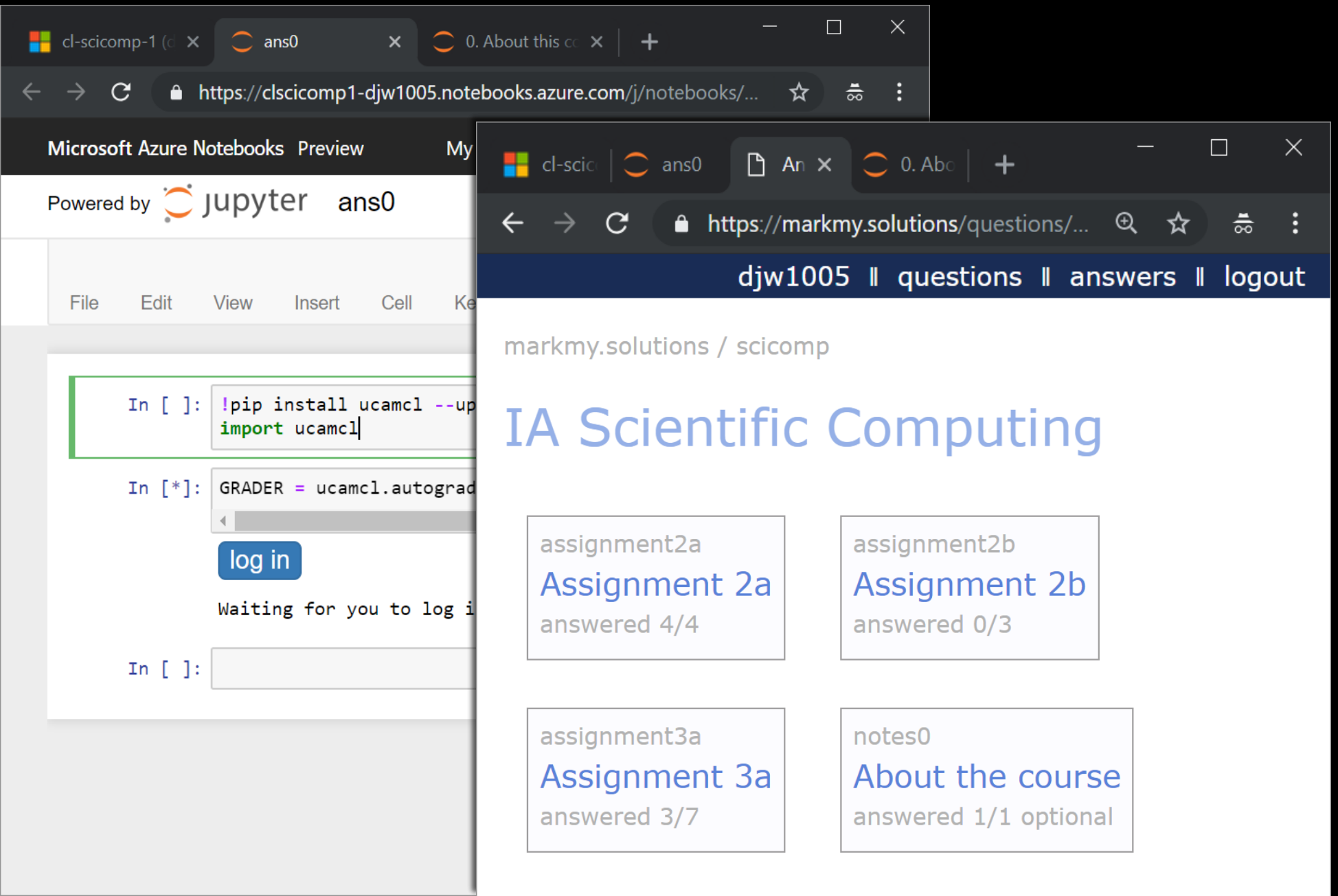
while working



after you’ve finished

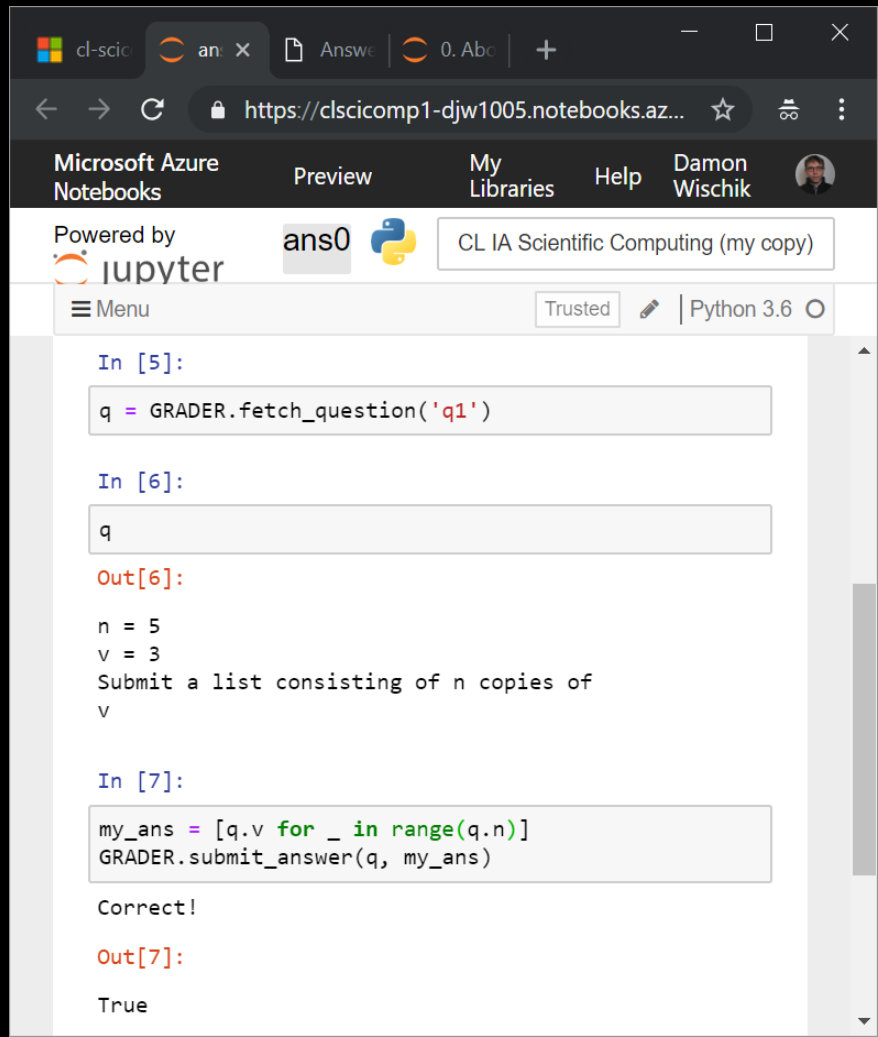
```
imports

utility functions

run-once setup code
functions that implement
   your solutions

import autograder
submit solutions to
   autograder
```

Submitting your answers to the autograder

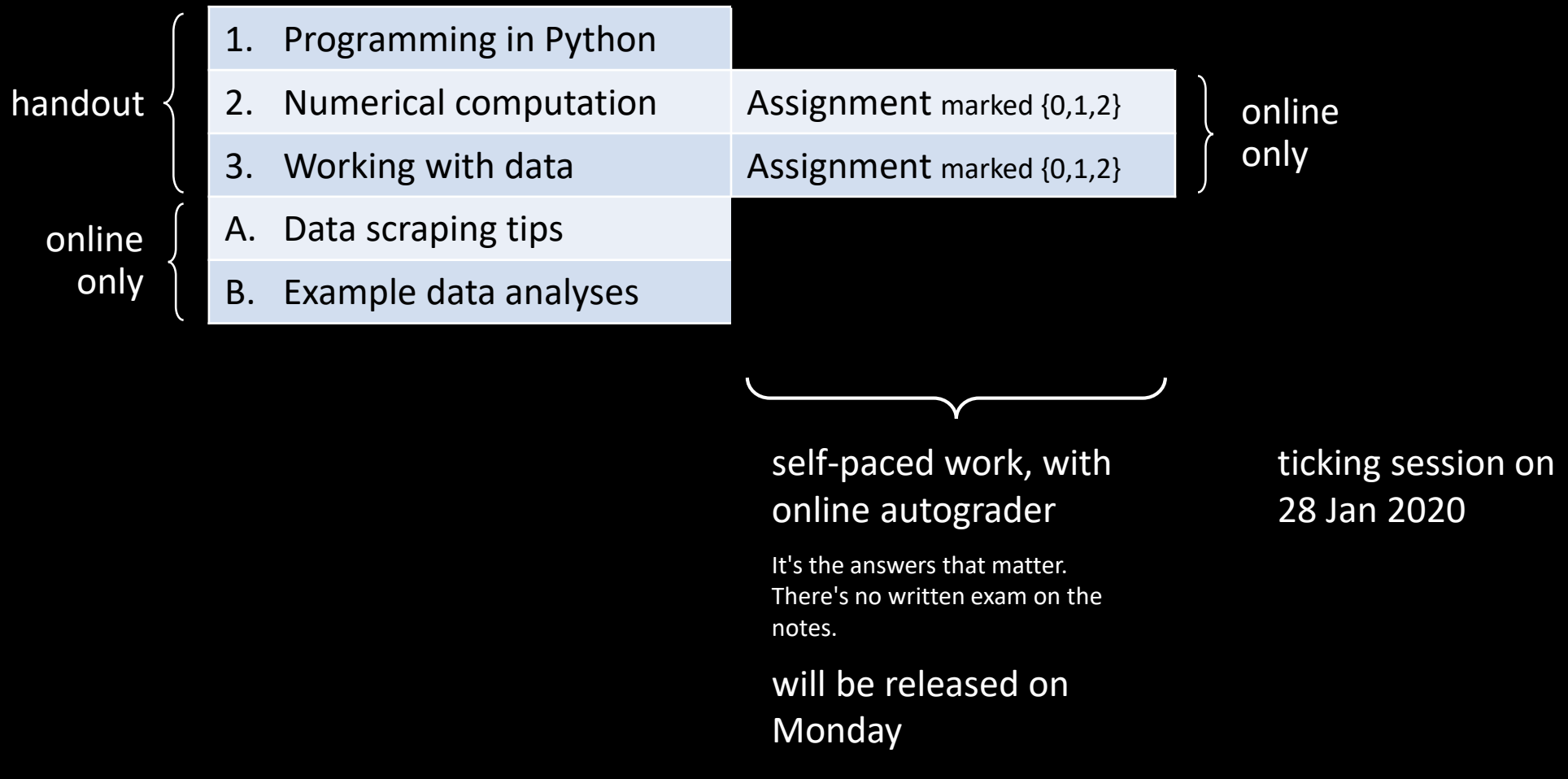


Submitting your answers to the autograder

```
# Code to run in the preamble to the “submit solutions” section of your notebook:

!pip3 install ucamcl
import ucamcl
GRADER = ucamcl.autograder(‘https://markmy.solutions’, course=’scicomp’).subsection(’notes0’)

# To answer a question:

q = GRADER.fetch_question('q1’)
my_ans = ... # using the parameters in q
GRADER.submit_answer(q, my_ans)
```

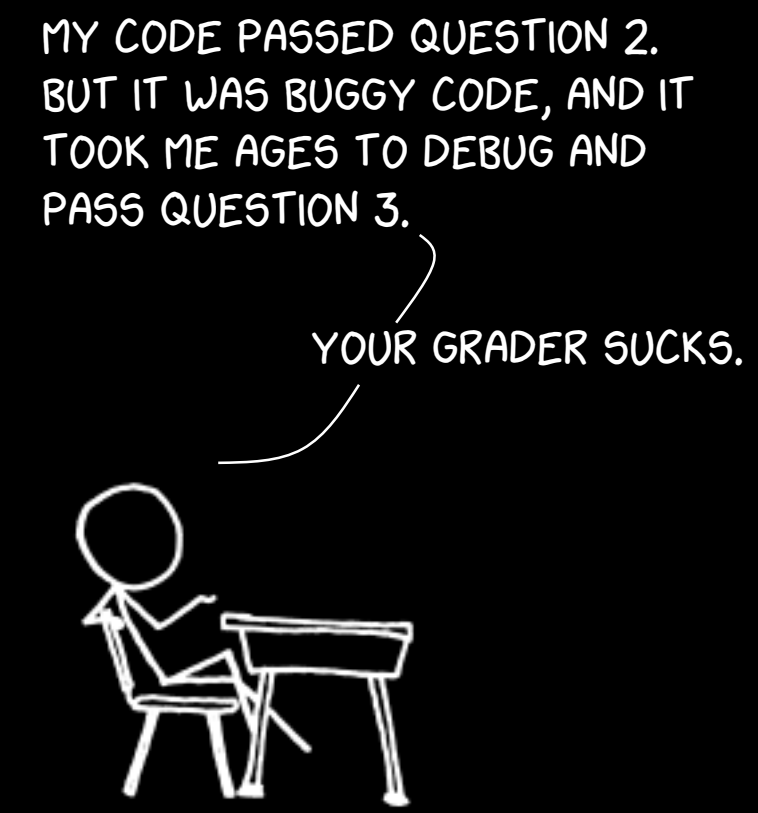


How is the course structured and examined?

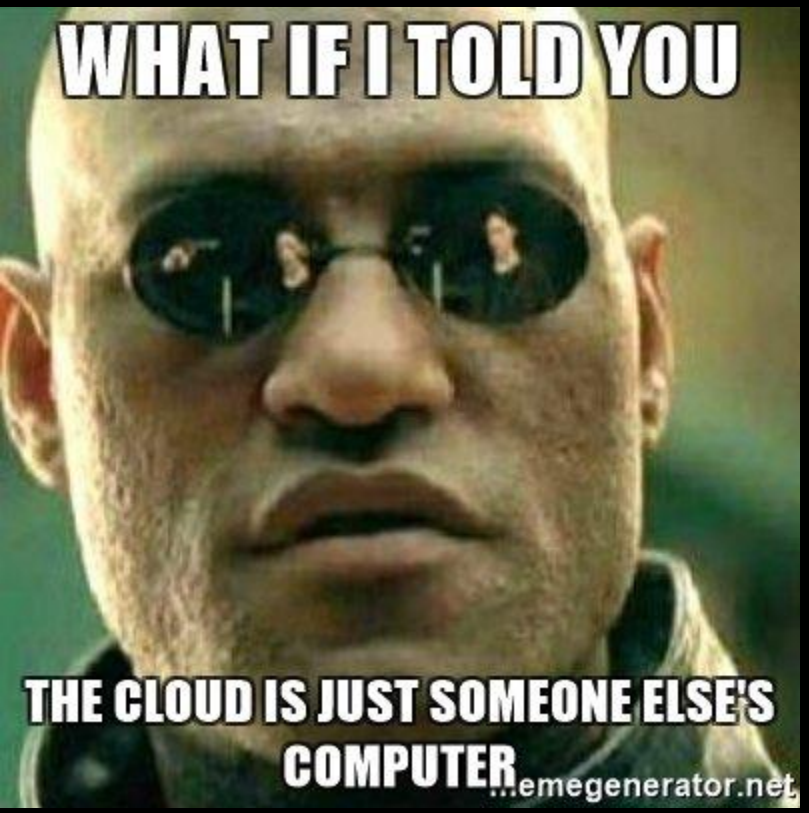


marking academic work ≠ testing code

- It's *your job* to write your own unit tests.
- Nature's datasets are full of quirks, and nature is out to deceive you.
- As a data scientist you need to be ever vigilant, always checking your assumptions.



Running Jupyter



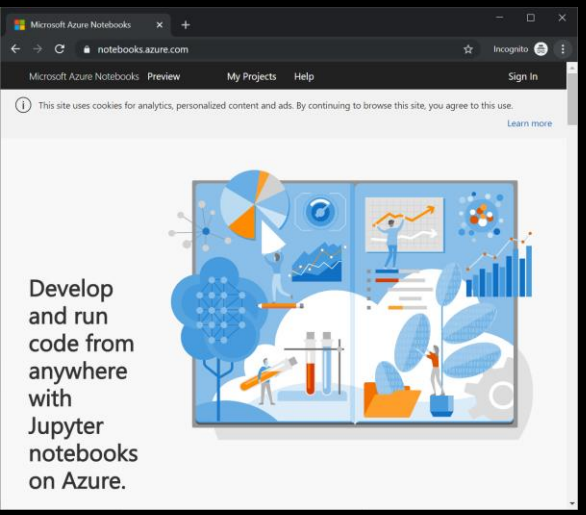
... and it will fail when you need it most.

Backup!

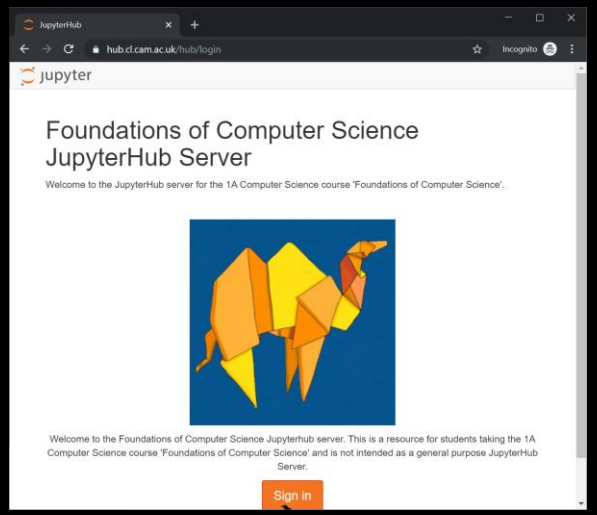
Running Jupyter

The autograder will run wherever you run Jupyter + Python3.

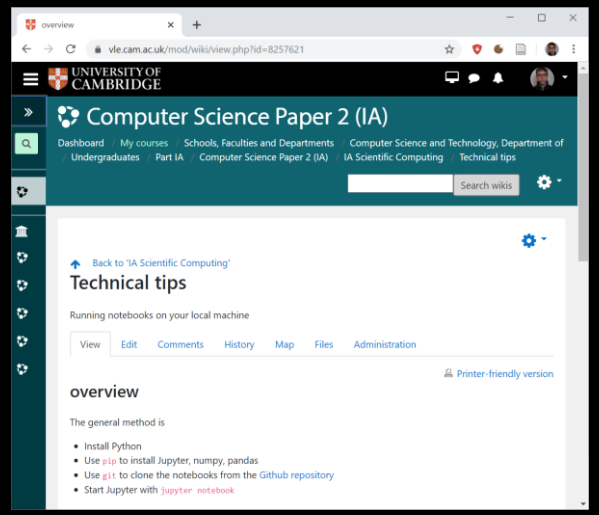
notebooks.azure.com



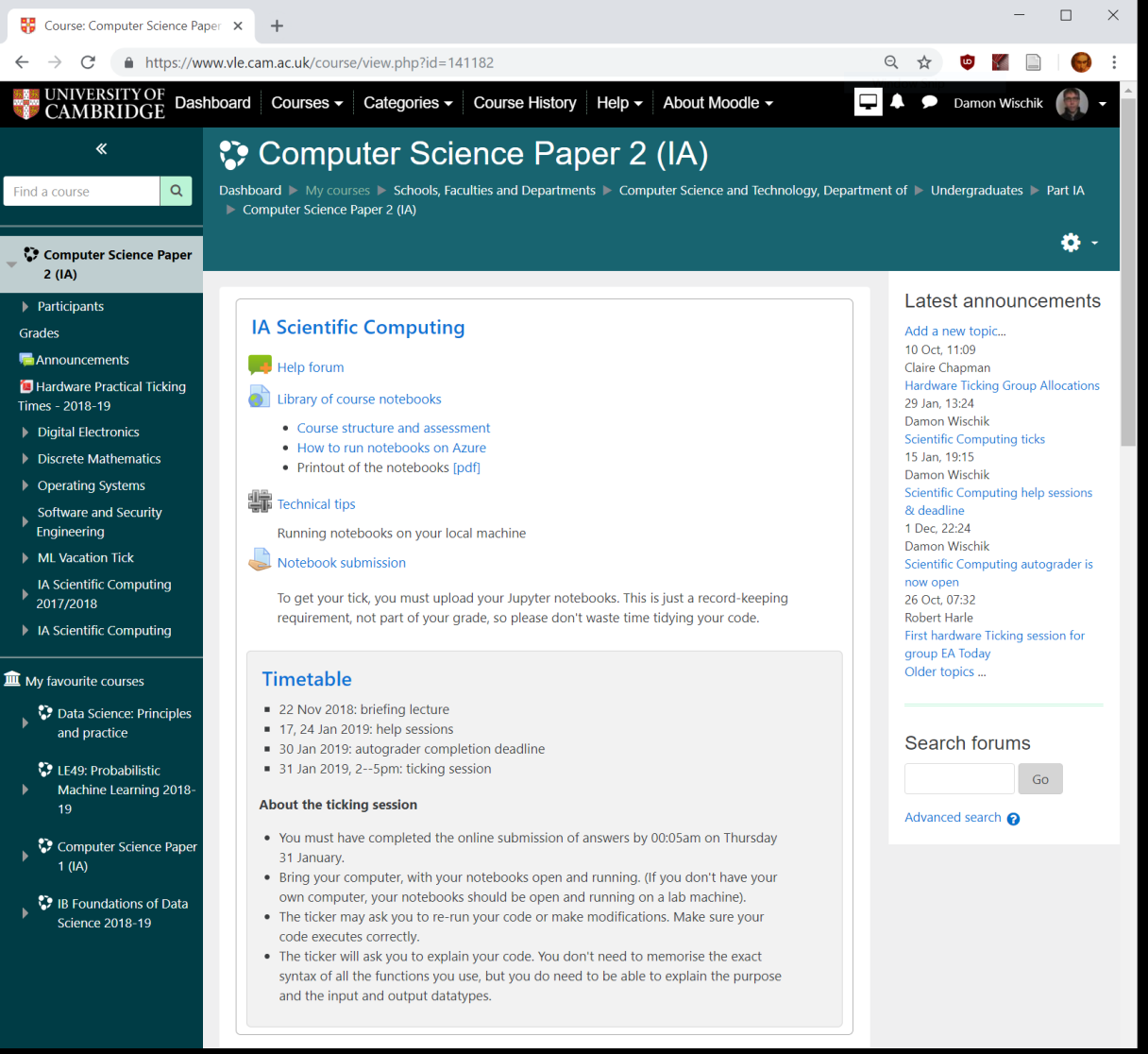
hub.cl.cam.ac.uk



your own machine



(installation tips on Moodle)



Where do I go for help?

1. StackOverflow
2. Moodle help forum
3. Help sessions at the beginning of Lent term