IA Scientific Computing

BRIEFING LECTURE

In this years performance of IA Scientific Computing

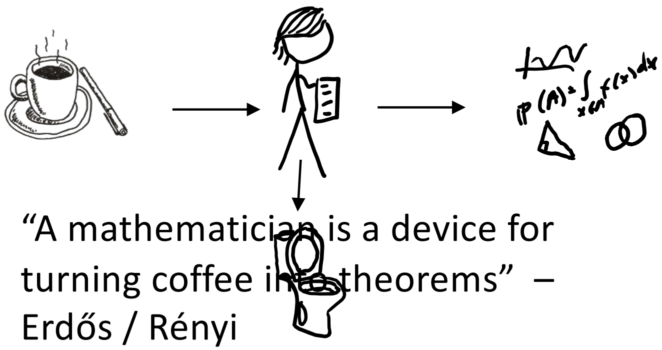
Damon Wischik's part will be played by
the understudy Andrew Moore

Scientific computing

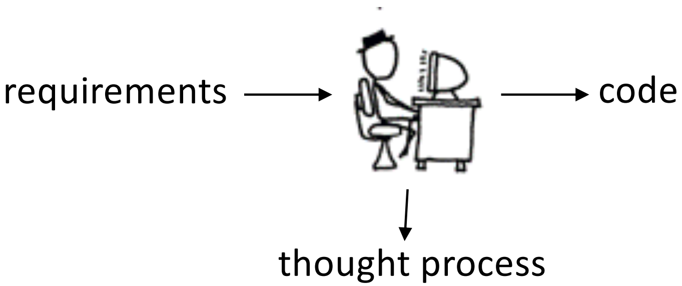
- computing as a tool for doing science

Computer science

- the study of computation

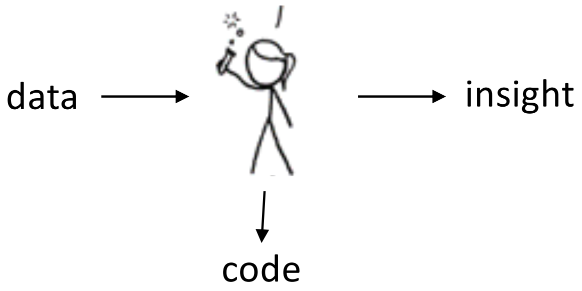


"A mathematician is a device for turning coffee into theorems" – Erdős / Rényi



requirements → code

thought process

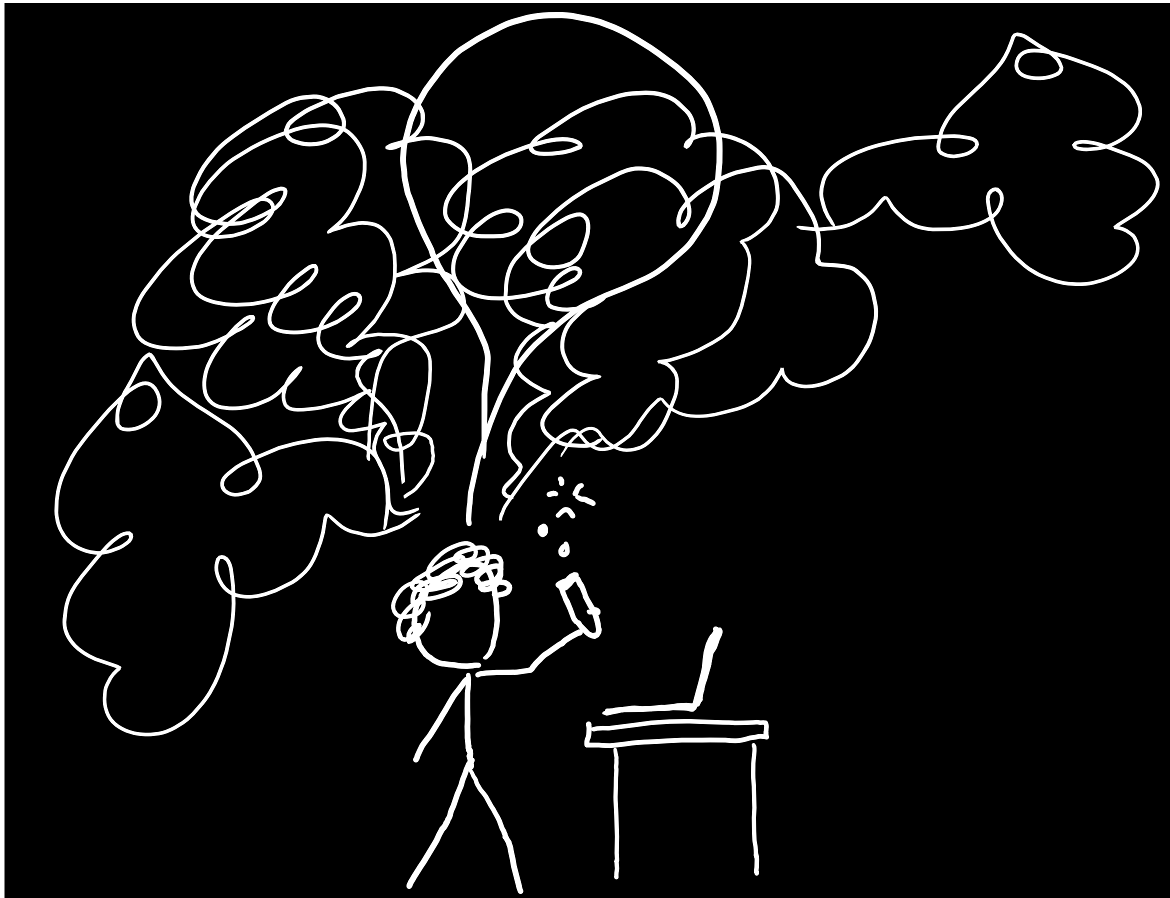


data → insight

code

SCIENTIFIC COMPUTING

Try out an idea ✦ see what happens ✦ refine your idea ✦ try something else ✦ iterate … ✦ share what you’ve learnt

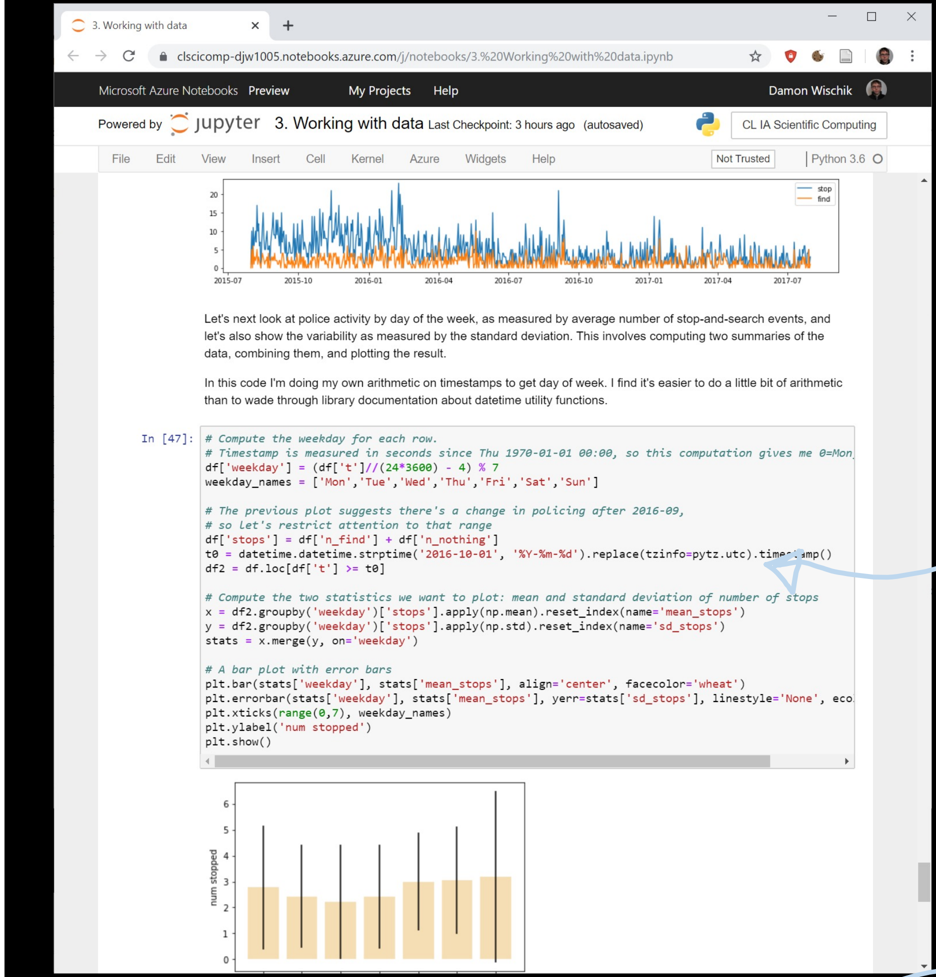


CODE AT THE SPEED OF THOUGHT

- Concise one- or two-liners for one-off tasks
- Rich, expressive libraries & glue code

Scientific computing

= Python + numpy + plotting + pandas
+ Jupyter notebooks



Let's next look at police activity by day of the week, as measured by average number of stop-and-search events, and let's also show the variability as measured by the standard deviation. This involves computing two summaries of the data, combining them, and plotting the result.

In this code I'm doing my own arithmetic on timestamps to get day of week. I find it's easier to do a little bit of arithmetic than to wade through library documentation about datetime utility functions.

In [47]:

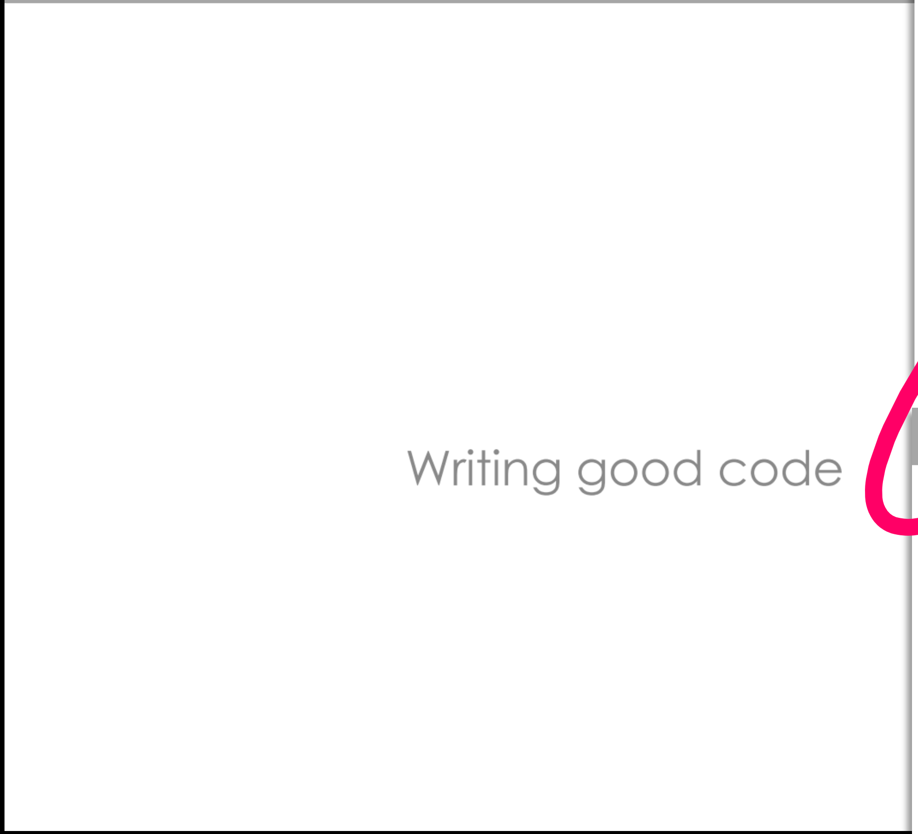
```
# Compute the weekday for each row.
# Timestamp is measured in seconds since Thu 1970-01-01 00:00, so this computation gives me 0=Mon,
df['weekday'] = (df['t']//(24*3600) - 4) % 7
weekday_names = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun']

# The previous plot suggests there's a change in policing after 2016-09,
# so let's restrict attention to that range
df['stops'] = df['n_find'] + df['n_nothing']
t0 = datetime.datetime.strptime('2016-10-01', '%Y-%m-%d').replace(tzinfo=pytz.utc).timestamp()
df2 = df.loc[df['t'] >= t0]

# Compute the two statistics we want to plot: mean and standard deviation of number of stops
x = df2.groupby('weekday')['stops'].apply(np.mean).reset_index(name='mean_stops')
y = df2.groupby('weekday')['stops'].apply(np.std).reset_index(name='sd_stops')
stats = x.merge(y, on='weekday')

# A bar plot with error bars
plt.bar(stats['weekday'], stats['mean_stops'], align='center', facecolor='wheat')
plt.errorbar(stats['weekday'], stats['mean_stops'], yerr=stats['sd_stops'], linestyle='None', eco
plt.xticks(range(0,7), weekday_names)
plt.ylabel('num stopped')
plt.show()
```

First I ran this cell up here

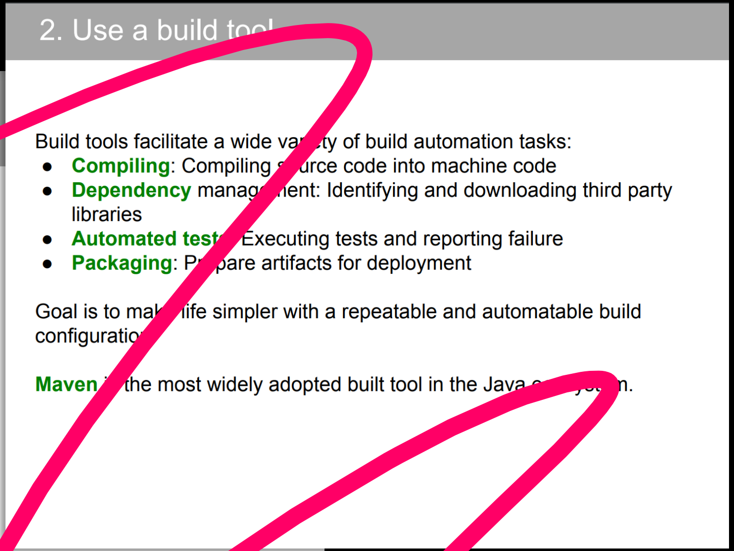
And now this cell is producing strange answers

Then this one, I think.

Lecture notes from IA OOP



Writing good code



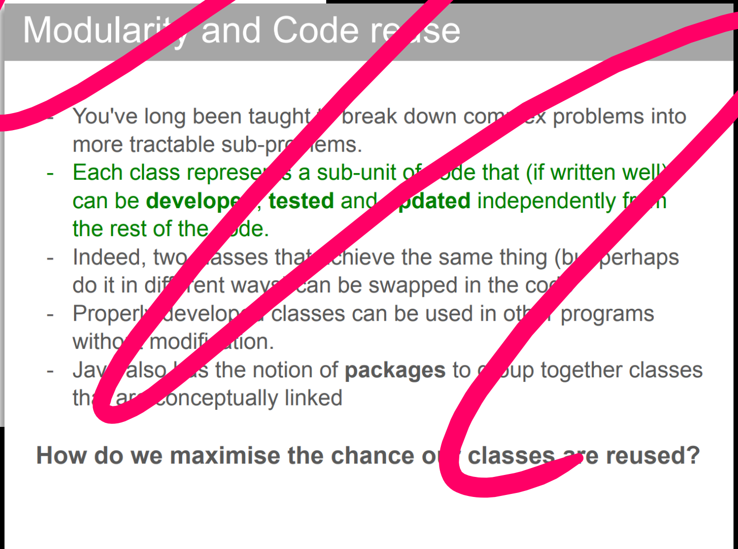
2. Use a build tool

Build tools facilitate a wide variety of build automation tasks:

- **Compiling**: Compiling source code into machine code
- **Dependency** management: Identifying and downloading third party libraries
- **Automated testing**: Executing tests and reporting failure
- **Packaging**: Prepare artifacts for deployment

Goal is to make life simpler with a repeatable and automatable build configuration.

Maven is the most widely adopted built tool in the Java ecosystem.



Modularity and Code reuse

- 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.
- Java also has the notion of **packages** to group together classes that are conceptually linked

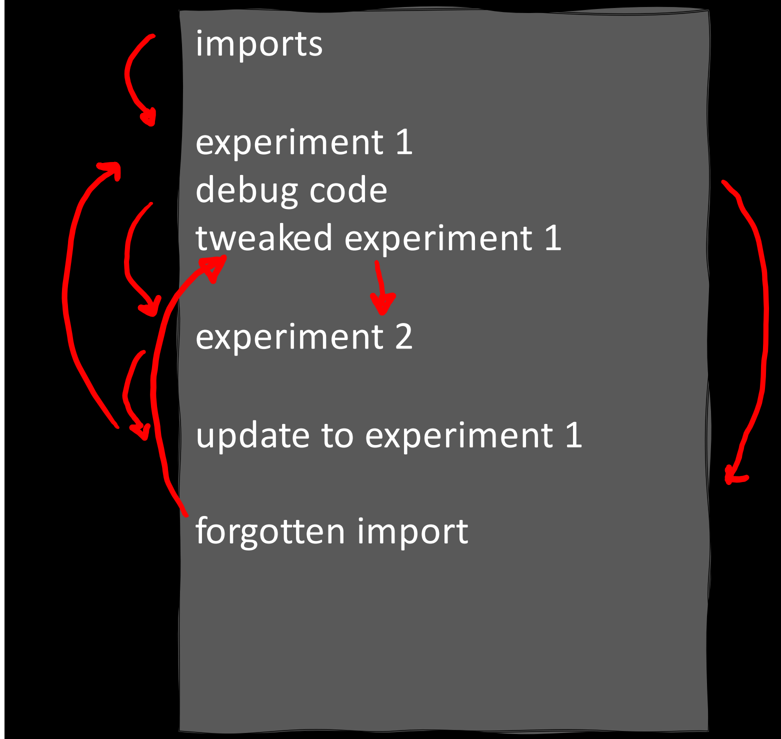
How do we maximise the chance of classes are reused?

Bad advice for scientific computing

Marie Kondo,
de-cluttering guru

Look at each line of your code and ask yourself: ‘does this spark joy?’
If not, delete it.

while working



imports

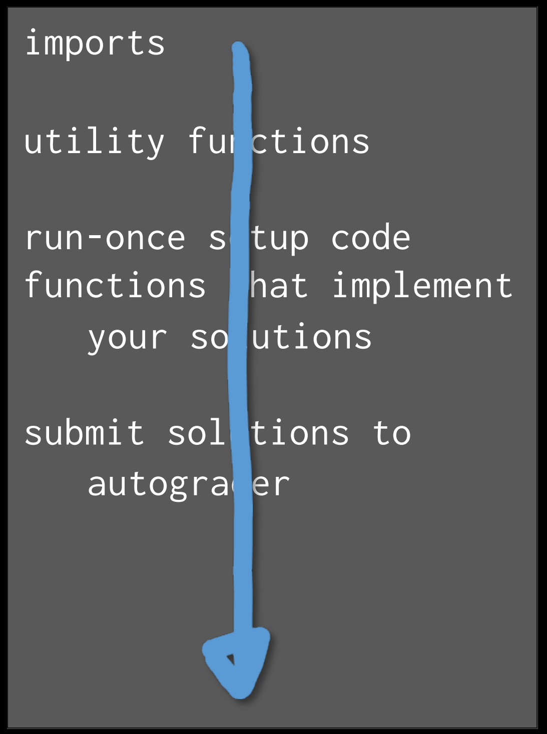
experiment 1
debug code
tweaked experiment 1

experiment 2

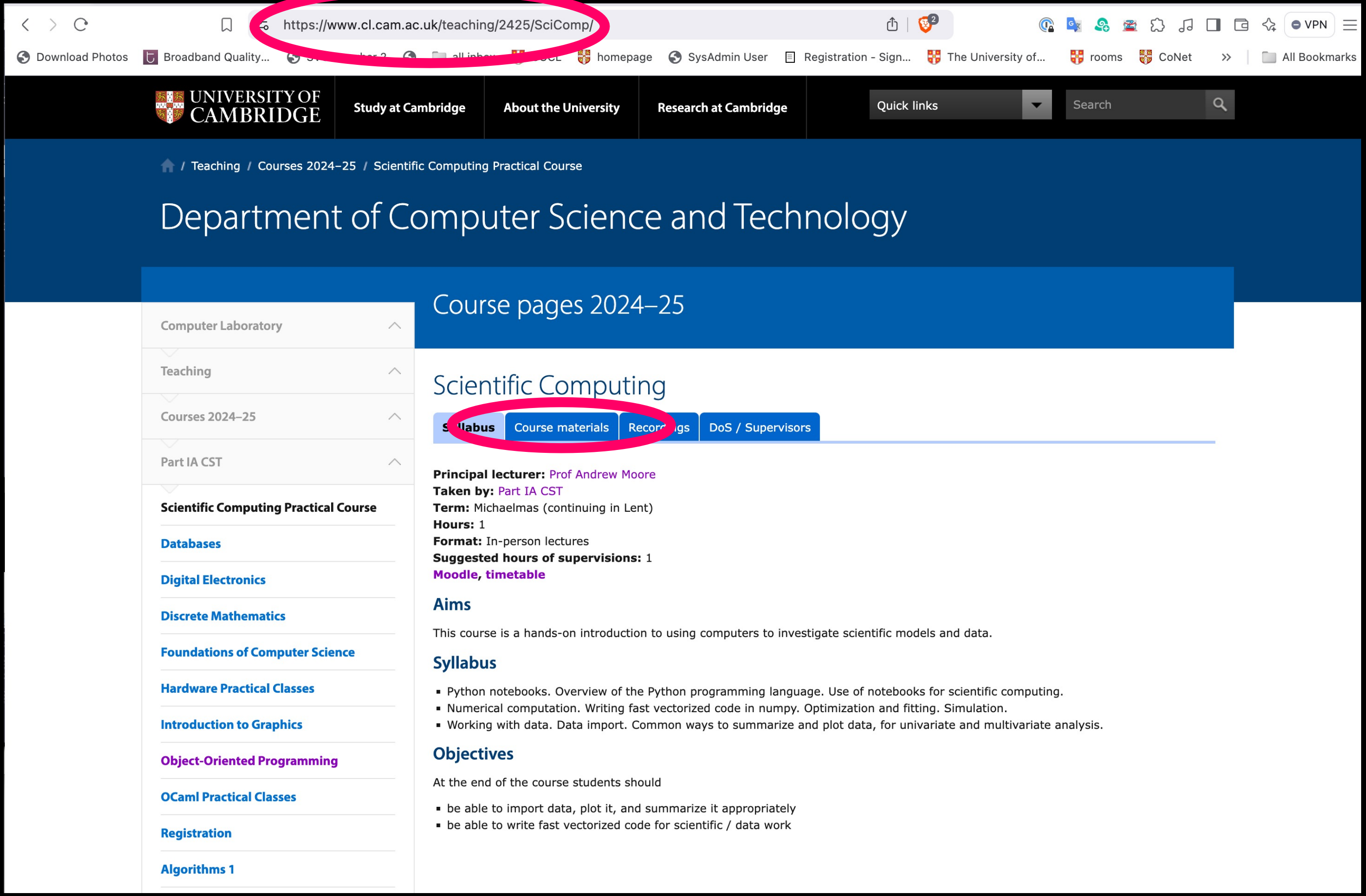
update to experiment 1

forgotten import

after you've finished



```
imports

utility functions

run-once setup code
functions that implement
    your solutions

submit solutions to
    autograder
```



Teaching / Courses 2024–25 / Scientific Computing Practical Course

Department of Computer Science and Technology

Course pages 2024–25

Scientific Computing

Principal lecturer: Prof Andrew Moore
Taken by: Part IA CST
Term: Michaelmas (continuing in Lent)
Hours: 1
Format: In-person lectures
Suggested hours of supervisions: 1
Moodle, **timetable**

Aims

This course is a hands-on introduction to using computers to investigate scientific models and data.

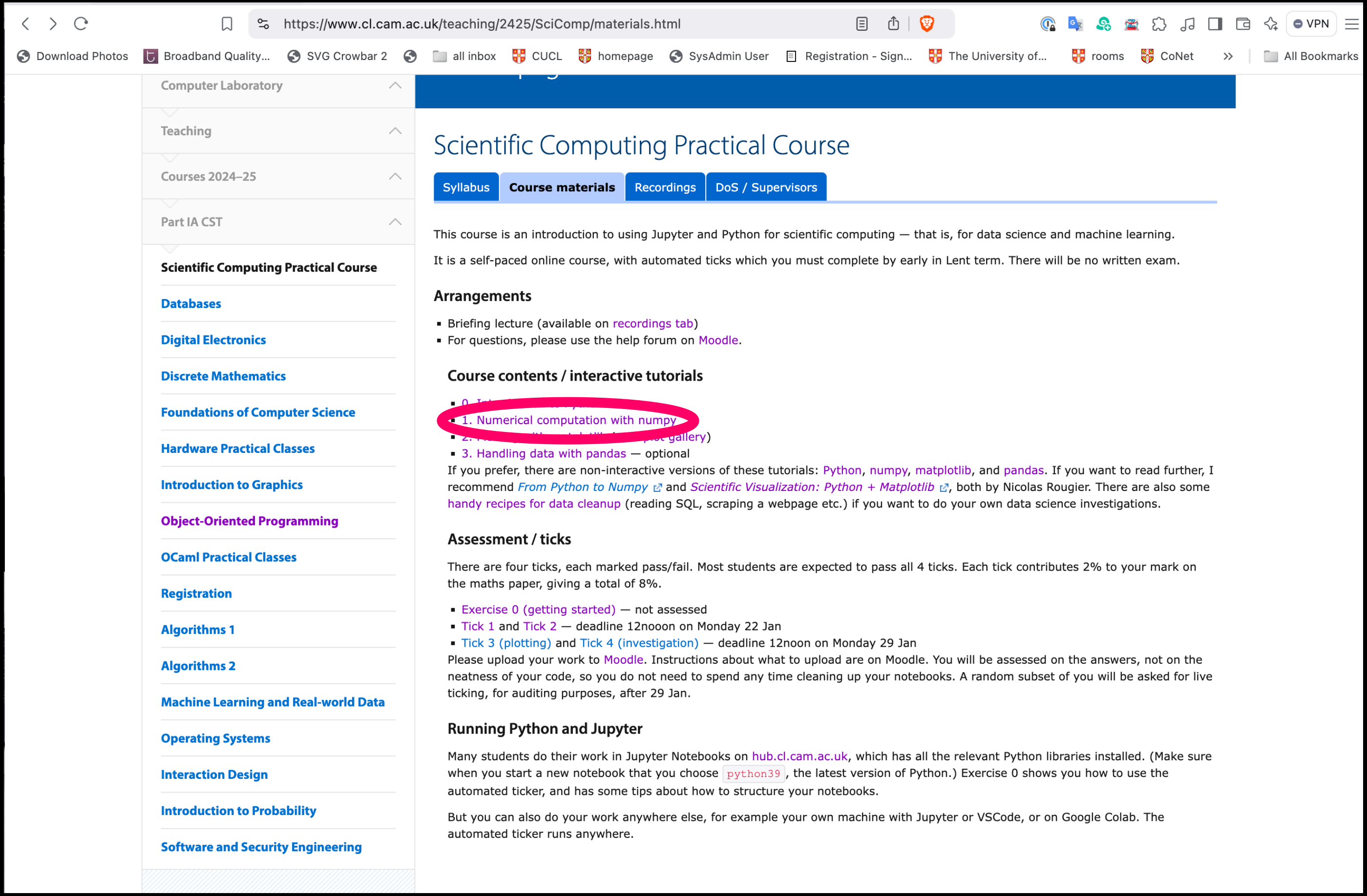
Syllabus

- Python notebooks. Overview of the Python programming language. Use of notebooks for scientific computing.
- Numerical computation. Writing fast vectorized code in numpy. Optimization and fitting. Simulation.
- Working with data. Data import. Common ways to summarize and plot data, for univariate and multivariate analysis.

Objectives

At the end of the course students should

- be able to import data, plot it, and summarize it appropriately
- be able to write fast vectorized code for scientific / data work



Scientific Computing Practical Course

This course is an introduction to using Jupyter and Python for scientific computing — that is, for data science and machine learning.

It is a self-paced online course, with automated ticks which you must complete by early in Lent term. There will be no written exam.

Arrangements

- Briefing lecture (available on recordings tab)
- For questions, please use the help forum on Moodle.

Course contents / interactive tutorials

- 0. Int[illegible] Python
- 1. Numerical computation with numpy
- 2. Pl[illegible] with matplotlib ([illegible] plot gallery)
- 3. Handling data with pandas — optional

If you prefer, there are non-interactive versions of these tutorials: Python, numpy, matplotlib, and pandas. If you want to read further, I recommend *From Python to Numpy* and *Scientific Visualization: Python + Matplotlib*, both by Nicolas Rougier. There are also some handy recipes for data cleanup (reading SQL, scraping a webpage etc.) if you want to do your own data science investigations.

Assessment / ticks

There are four ticks, each marked pass/fail. Most students are expected to pass all 4 ticks. Each tick contributes 2% to your mark on the maths paper, giving a total of 8%.

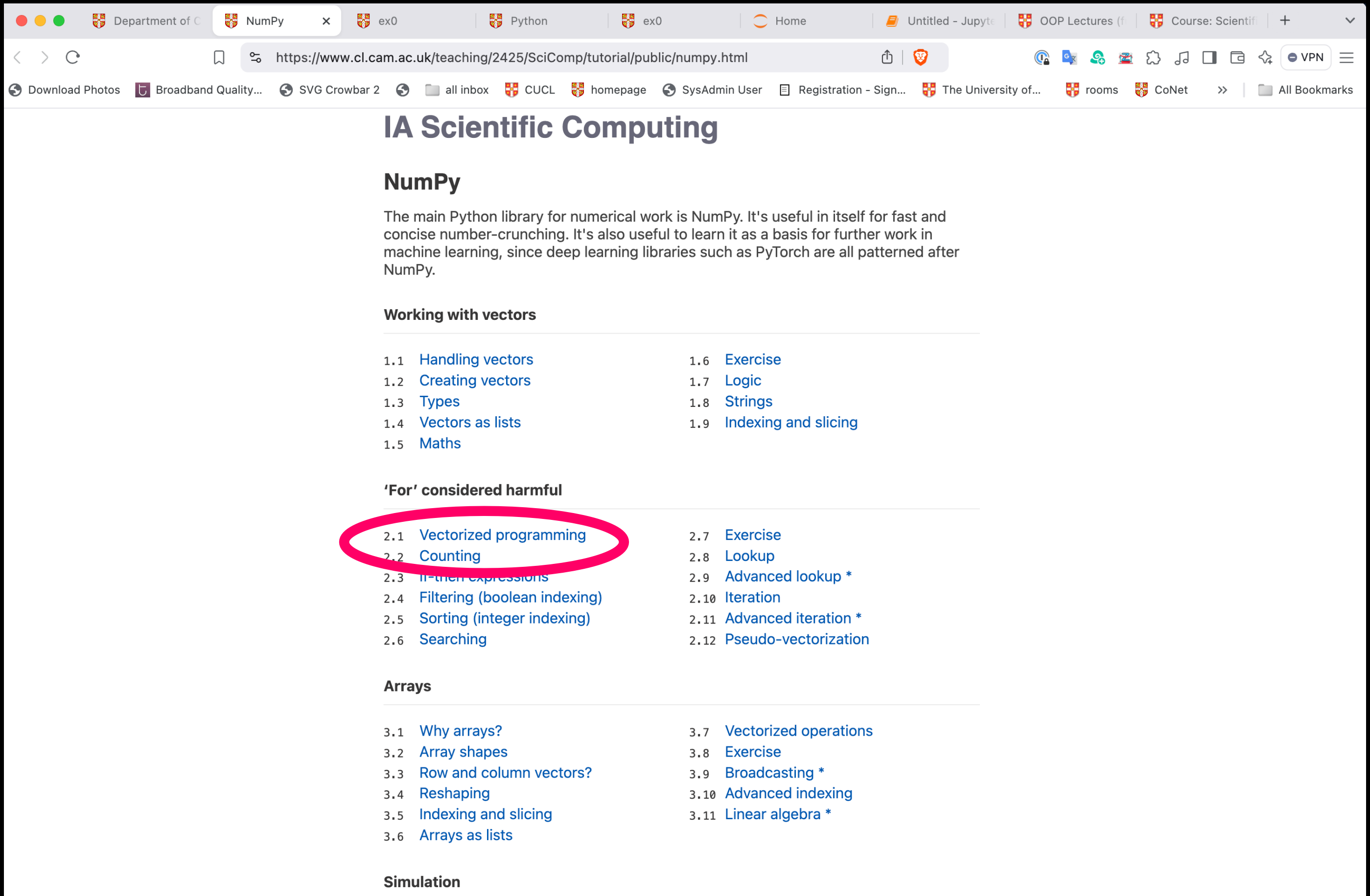
- Exercise 0 (getting started) — not assessed
- Tick 1 and Tick 2 — deadline 12nooon on Monday 22 Jan
- Tick 3 (plotting) and Tick 4 (investigation) — deadline 12noon on Monday 29 Jan

Please upload your work to Moodle. Instructions about what to upload are on Moodle. You will be assessed on the answers, not on the neatness of your code, so you do not need to spend any time cleaning up your notebooks. A random subset of you will be asked for live ticking, for auditing purposes, after 29 Jan.

Running Python and Jupyter

Many students do their work in Jupyter Notebooks on hub.cl.cam.ac.uk, which has all the relevant Python libraries installed. (Make sure when you start a new notebook that you choose `python39`, the latest version of Python.) Exercise 0 shows you how to use the automated ticker, and has some tips about how to structure your notebooks.

But you can also do your work anywhere else, for example your own machine with Jupyter or VSCode, or on Google Colab. The automated ticker runs anywhere.



IA Scientific Computing

NumPy

The main Python library for numerical work is NumPy. It's useful in itself for fast and concise number-crunching. It's also useful to learn it as a basis for further work in machine learning, since deep learning libraries such as PyTorch are all patterned after NumPy.

Working with vectors

1.1 Handling vectors
1.2 Creating vectors
1.3 Types
1.4 Vectors as lists
1.5 Maths
1.6 Exercise
1.7 Logic
1.8 Strings
1.9 Indexing and slicing

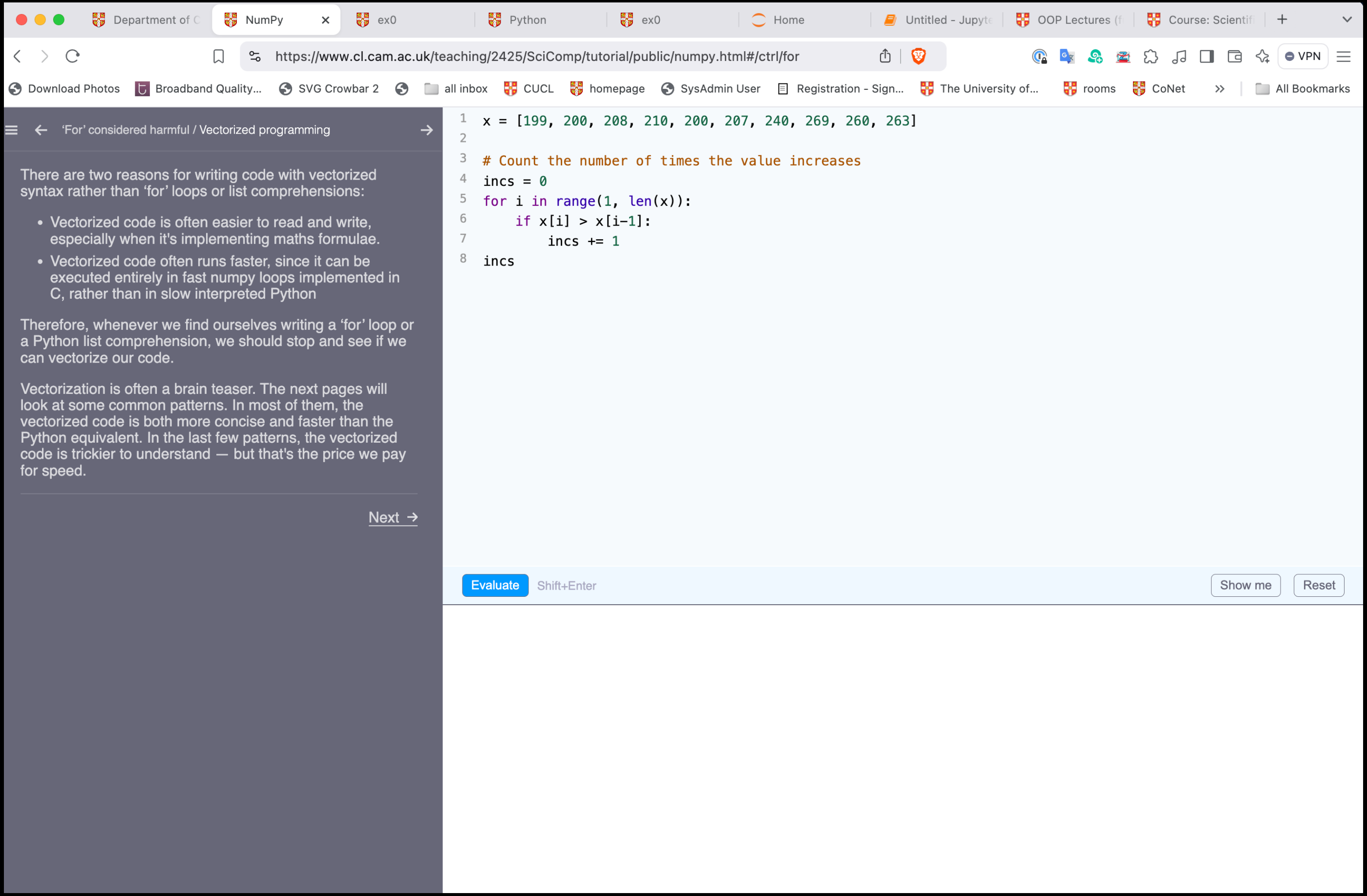
‘For’ considered harmful

2.1 Vectorized programming
2.2 Counting
2.3 If-then expressions
2.4 Filtering (boolean indexing)
2.5 Sorting (integer indexing)
2.6 Searching
2.7 Exercise
2.8 Lookup
2.9 Advanced lookup *
2.10 Iteration
2.11 Advanced iteration *
2.12 Pseudo-vectorization

Arrays

3.1 Why arrays?
3.2 Array shapes
3.3 Row and column vectors?
3.4 Reshaping
3.5 Indexing and slicing
3.6 Arrays as lists
3.7 Vectorized operations
3.8 Exercise
3.9 Broadcasting *
3.10 Advanced indexing
3.11 Linear algebra *

Simulation



'For' considered harmful / Vectorized programming

There are two reasons for writing code with vectorized syntax rather than 'for' loops or list comprehensions:

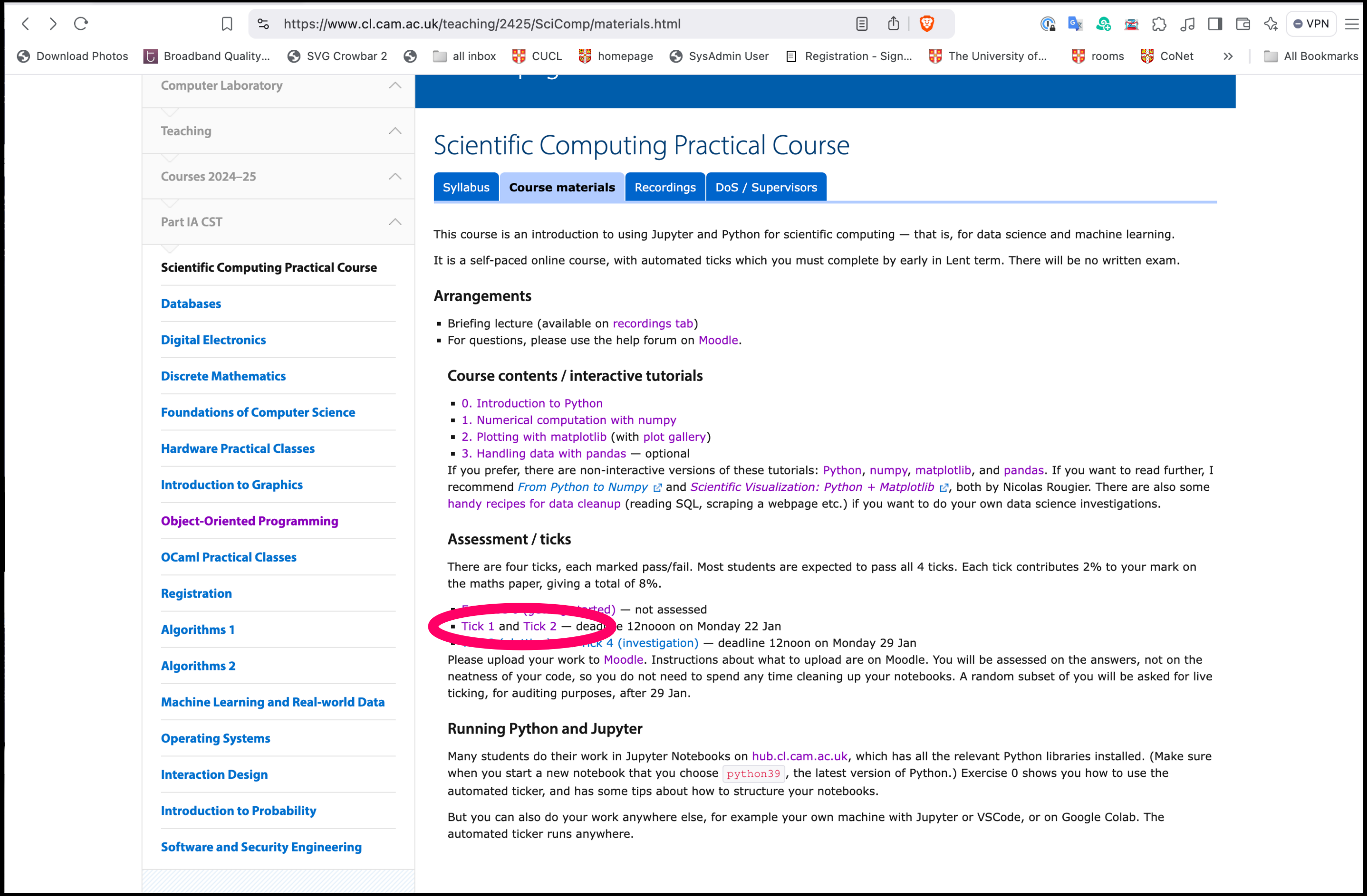
- Vectorized code is often easier to read and write, especially when it's implementing maths formulae.
- Vectorized code often runs faster, since it can be executed entirely in fast numpy loops implemented in C, rather than in slow interpreted Python

Therefore, whenever we find ourselves writing a 'for' loop or a Python list comprehension, we should stop and see if we can vectorize our code.

Vectorization is often a brain teaser. The next pages will look at some common patterns. In most of them, the vectorized code is both more concise and faster than the Python equivalent. In the last few patterns, the vectorized code is trickier to understand — but that's the price we pay for speed.

Next →

```
x = [199, 200, 208, 210, 200, 207, 240, 269, 260, 263]

# Count the number of times the value increases
incs = 0
for i in range(1, len(x)):
    if x[i] > x[i-1]:
        incs += 1
incs
```



Scientific Computing Practical Course

This course is an introduction to using Jupyter and Python for scientific computing — that is, for data science and machine learning.

It is a self-paced online course, with automated ticks which you must complete by early in Lent term. There will be no written exam.

Arrangements

- Briefing lecture (available on recordings tab)
- For questions, please use the help forum on Moodle.

Course contents / interactive tutorials

- 0. Introduction to Python
- 1. Numerical computation with numpy
- 2. Plotting with matplotlib (with plot gallery)
- 3. Handling data with pandas — optional

If you prefer, there are non-interactive versions of these tutorials: Python, numpy, matplotlib, and pandas. If you want to read further, I recommend *From Python to Numpy* and *Scientific Visualization: Python + Matplotlib*, both by Nicolas Rougier. There are also some handy recipes for data cleanup (reading SQL, scraping a webpage etc.) if you want to do your own data science investigations.

Assessment / ticks

There are four ticks, each marked pass/fail. Most students are expected to pass all 4 ticks. Each tick contributes 2% to your mark on the maths paper, giving a total of 8%.

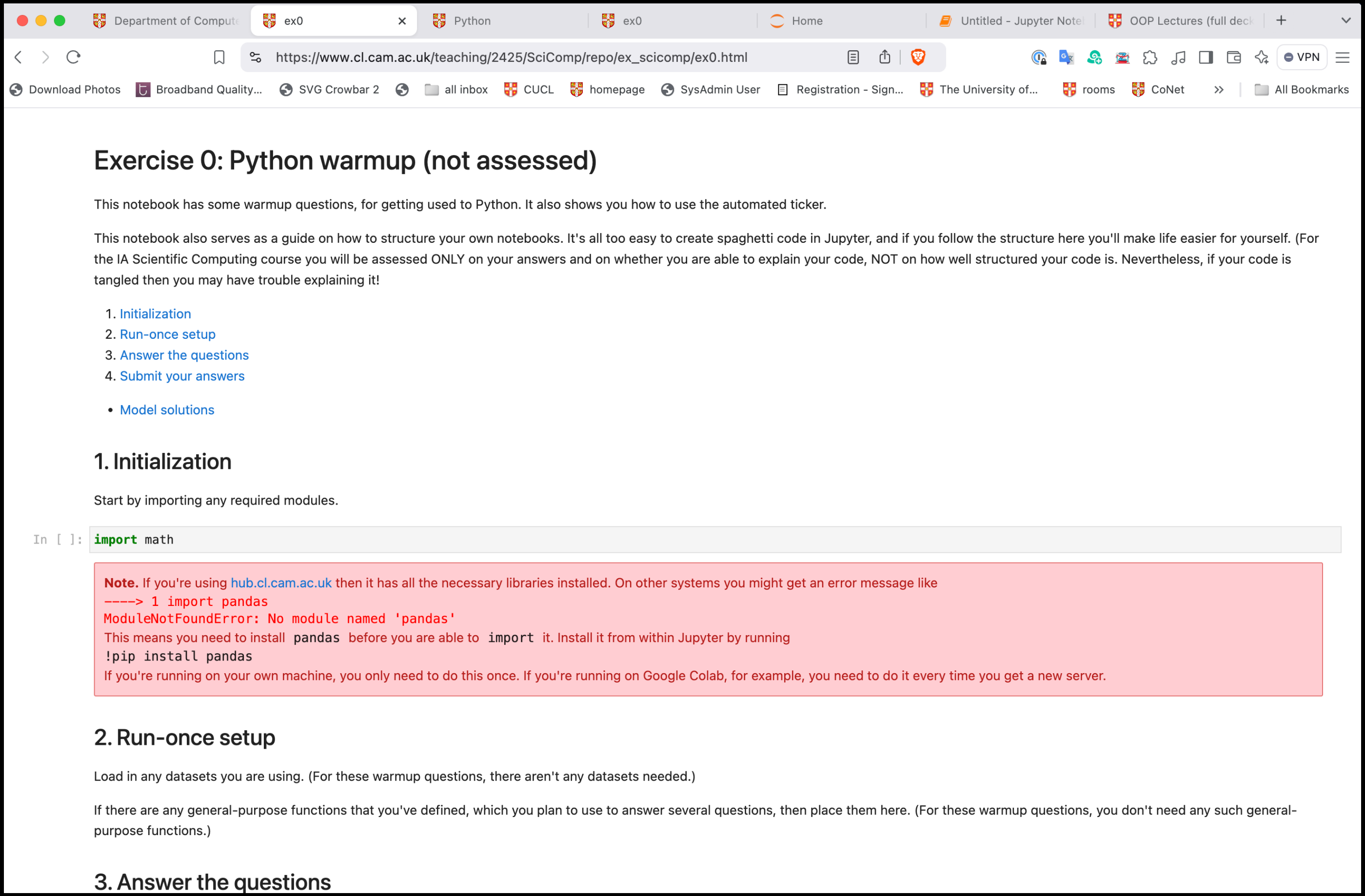
- Exercise 0 (getting started) — not assessed
- Tick 1 and Tick 2 — deadline 12nooon on Monday 22 Jan
- Tick 3 (plotting) and Tick 4 (investigation) — deadline 12noon on Monday 29 Jan

Please upload your work to Moodle. Instructions about what to upload are on Moodle. You will be assessed on the answers, not on the neatness of your code, so you do not need to spend any time cleaning up your notebooks. A random subset of you will be asked for live ticking, for auditing purposes, after 29 Jan.

Running Python and Jupyter

Many students do their work in Jupyter Notebooks on hub.cl.cam.ac.uk, which has all the relevant Python libraries installed. (Make sure when you start a new notebook that you choose `python39`, the latest version of Python.) Exercise 0 shows you how to use the automated ticker, and has some tips about how to structure your notebooks.

But you can also do your work anywhere else, for example your own machine with Jupyter or VSCode, or on Google Colab. The automated ticker runs anywhere.



Exercise 0: Python warmup (not assessed)

This notebook has some warmup questions, for getting used to Python. It also shows you how to use the automated ticker.

This notebook also serves as a guide on how to structure your own notebooks. It's all too easy to create spaghetti code in Jupyter, and if you follow the structure here you'll make life easier for yourself. (For the IA Scientific Computing course you will be assessed ONLY on your answers and on whether you are able to explain your code, NOT on how well structured your code is. Nevertheless, if your code is tangled then you may have trouble explaining it!

1. Initialization
2. Run-once setup
3. Answer the questions
4. Submit your answers

- Model solutions

1. Initialization

Start by importing any required modules.

```
In [ ]: import math
```

Note. If you're using hub.cl.cam.ac.uk then it has all the necessary libraries installed. On other systems you might get an error message like

```
----> 1 import pandas
ModuleNotFoundError: No module named 'pandas'
```

This means you need to install `pandas` before you are able to `import` it. Install it from within Jupyter by running

```
!pip install pandas
```

If you're running on your own machine, you only need to do this once. If you're running on Google Colab, for example, you need to do it every time you get a new server.

2. Run-once setup

Load in any datasets you are using. (For these warmup questions, there aren't any datasets needed.)

If there are any general-purpose functions that you've defined, which you plan to use to answer several questions, then place them here. (For these warmup questions, you don't need any such general-purpose functions.)

3. Answer the questions

TUTORIALS

0. Programming in Python
   language quirks

1. Numerical computation
   `numpy`

2. Plotting data
   `matplotlib`

3. Working with data
   `pandas`

A. Data scraping recipes

ASSESSMENT

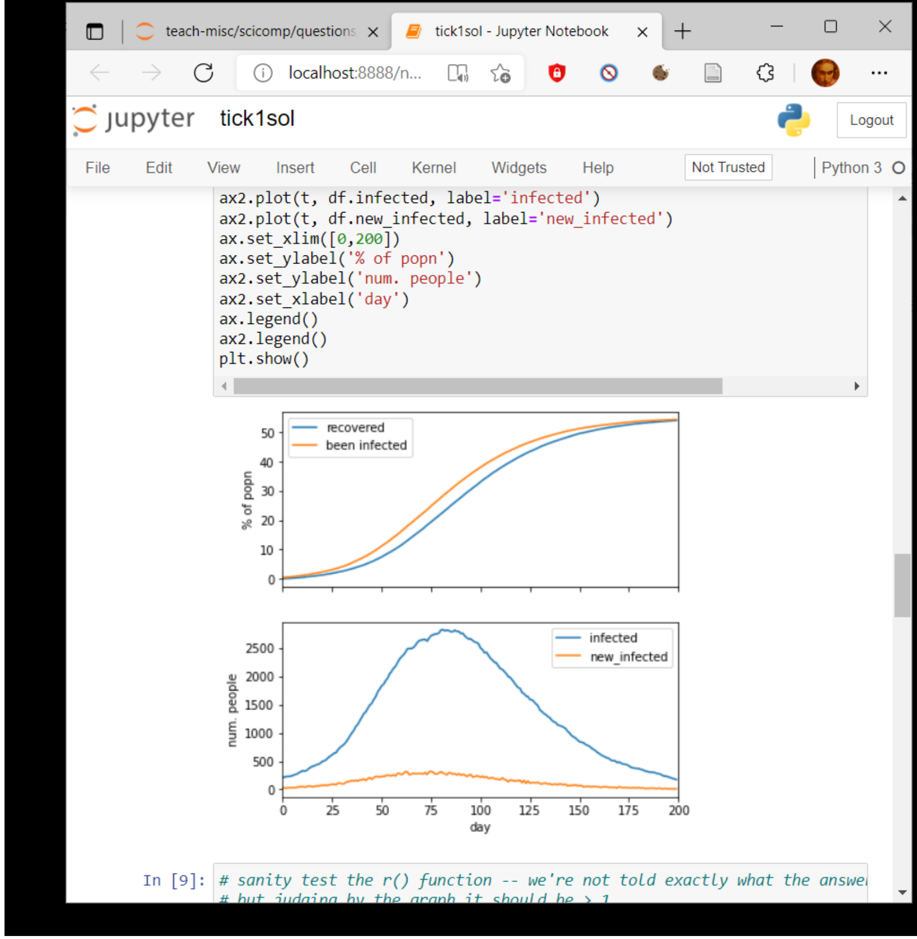
(maths paper mark = 92% exam + 8% Scientific Computing ticks)

No written exam

Four ticks, each marked pass/fail
Ticks 1 and 2: pass the autograder & submit notebook by 27 Jan
Ticks 3 and 4: submit pdfs and notebook by 3 Feb

Some of you will have a viva.

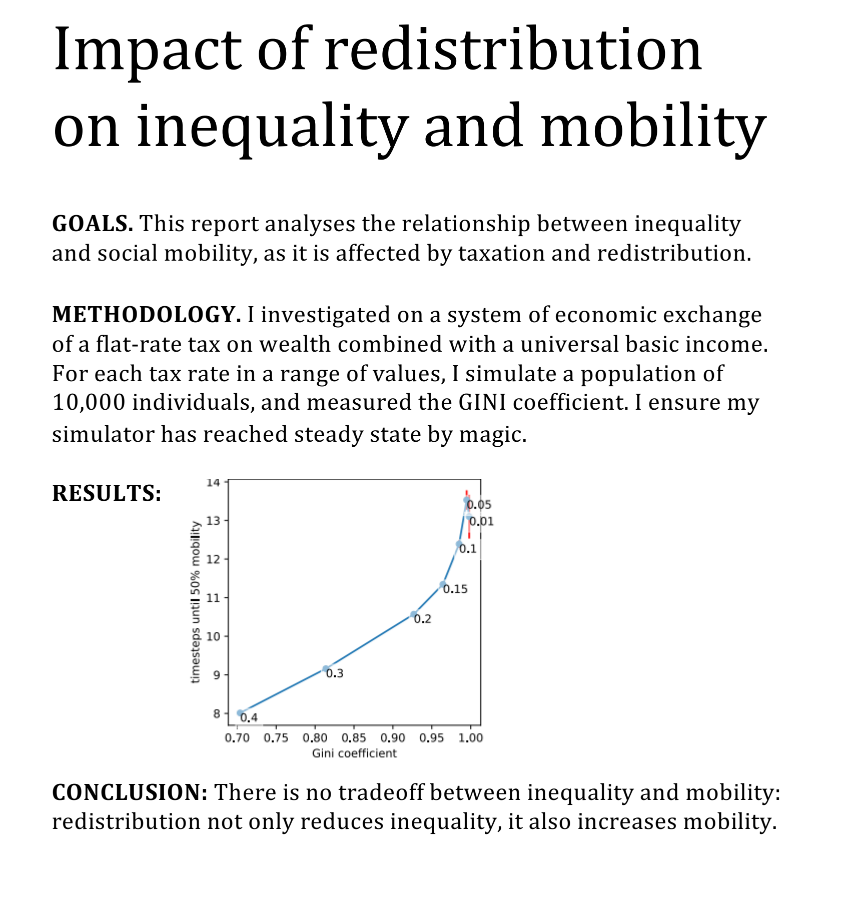
Tick 1,2: Econo-physics simulator (with answers checked by autograder)



```
ax2.plot(t, df.infected, label='infected')
ax2.plot(t, df.new_infected, label='new_infected')
ax.set_xlim([0,200])
ax.set_ylabel('% of popn')
ax2.set_ylabel('num. people')
ax2.set_xlabel('day')
ax.legend()
ax2.legend()
plt.show()
```

```
In [9]: # sanity test the r() function -- we're not told exactly what the answe
        # but judging by the graph it should be > 1
```

Tick 3: plots
Tick 4: One-page scientific report



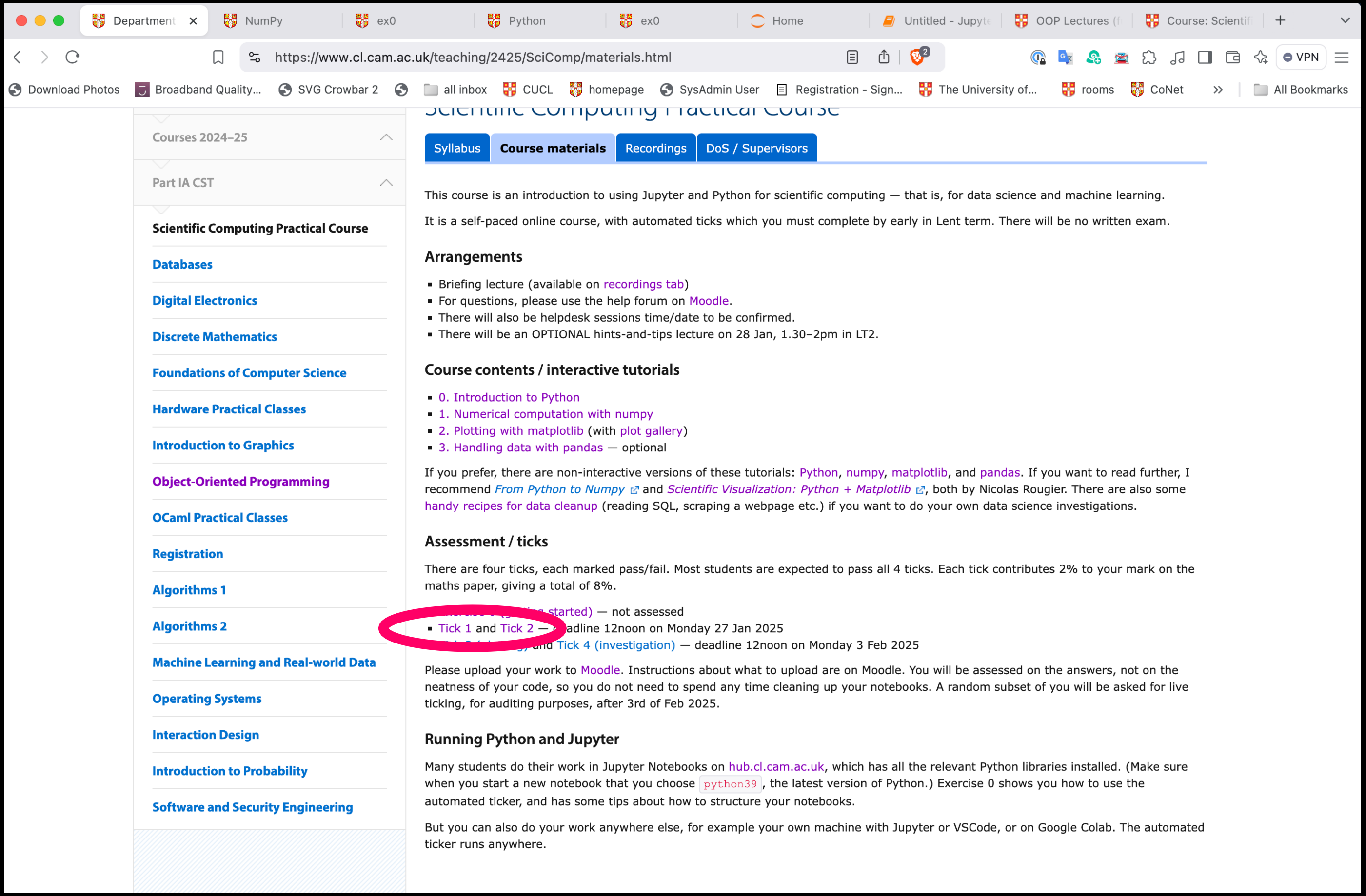
Impact of redistribution on inequality and mobility

GOALS. This report analyses the relationship between inequality and social mobility, as it is affected by taxation and redistribution.

METHODOLOGY. I investigated on a system of economic exchange of a flat-rate tax on wealth combined with a universal basic income. For each tax rate in a range of values, I simulate a population of 10,000 individuals, and measured the GINI coefficient. I ensure my simulator has reached steady state by magic.

RESULTS:

CONCLUSION: There is no tradeoff between inequality and mobility: redistribution not only reduces inequality, it also increases mobility.



Scientific Computing Practical Course

This course is an introduction to using Jupyter and Python for scientific computing — that is, for data science and machine learning.

It is a self-paced online course, with automated ticks which you must complete by early in Lent term. There will be no written exam.

Arrangements

- Briefing lecture (available on recordings tab)
- For questions, please use the help forum on Moodle.
- There will also be helpdesk sessions time/date to be confirmed.
- There will be an OPTIONAL hints-and-tips lecture on 28 Jan, 1.30–2pm in LT2.

Course contents / interactive tutorials

- 0. Introduction to Python
- 1. Numerical computation with numpy
- 2. Plotting with matplotlib (with plot gallery)
- 3. Handling data with pandas — optional

If you prefer, there are non-interactive versions of these tutorials: Python, numpy, matplotlib, and pandas. If you want to read further, I recommend *From Python to Numpy* and *Scientific Visualization: Python + Matplotlib*, both by Nicolas Rougier. There are also some handy recipes for data cleanup (reading SQL, scraping a webpage etc.) if you want to do your own data science investigations.

Assessment / ticks

There are four ticks, each marked pass/fail. Most students are expected to pass all 4 ticks. Each tick contributes 2% to your mark on the maths paper, giving a total of 8%.

- Exercise 0 (getting started) — not assessed
- Tick 1 and Tick 2 — deadline 12noon on Monday 27 Jan 2025
- Tick 3 (plotting) and Tick 4 (investigation) — deadline 12noon on Monday 3 Feb 2025

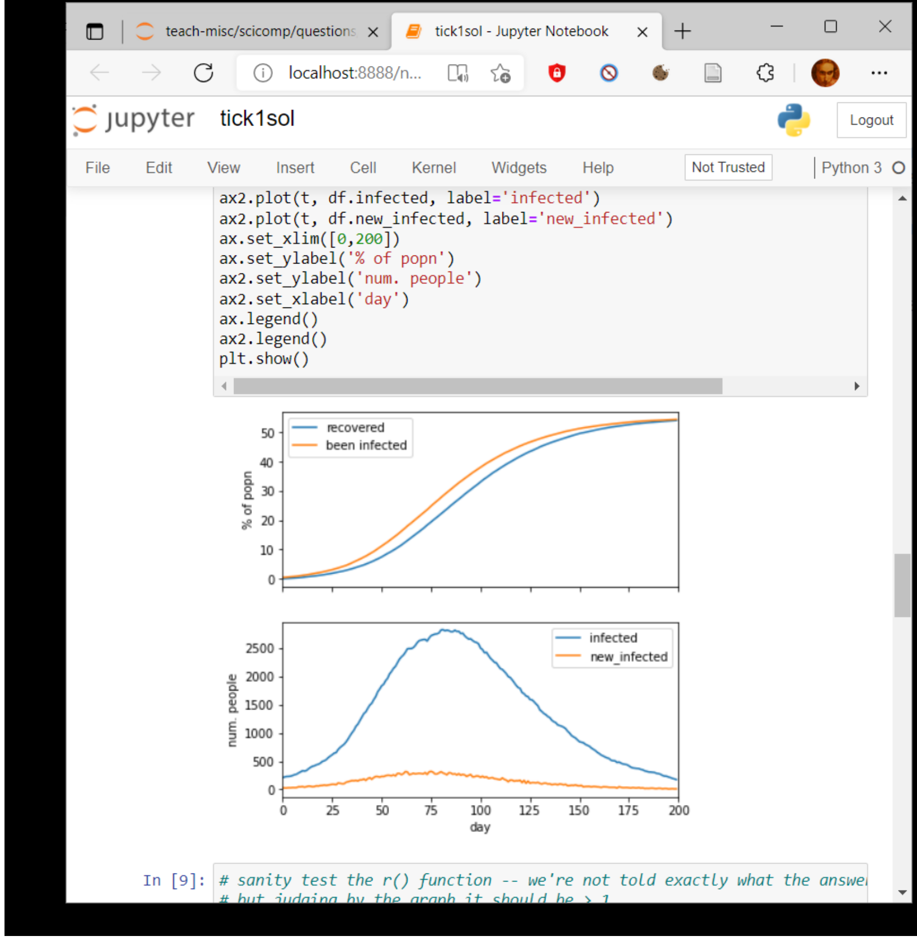
Please upload your work to Moodle. Instructions about what to upload are on Moodle. You will be assessed on the answers, not on the neatness of your code, so you do not need to spend any time cleaning up your notebooks. A random subset of you will be asked for live ticking, for auditing purposes, after 3rd of Feb 2025.

Running Python and Jupyter

Many students do their work in Jupyter Notebooks on hub.cl.cam.ac.uk, which has all the relevant Python libraries installed. (Make sure when you start a new notebook that you choose `python39`, the latest version of Python.) Exercise 0 shows you how to use the automated ticker, and has some tips about how to structure your notebooks.

But you can also do your work anywhere else, for example your own machine with Jupyter or VSCode, or on Google Colab. The automated ticker runs anywhere.

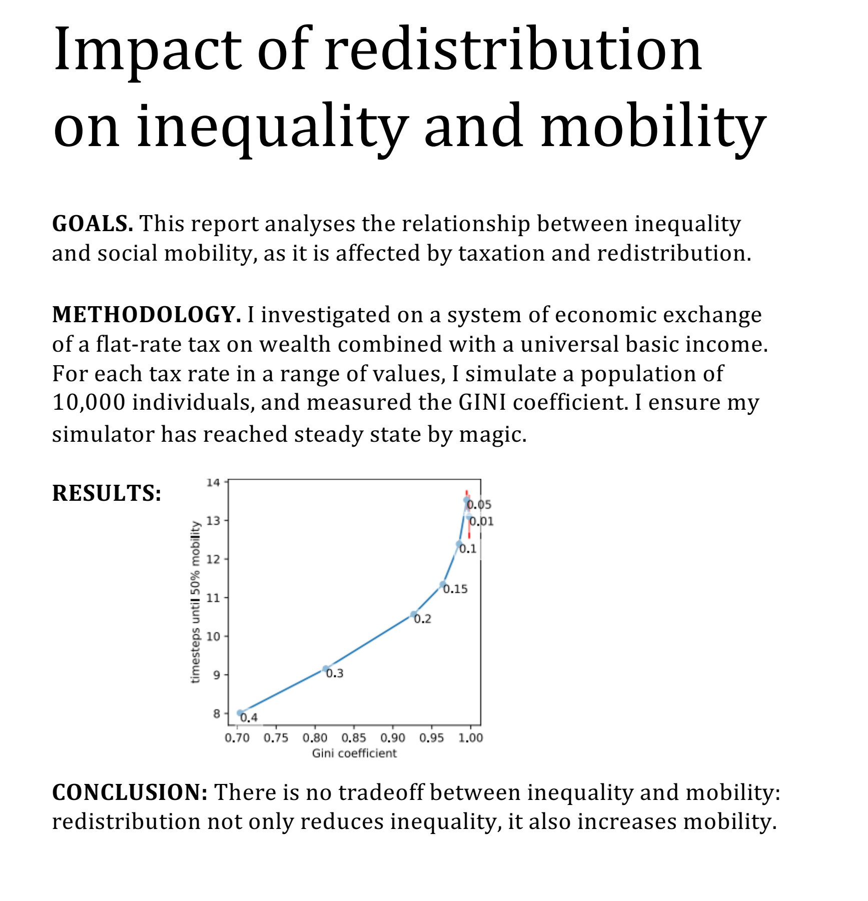
Tick 1,2: Econo-physics simulator (with answers checked by autograder)



```
ax2.plot(t, df.infected, label='infected')
ax2.plot(t, df.new_infected, label='new_infected')
ax.set_xlim([0,200])
ax.set_ylabel('% of popn')
ax2.set_ylabel('num. people')
ax2.set_xlabel('day')
ax.legend()
ax2.legend()
plt.show()
```

```
In [9]: # sanity test the r() function -- we're not told exactly what the answe
        # but judging by the graph it should be > 1
```

Tick 3: plots
Tick 4: One-page scientific report


14
13
12
11
10
9
8
timesteps until 50% mobility
0.70
0.75
0.80
0.85
0.90
0.95
1.00
Gini coefficient
0.05
0.01
0.1
0.15
0.2
0.3
0.4

Impact of redistribution on inequality and mobility

GOALS. This report analyses the relationship between inequality and social mobility, as it is affected by taxation and redistribution.

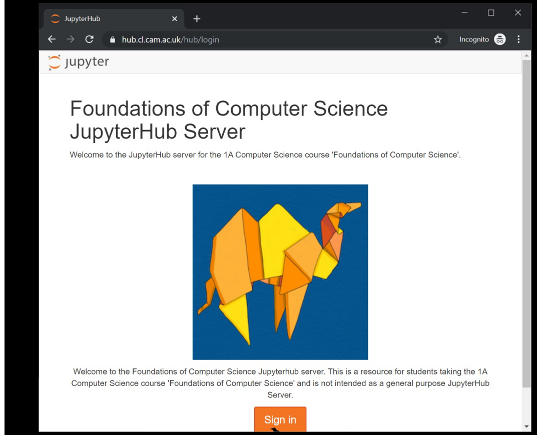
METHODOLOGY. I investigated on a system of economic exchange of a flat-rate tax on wealth combined with a universal basic income. For each tax rate in a range of values, I simulate a population of 10,000 individuals, and measured the GINI coefficient. I ensure my simulator has reached steady state by magic.

RESULTS:

CONCLUSION: There is no tradeoff between inequality and mobility: redistribution not only reduces inequality, it also increases mobility.

The autograder will run wherever you run Python3

hub.cl.cam.ac.uk

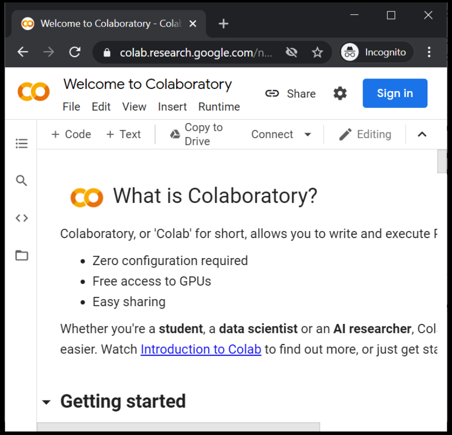


Foundations of Computer Science JupyterHub Server

Welcome to the JupyterHub server for the 1A Computer Science course 'Foundations of Computer Science'.

Welcome to the Foundations of Computer Science Jupyterhub server. This is a resource for students taking the 1A Computer Science course 'Foundations of Computer Science' and is not intended as a general purpose JupyterHub Server.

Google colab



What is Colaboratory?

Colaboratory, or 'Colab' for short, allows you to write and execute F

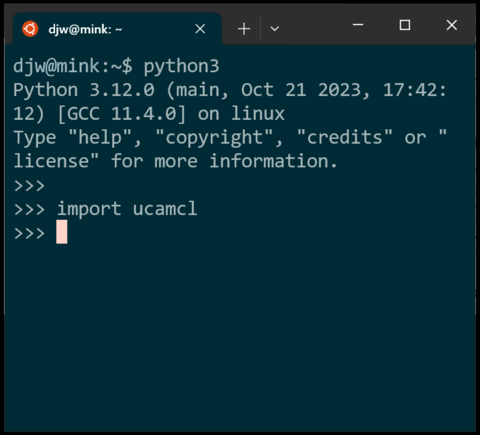
- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Col easier. Watch Introduction to Colab to find out more, or just get sta

Getting started

VSCode

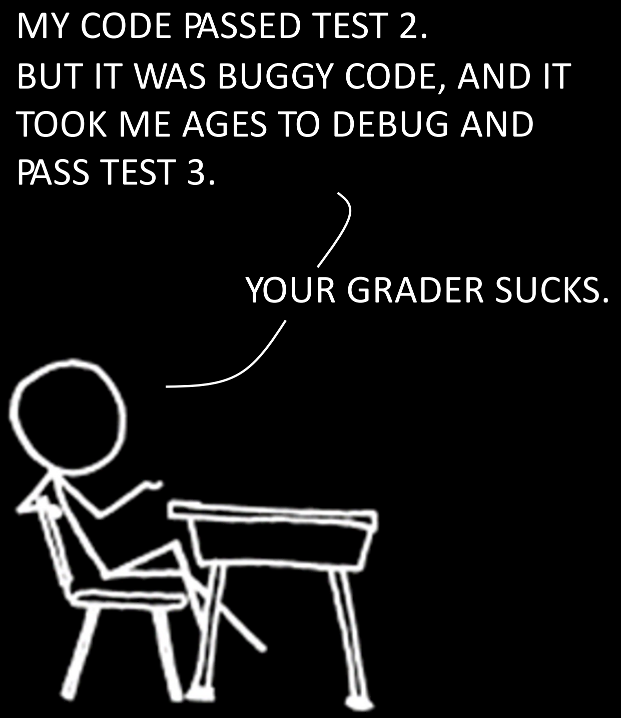
Command line



```
djw@mink:~$ python3
Python 3.12.0 (main, Oct 21 2023, 17:42:
12) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "
license" for more information.
>>>
>>> import ucamcl
>>> 
```

Help and support

- Moodle help forum
- Helpdesk sessions early in Lent term
- Optional hints-and-tips lecture early in Lent term


MY CODE PASSED TEST 2.
BUT IT WAS BUGGY CODE, AND IT TOOK ME AGES TO DEBUG AND PASS TEST 3.
YOUR GRADER SUCKS.

Scientific computing isn't about meeting requirements, **it's about discovery.**

✦ chart your own path ✦ write your own tests ✦ invent a few small test cases ✦ work them out with pen and paper ✦ make sure your code agrees

Can I use ChatGPT?

HM, HOW DO SCINTIC COMPUTING PROBLEM?
HM, HOW DO' E SICI COMUTING PR₀BLEM?
HM, HOW DO SOLVE THIS ENTHSC SCOIBUTICS SCIENTIC C COMPTENG = ?
WOW, AI IS LIKE A TUTOR!!
WOW, - Chat G6F?) - - Chat GG(A) - ... ... - Chat GSP) - ... ...-(GF?) + 4t
THIS = HOMEWORK LIKE A TUTOR!!

Can I use ChatGPT?

Yes, feel free.

Can I use ChatGPT to save me time and effort?

Unlikely.

Can I use ChatGPT to sharpen my thinking?

Yes !!!👍🔥👍🔥👍