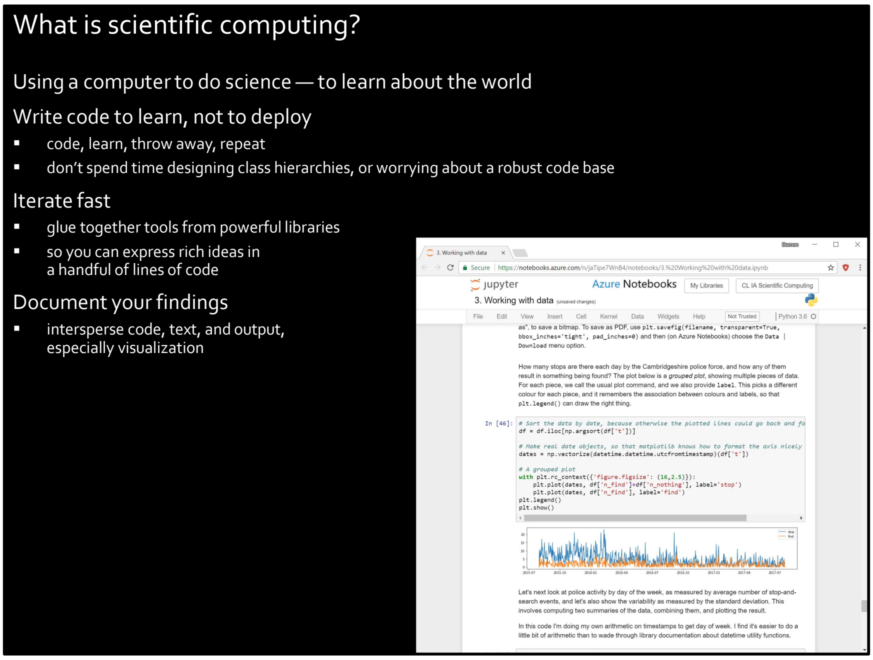
Briefing for Computer Science undergraduates

IA Scientific Computing

Dr Damon Wischik



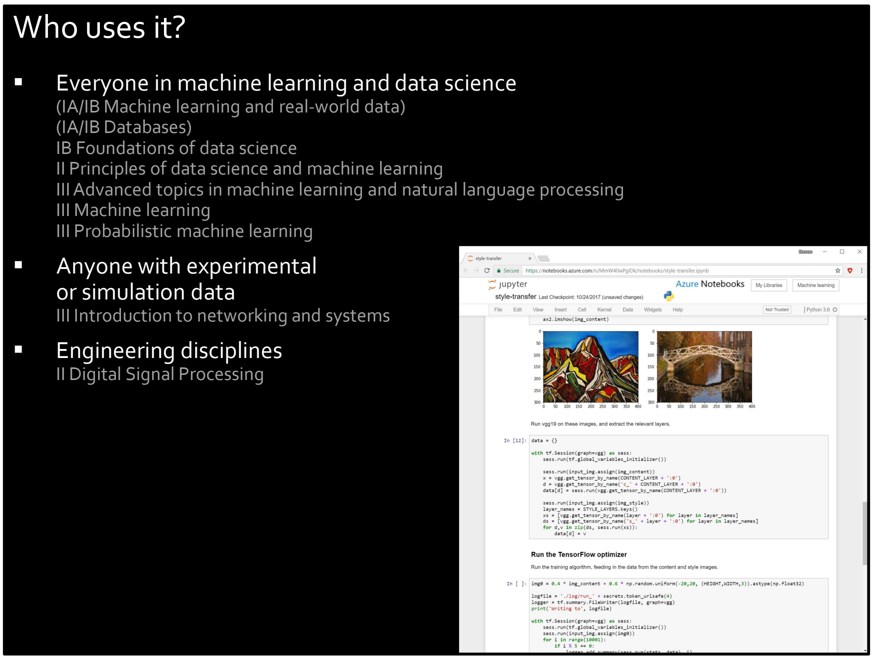
So far you've learnt

- ML – it gives you a better understanding of what Computer Science *is,* what types really are, what functions really are, etc.
- Java – it teaches you the good practices of software engineering, how to build good resilient code that won't break when you have new feature requests or new programmers joining your team

Now you're learning Scientific Computing, i.e. how to use a computer to do science, i.e. trying to learn about the world. You're not coding in order to deploy a product, you're coding in order to learn something. Your end product is A Piece of Understanding.

For scientific computing, you want exploration to be as fast as possible: you have a thought, you type in one or two lines, you see the answer, you have another thought, ... You can't spend a week trying to design a good architecture – you want a language that doesn't impede you in any way; and you want flexible powerful toolkits. So your code is typically just a few lines long.

Your end product isn't a product, it's a piece of *documented* understanding about the world. There’s a big emphasis on documenting your reasoning, your findings (especially tables and figures), and your conclusions. We’ll use Jupyter Notebooks, a widespread platform for doing / documenting scientific computing. It’s great for combining code and output and text.



- Everyone doing data science: In companies like Google and Facebook, whenever you're working on data and communicating your results to the rest of your team (e.g. tracking your userbase, monitoring your system's performance), you'll probably be using Jupyter Notebooks. (When you build the finished product, you won't use notebooks, you'll use a proper software engineering toolchain including version control etc.)
- Several other tripos courses on machine learning and data science use notebooks. (At the moment, IA/IB MLRD and Databases don't use Jupyter Notebooks, but they'll probably switch over soon.)
- Machine learning: this notebook illustrates using TensorFLow in a Jupyter Notebook, using the Python programming language. This has become the de facto standard for developing machine learning tools and trying them out on data.

What does the course teach?

- Scientific computing using Jupyter Notebooks, hosted by Azure
- Python
  (This is to save you from MATLAB, which NST students learn as part of IA Maths for NST)
- Python libraries for scientific computing: arrays, data, plots

Python is handy for scientific computing. It doesn't have any of the boilerplate of Java.

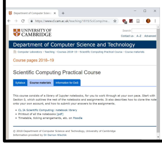
In this course,
- We'll be programming in Python
- We'll learn the specific Python toolkit for scientific computing tasks, especially for efficient processing of arrays and datasets, and for plotting
- We'll be using Jupyter Notebooks to prepare documents of Python code plus text plus output (though the notebooks can be used for other languages also)
- We'll be using Jupyter Notebooks hosted on the Microsoft Azure cloud platform (though you can just as well run it on your own machine)

In previous years, students taking IA Maths for NST also learnt "Scientific Computing in MATLAB", and the NST students still do. When I asked students to plot some data, they used Excel. I asked why, when they've all learnt MATLAB. They said "I never got my head round it, and it didn't seem like a real language." I hope you'll learn to love Python for data handling and plotting!

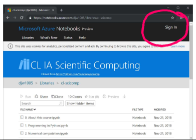
Getting started with Jupyter / Azure

1. Go to the course website
   https://www.cl.cam.ac.uk/teaching/1819/SciComp/materials.html
2. Follow the link to *notebook library*
3. Click on *Sign In*, and sign in using csrid@cam.ac.uk
   You'll be redirected to a Raven login page
4. Click on Clone, to copy all my teaching notebooks to your own personal Azure Notebook space. (You need to do this before you can run any code.) Give it any name and ID you like.
5. Click on any one of the notebooks, and you can start reading it and running code in it. (Notebook 1 tells you how to run code.)

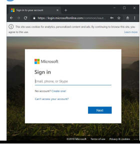
1



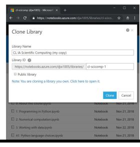
2



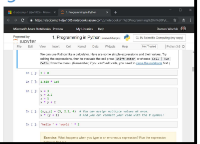
3

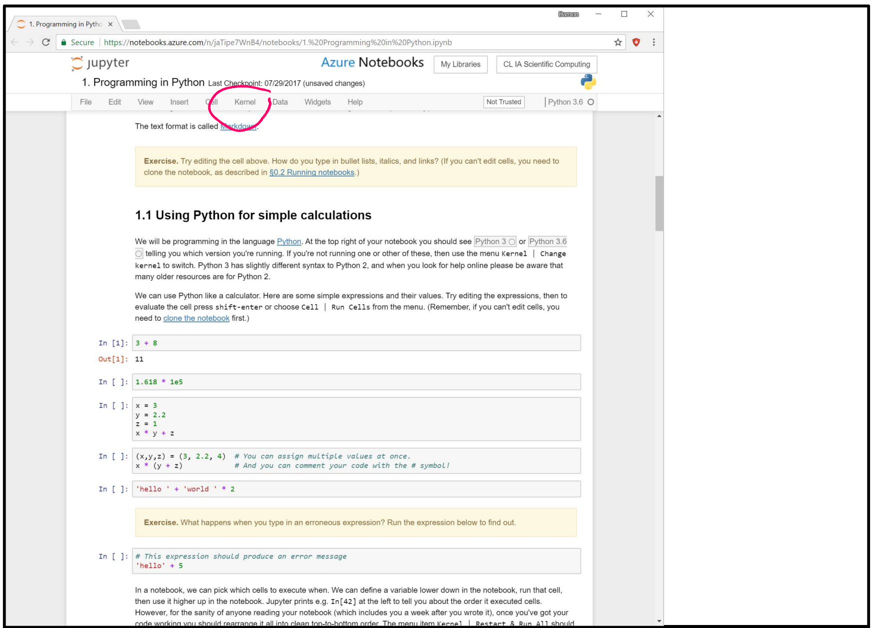


4



5





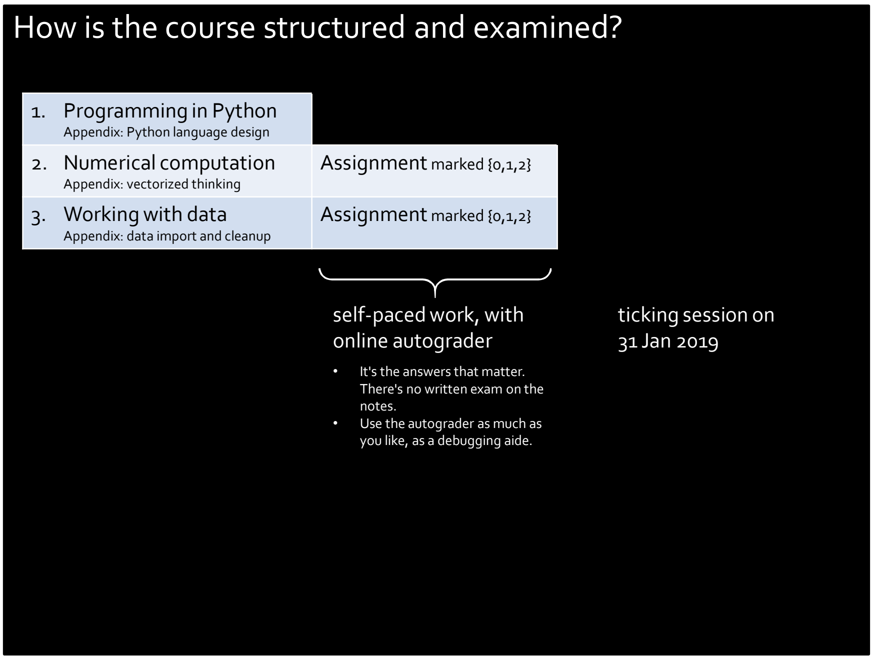
Some gotchas: (1) Notebooks are interactive documents, (2) check your Python version

Jupyter notebooks

- A notebook You choose when to execute cells, and in what order. You can execute a cell at the bottom of the notebook, then execute a cell at the top of the notebook, then edit the cell at the bottom – so that no one reading your notebook has any clue what is what. You'll almost certainly end up producing a spaghetti notebook.
- I recommend you periodically restart the kernel (Kernel | Restart & Clear Output), and make sure that all your code does the right thing when you run it top-to-bottom. This is for the sanity of anyone else who reads your notebook (including you, in the ticking session).
- A notebook is an interactive document. It isn't a source file for a compiler, nor is it a log from an interactive session.

Python versions

- Python has two main versions, 2.7 and 3.x. There are a few big syntactic differences. When you look up help on StackOverflow etc., make sure you're looking at 3.x code.
- Make sure your notebook is running Python 3.6 (your notebook will say, in the top right). If it's not, choose Kernel | Change Kernel.



The handouts are just a printout of the notebooks in the notebook library. There are three notebooks.

- Notebook 1: general introduction to Python. If you've programmed in Python before, you probably don't even need to look at this. If you know Java, then there are a few weird bits of Python syntax that you should know, so have a quick skim through.
- Notebook 2: the Python toolkit for working efficiently with arrays and matrices, the mainstay of scientific computing. And elementary plotting
- Notebook 3: how to load a dataset, how to clean it up, some data transforms, some more plots.

Each assignment is marked out of 2, so the total mark for this course is 4. Nearly everyone gets 4 marks. These are ticks, not tricky challenging exercises.

It's self-paced work. You can start right away. This course should take 10 to 15 hours of study time. If my estimate is way out, let me know.

There is an online autograder. It'll tell you instantly whether your answers are correct or not. Use it as much as you like.

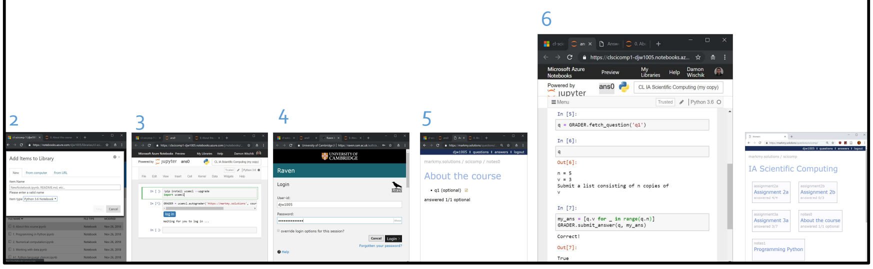
To get the tick, you must (1) pass the autograder, (2) answer some questions about your work.

Using the autograder

1. Go to your notebook library
2. Click on + New to create a new notebook, which you'll use to document your answers. Make sure it's type "Python 3.6 notebook" and give it a name ending in .ipynb
3. Run the magic commands listed in Notebook 0:

```
!pip install ucamcl -upgrade
import ucamcl
GRADER = ucamcl.autograder('https://markmy.solutions', course='scicomp', section='notes0')
```

4. Click on log in and log in using your Raven csrid
5. The login page will change to show you your current marks
6. Go back to the Jupyter page, and you can fetch questions and submit answers, following the instructions in Notebook 0.



Some notes

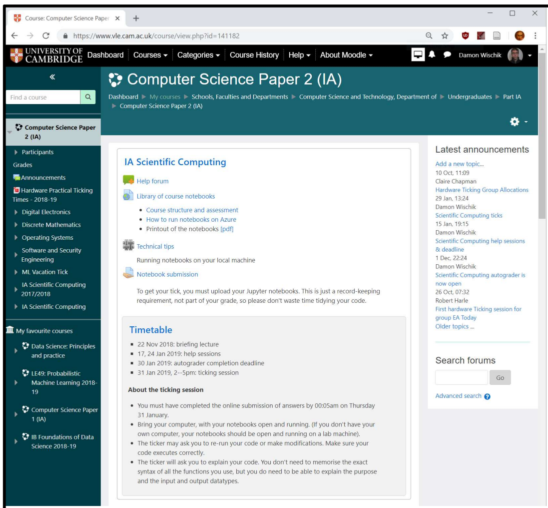
- While a command is executing, it shows as "In [*]"
- Each assignment and section of the notes will tell you what string to use for "section"
- In the assignment, each question will tell you something like

```
q = GRADER.fetch_question('q1')
# Let my_ans be a list consisting of q.n copies of q.v
GRADER.submit_answer(q, my_ans)
```

  It's your job to write the code in the middle. In this case,
  my_ans = [q.v for _ in range(q.n)]
- Each question object **q** comes with question parameters. Each student gets questions with different parameters, each time you try to submit an answer. You can see the full list of parameters in a question just by evaluating **q**, and you can access the parameters by e.g. **q.n**
- If you submit the wrong answer, it'll tell you why your answer is wrong (e.g. wrong type, wrong length of list, etc.). In that case you should fetch the question again, and try again. (If you try to re-use the old **q** object, it will print out a message "That question is stale".)
- On the Autograder/Mark page (the page you see after clicking on "log in"), it

shows you your marks so far. (The page doesn’t auto-refresh, so you may need to refresh it yourself.) If you answer correctly then answer incorrectly, it’ll only remember your correct answer.

- You can browse around the Autograder/Mark page, to show you marks for all the other sections of this course
- There’s no penalty for using the autograder. Use it as often as you like, to help with debugging. Think of it a bit like unit testing.



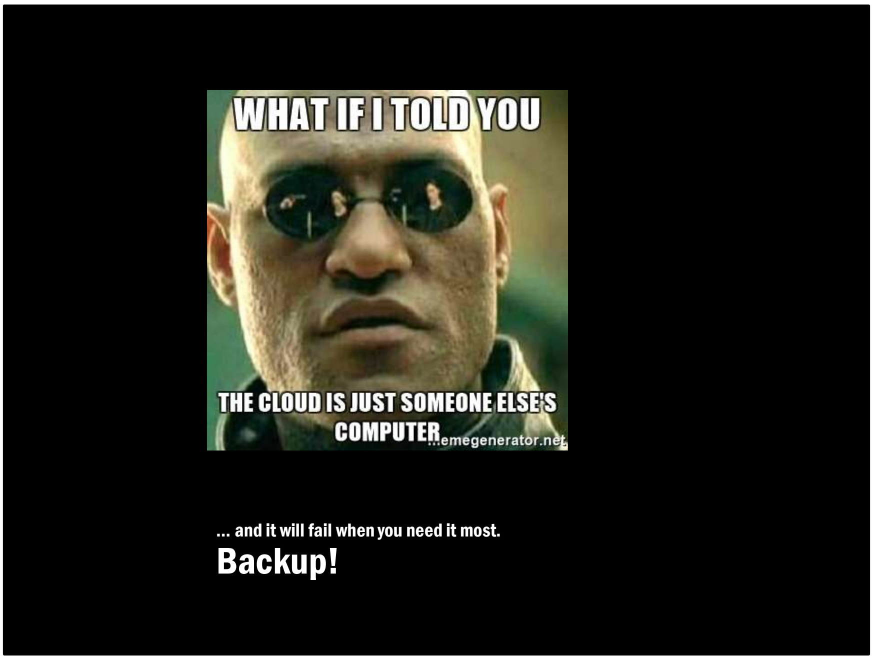
Where do I go for help?

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

Please go and look at other resources, e.g. StackOverflow and documentation. On the Moodle forum, please answer each other's questions, rather than waiting for me!

The point of scientific computing is that it's a glue language, for assembling powerful tools and gluing them together. So you should spend a lot of your time hunting around for the right tools. The only real skills of scientific computing are (a) knowing roughly what tools are available, (b) being able to find help on the exact way to use them. And you need to know the bare bones of Python, to be able to glue pieces together.

Help sessions at the beginning of Lent term: dates are listed on Moodle.



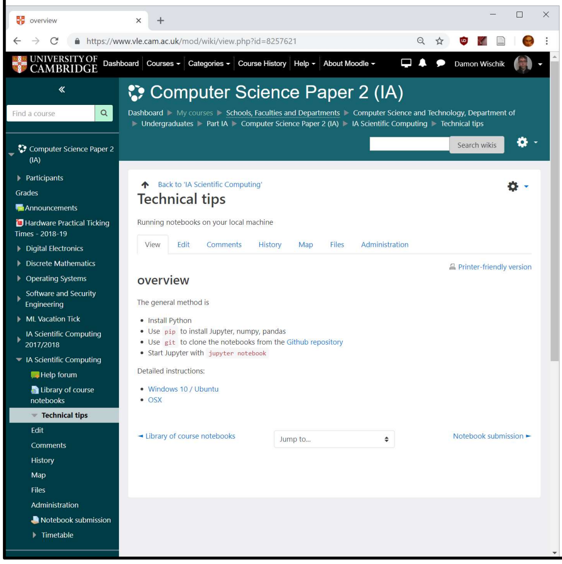
We're using Jupyter notebooks hosted by Microsoft Azure.

- It's very handy: all you need is a web browser, and you don't need to bother about installing software or software versioning etc.
- It's in the cloud, i.e. on someone else's computer, and it *will* fail. I will not have any sympathy if it fails the night before the ticking deadline. It's an important life lesson: don't rely on the cloud!

Back up your work! It's unlikely there'll be a complete failure of Microsoft Azure storage, but you should always keep backups. Section 0 of the notes explains how.

Microsoft wants you to get used to using Microsoft's cloud. "You only get to sell your soul once. Make sure you get a good price." If you prefer not to use Jupyter Notebooks hosted by Microsoft Azure, you can also run Jupyter locally...

Running Jupyter on your local machine



If you have tips, please contribute to the wiki on the Moodle course page.

You can also run Jupyter notebooks on your local machine. There are some instructions in a wiki section on Moodle.
(This is my best attempt, but everyone’s computer is different. If you have any helpful tips, please add them to the wiki.)