Introduction to Computational Physics (PHYS105)

Lecture 1: Introducing Python and Jupyter

Carl Gwilliam

(C.Gwilliam@liverpool.ac.uk)

UNIVERSITY OF LIVERPOOL

Attendance Code: 821816

Organisation

- Lectures
  - Tuesday @ 10:00-11:00
  - In Thomson-Yates Lecture Theatre (Rm 104)
- Computer Classes
  - Monday @ 09:00--11:00
  - In CTL PC teaching centre (PCTC)
- Assessment
  - 100% coursework (no exam)
  - Exercises in computing classes
    - Several formative / week
    - One summative / week (equal weighting)
    - Released before lecture; deadline end of Mon
  - Note: this course is exempt from the default Uni extension policy to allow rapid feedback
    - All students: 1 day extension (i.e. end of Tues)
    - Support plan: additional 3 days (i.e. end of Fri)

- Staff
  - Prof Carl Gwilliam (Rm 315)
    - Module Coordinator
    - C.Gwilliam@liverpool.ac.uk

  - Prof Neil McCauley (Rm 310)
    - Academic Demonstrator
    - N.McCauley@liverpool.ac.uk

- Along with several postgrad demonstrators
  - For the computer labs





Ned **Brad** **Sophie** **Sarah** **Ibrahim**

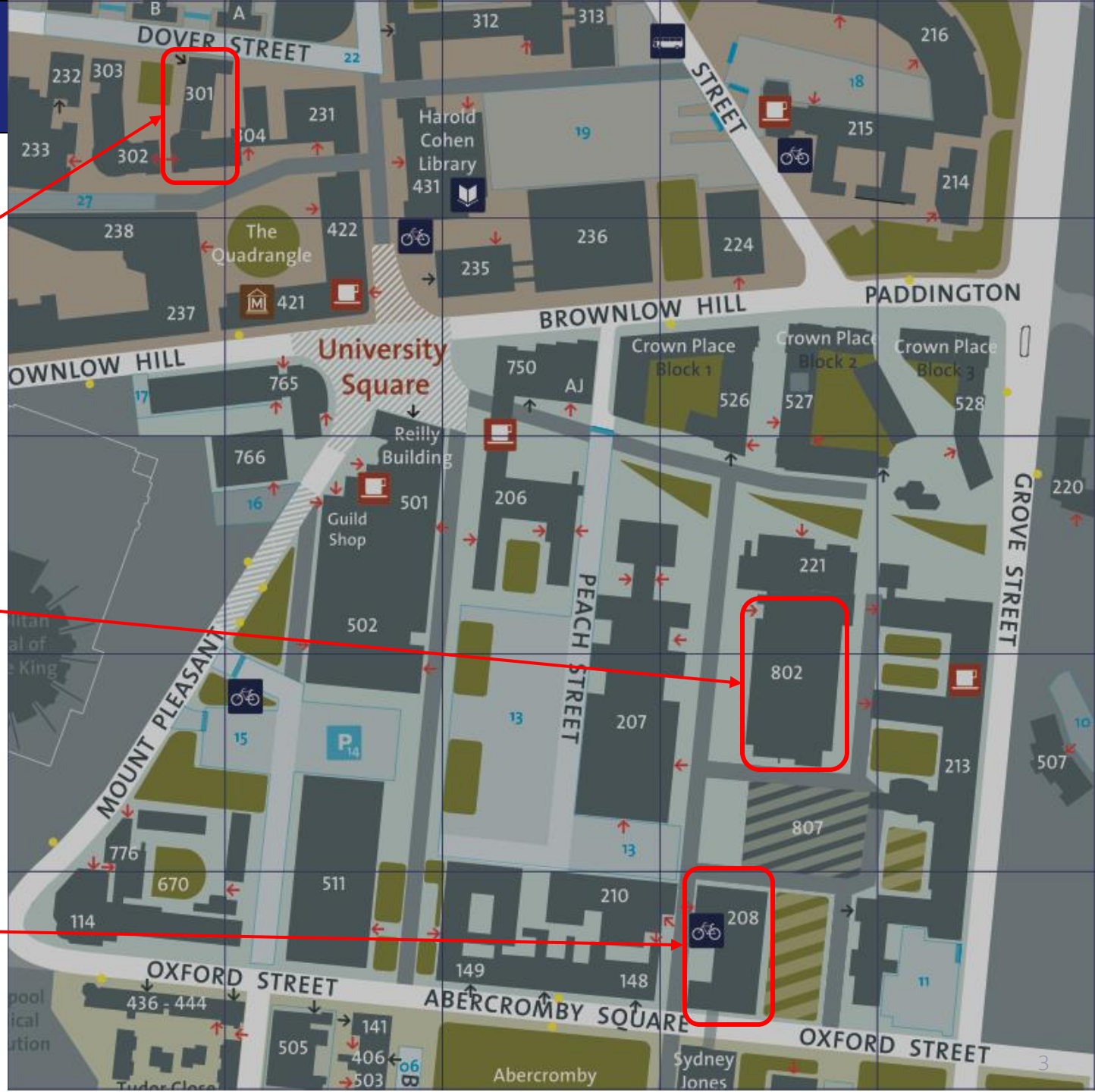
- Please get in touch at any point if need help

Locations

Lectures
- Thompson Yates Lecture Theatre
- Room 104

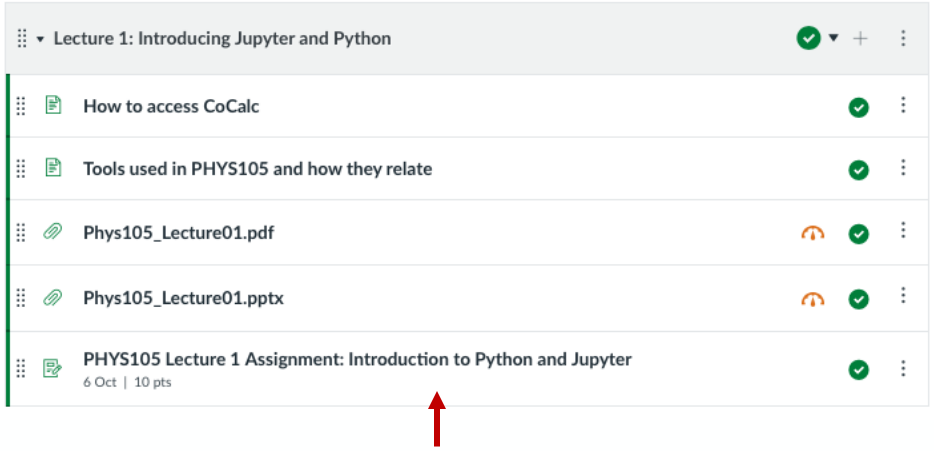
Computer Labs
- Central Teaching Lab
- PC Teaching Centre (PCTC)
  - All zones

Academic Offices
- Oliver Lodge Laboratory
- Room 315 & 310 (3rd floor)



Canvas

- All info is available on the course Canvas page
  - https://canvas.liverpool.ac.uk/courses/84543
- Click on links at top for course details
  - e.g. Aims & Syllabus, Assessment & Feedback, Contacts & Locations, Further Resource, etc
- Click on each image for that week's material
  - Slides will be released 24h before the lecture
  - Video recordings shortly after the lecture



- Submit work by clicking on assignment
  - And uploading saved PDF of notebook

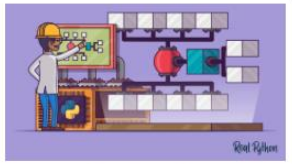
Introduction to Computational Physics (PHYS105)

Aims & Syllabus | Assessment & Feedback | Key Contacts & Locations | Further Resources | Core Values & Support

Please Note: This course is exempt from the default University extension policy (both standard and support-plan related) to allow rapid feedback between weekly assignments. For more information see Assessment and Feedback

Lecture 1

The basic structure of a python program and how to work with jupyter notebooks



Lecture 2

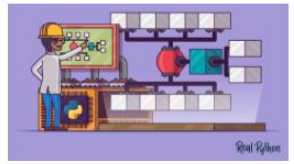
How to read in and store data, in particular using NumPy arrays

Lecture 3

How to write functions to manipulate data and use matplotlib to display the results

Lecture 4

How to execute pieces of code conditionally or multiple times



Lecture 5

How to retrieve user input, format output and read/write files.

Lecture 6

How to use the least-squares method to find the best fit of a function to data

Lecture 7

How to avoid, understand and fix various coding issues

Lecture 8

Evaluating equations using numerical techniques

Lecture 9

Using numerical techniques to model projectile motion with air resistance

Lecture 10

How to generate random numbers and use these to produce Monte Carlo models

Lecture 11

How to create your own python modules to allow your code to be reused

The end

We hope you enjoyed the course

Aims and outcomes

- The main aim of the course is to be able to create and run computer algorithms to solve real physical problems, breaking the problems down into steps amenable to such methods
  - Illustrating the insight into physics which can be obtained using computational methods

- In doing so, we will introduce
  - Basic computer algebra (using the python language)
  - Techniques for analysing and presenting data
  - Elementary numerical methods and “Monte Carlo” techniques

- By the end of the course you will be able to
  - Program and use simple (python) algorithms on a computer
  - Produce algorithms to solve simple physical problems.
  - Analyse (including numerically) and present physical data
  - Produce simple “Monte Carlo” (MC) models

- We will not assume any prior coding knowledge

What is your current programming level in general (not python specific)?

None - I have never done any computer programming before

0%

Basic - I can write a simple computer program

0%

Good - I am confident writing longer programs

0%

Excellent - I have significant programming experience and am confident writing detailed programs

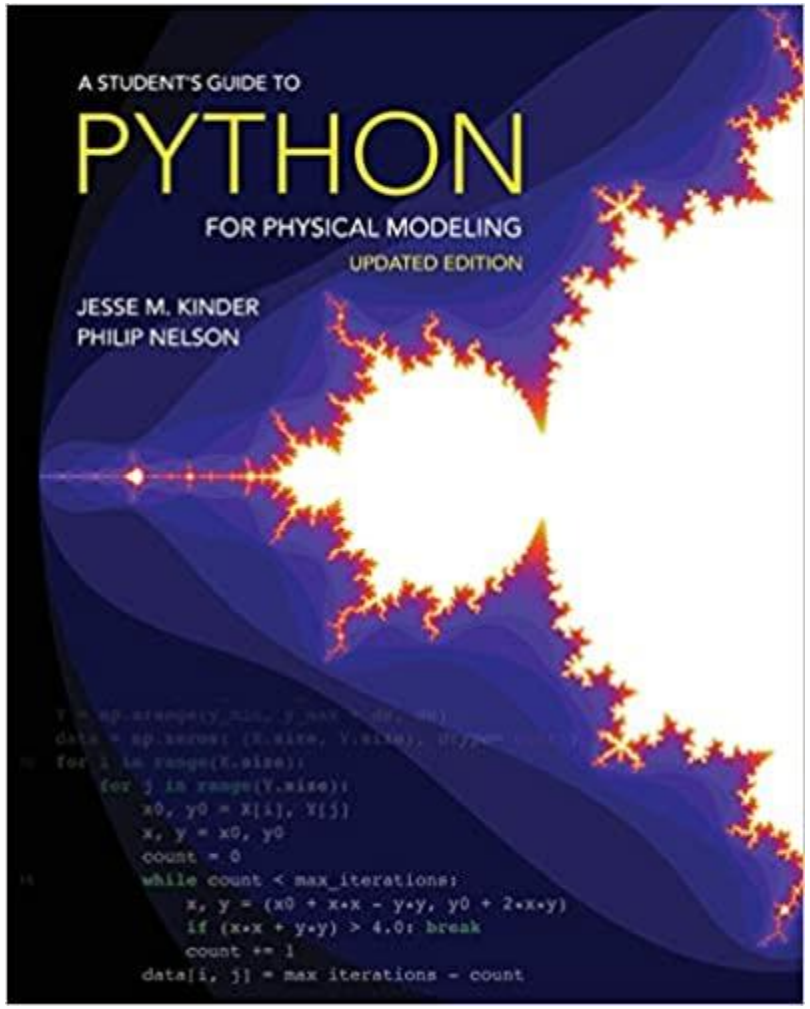
0%

Outline syllabus

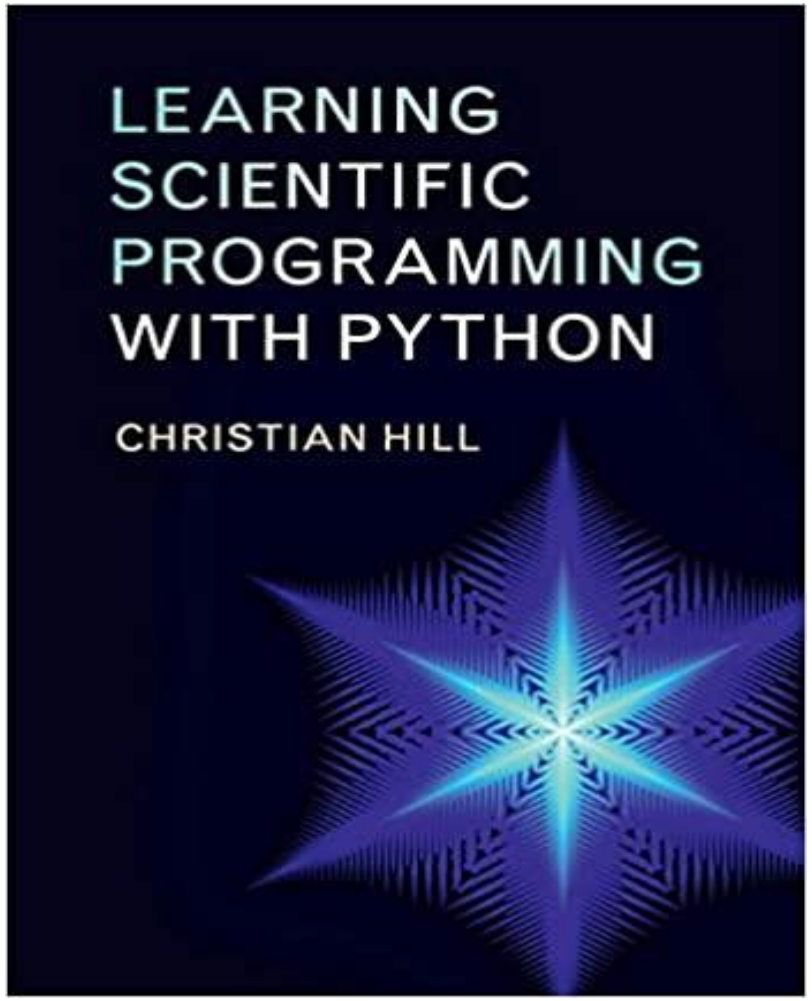
- The first part (a bit less than half) will develop your python skills
  - The second half will focus on practicing these by applying them to solve various physical problems
- Lecture 1
  - Python basics + Jupyter notebooks
- Lecture 2
  - Data structures
- Lecture 3
  - Functions and graphs
- Lecture 4
  - Looping and conditional execution
- Lecture 5
  - Input, Output and formatting
- Lecture 6
  - Fitting data
- Lecture 7
  - Good practice and debugging
- Lecture 8
  - Introduction to numerical methods
- Lecture 9
  - Further numeric techniques (projectile motion)
- Lecture 10
  - Monte Carlo methods
- Lecture 11
  - Python modules + fitting revisited

Recommended Textbooks

- A Student's Guide to Python for Physical Modelling - Kinder and Nelson
  - Princeton University Press



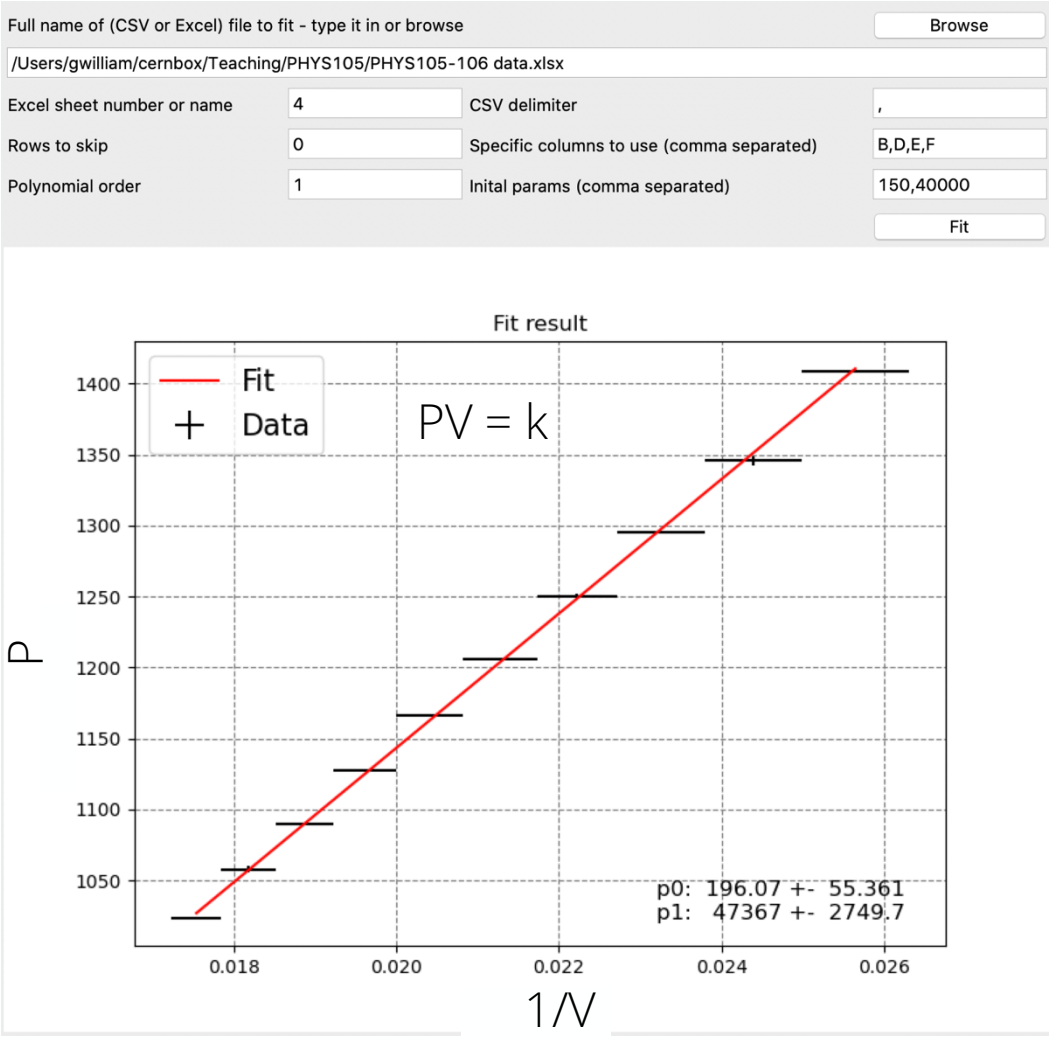
- Learning Scientific Programming with Python - Hill
  - Cambridge University Press



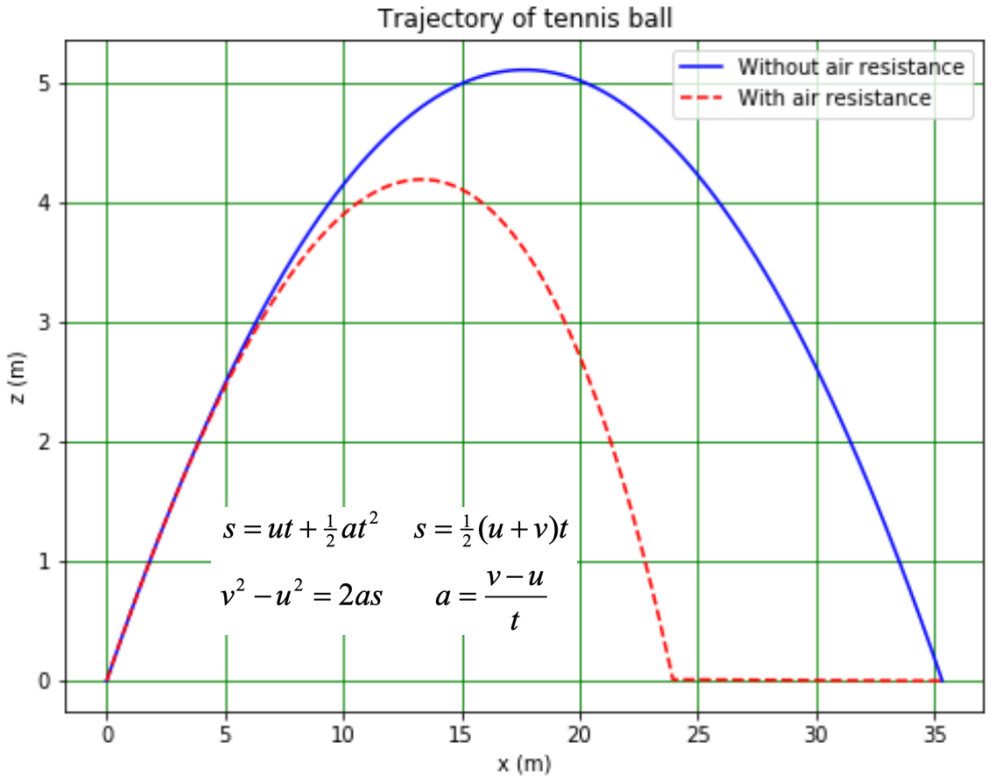
Why do I need to program?

Many cases where this will be essential in your physics degree

- Analyse + visualising experimental data
  - E.g. fitting data from Boyle's law experiment



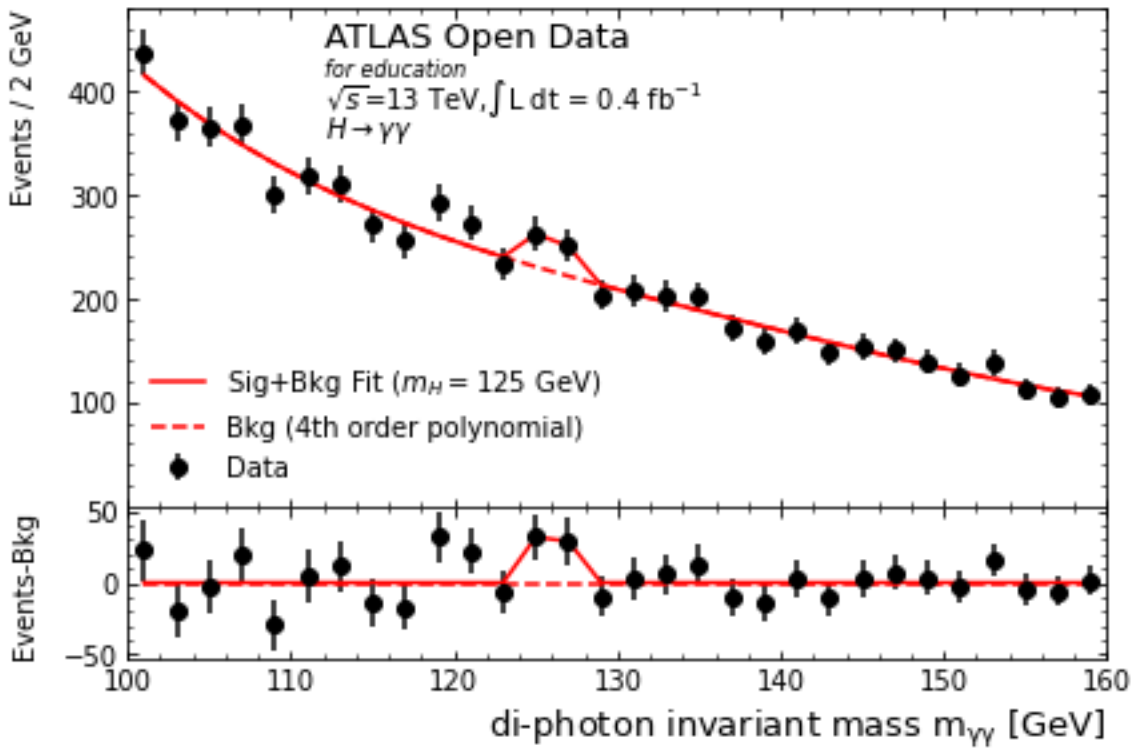
- Calculating results with no analytic solution
  - E.g distance travelled by tennis ball thrown at 20 m/s at an angle of 30° to horizontal
    - We know how to solve this using equations of motion, but result doesn't agree with real life
    - Need to include air resistance → numerical



Why do I need to program?

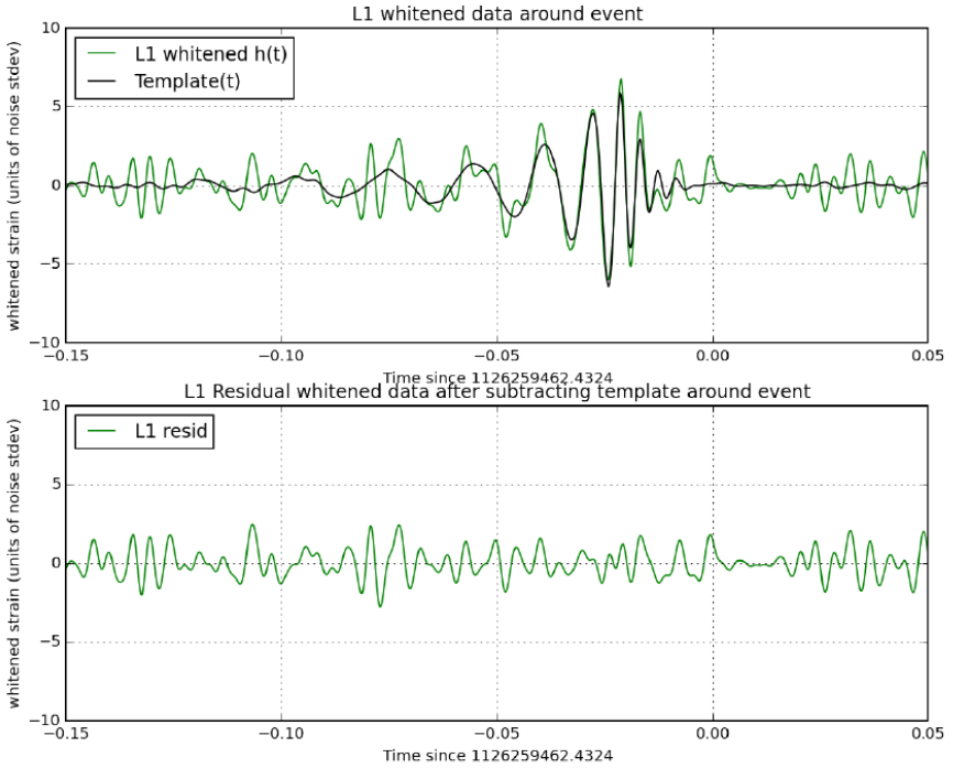
Used in almost all real physics (and wider scientific) research

- Particle physics (my research area)
  - E.g. finding the Higgs boson in ATLAS data



- https://atlas.cern/resources/opendata
- Analyse ATLAS Open Data with Jupyter Notebooks

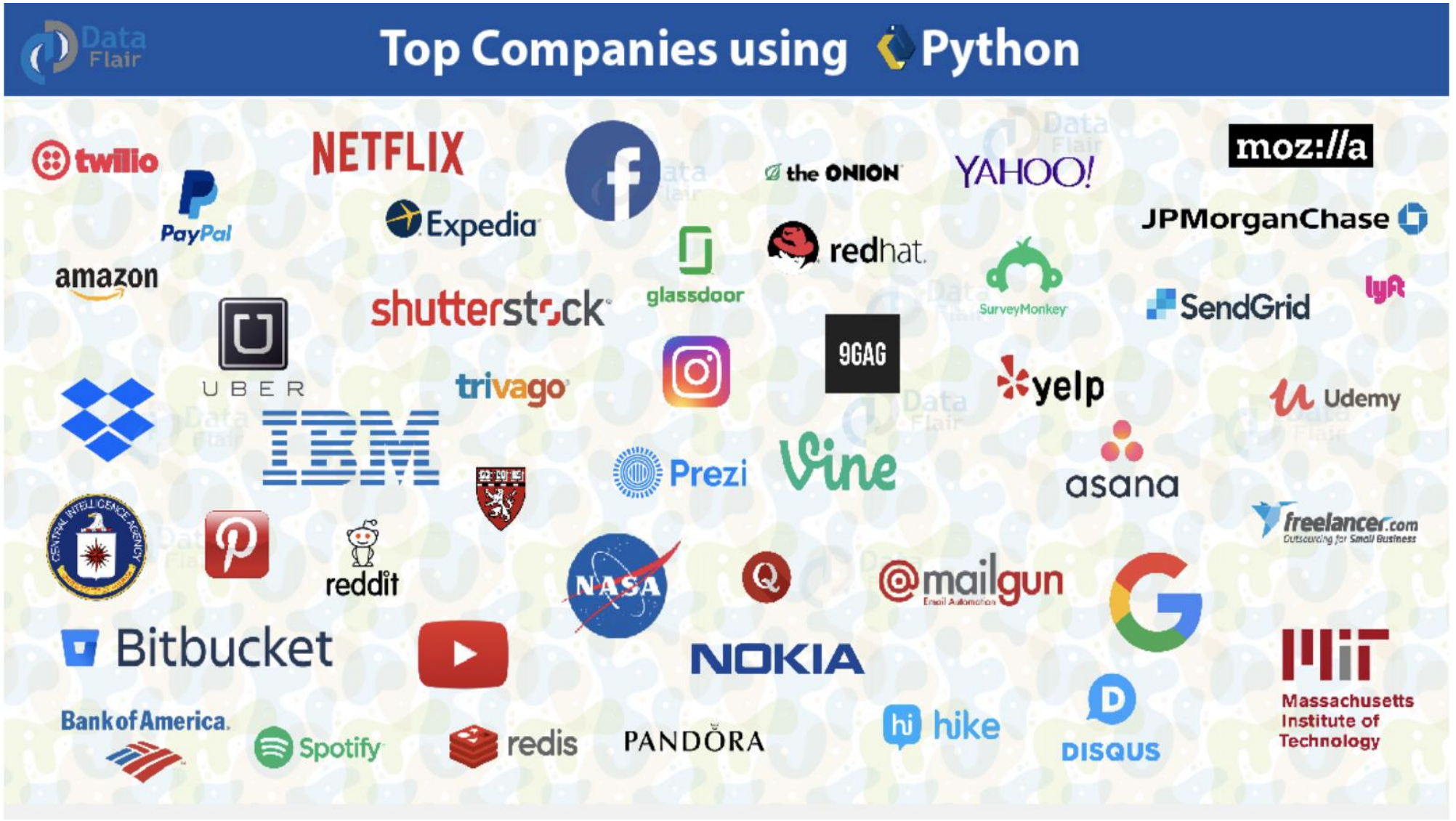
- Astronomy
  - E.g Gravitational wave detection with Ligo



- https://www.gw-openscience.org/tutorials/
- Guide to GW detection and noise

Why do I need to program?

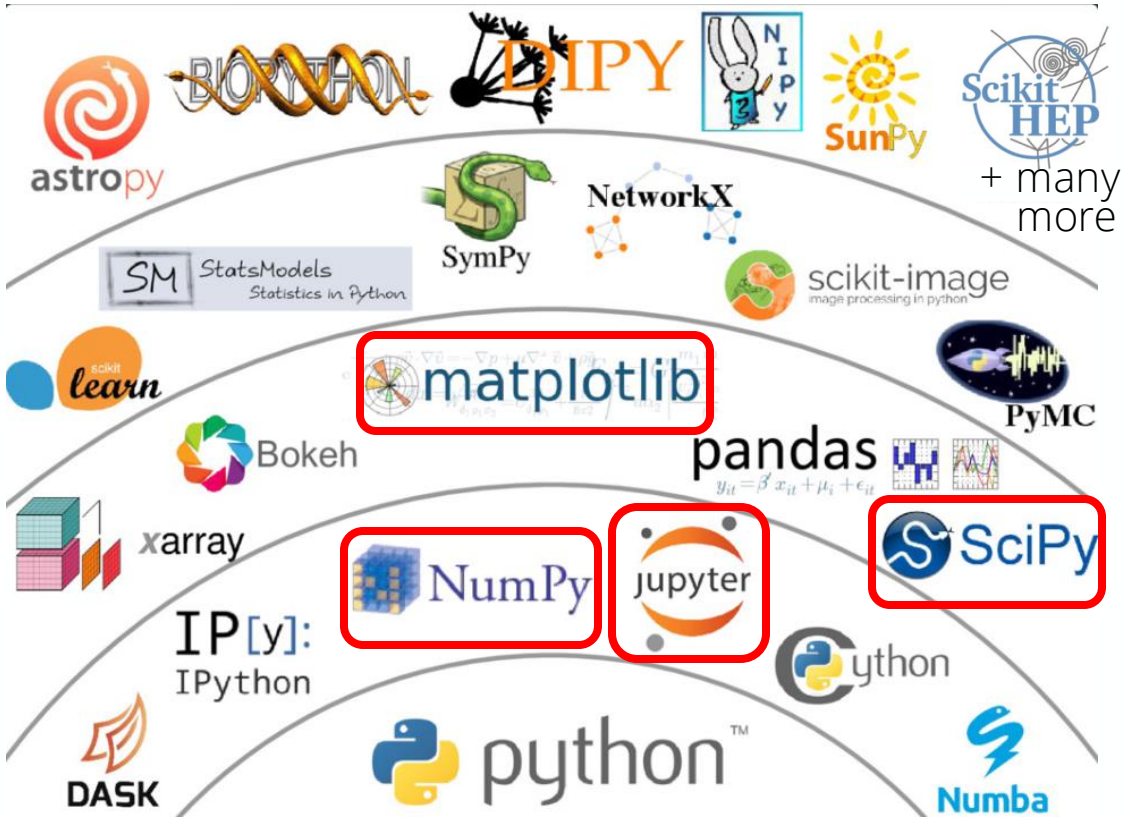
Used in many other careers outside science too

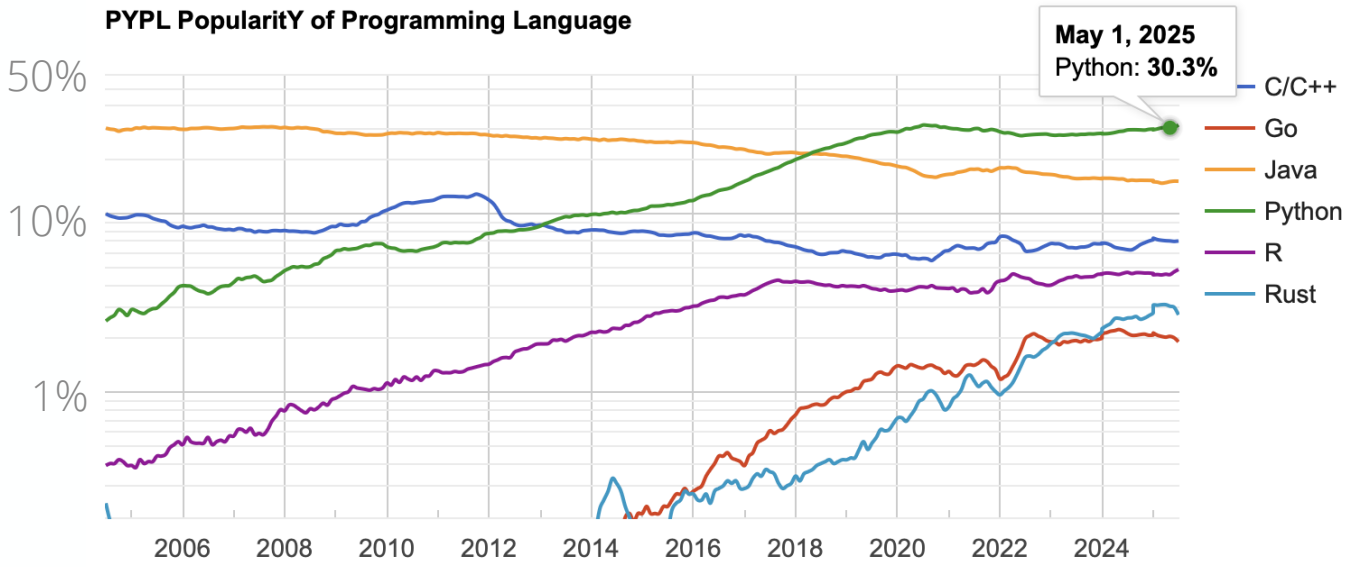


- https://data-flair.training/blogs/python-career-opportunities/

Why python?

- Powerful but easy to learn and use
  - Readable syntax
- Most popular and fastest growing
  - From google trends using PyPL
  - Similar trend within physics





- Large and mature library of tools
  - Both built-in and external
- Vast scientific ecosystem
  - Storage and manipulation of large data
  - Data processing and visualisation
  - Physics-specific packages

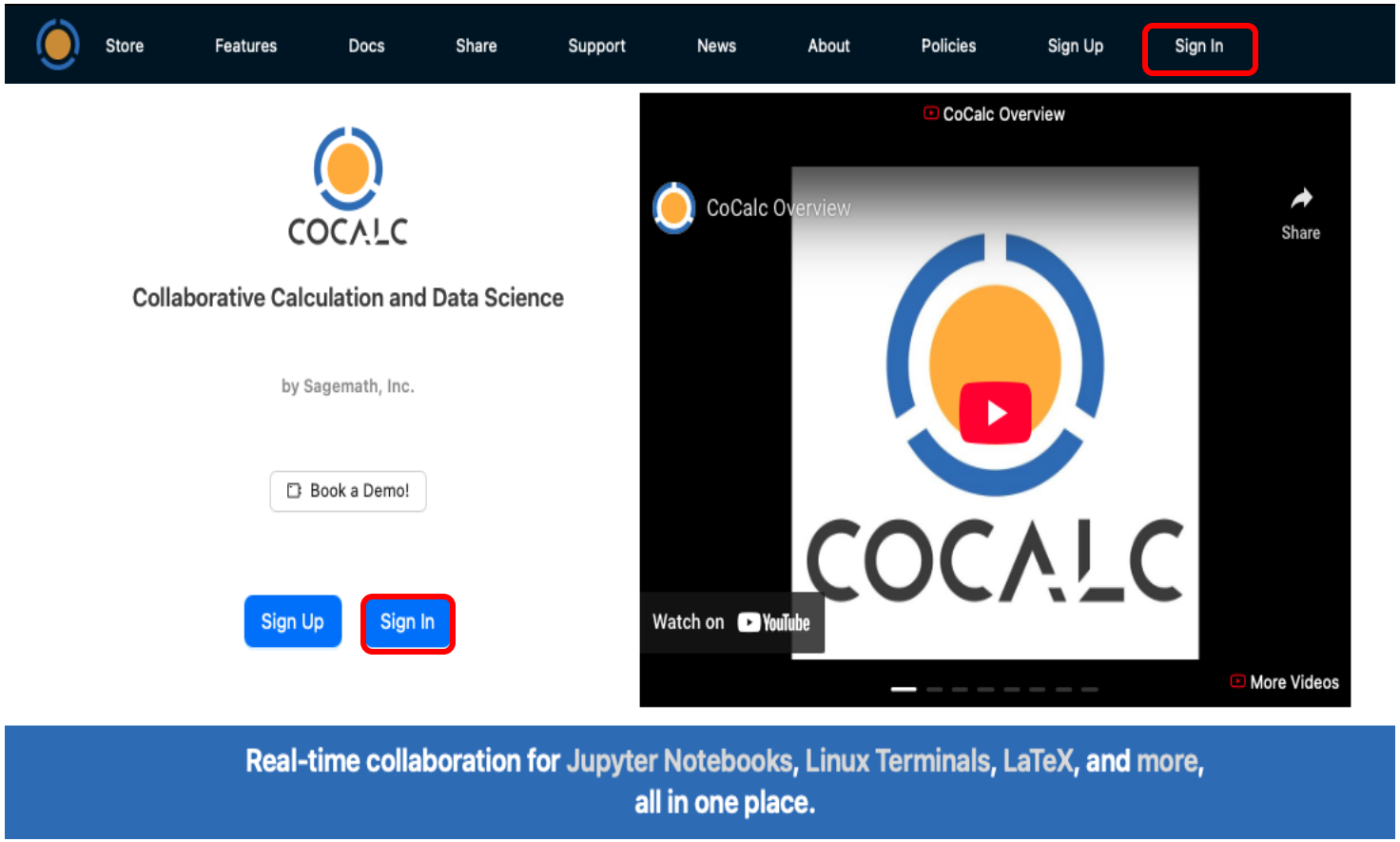
CoCalc

- We will be running our programmes using the web-based CoCalc platform: https://cocalc.com
  - A collaborative online workspace, allowing us to share documents, code and assignments

- Make sure you sign up to CoCalc
  - Using your university email in exactly the form the CoCalc invite was sent to

- Go to the first week's code by clicking on Projects and then in turn on
  - Your Name – PHYS105 Introduction to Computational Physics/PHYS105_2025
  - Phys105 Introduction to Computational Physics
  - ComputerClassesStudent
  - Phys105-Lecture01

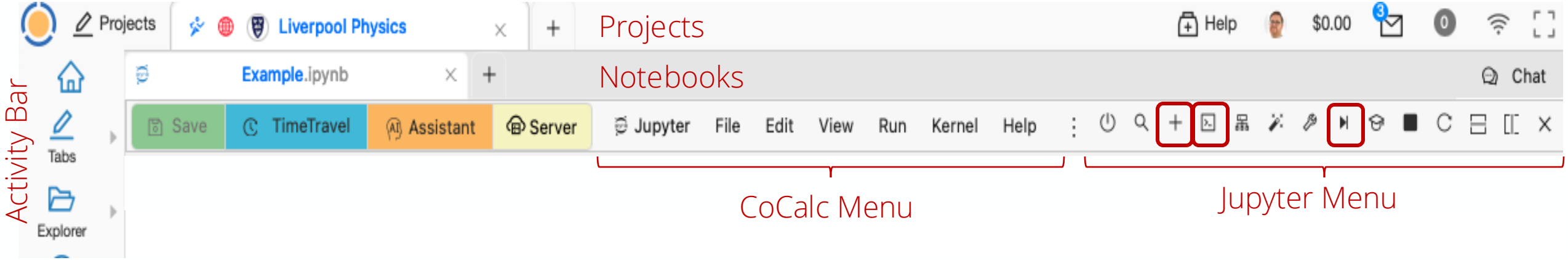
- Open the jupyter notebook
  - Phys105-Lecture01-student.ipynb
    - Make sure it is the .ipynb (notebook) not the .pdf



Note: If you can't see the folder/notebook you may need to click "Restart project" button and wait a bit first

Jupyter notebooks

- Jupyter notebooks are a web-based way of developing code interactively
  - They keep documentation, computer code and the resulting output together
  - Produce nicely formatted documents, which can be shared

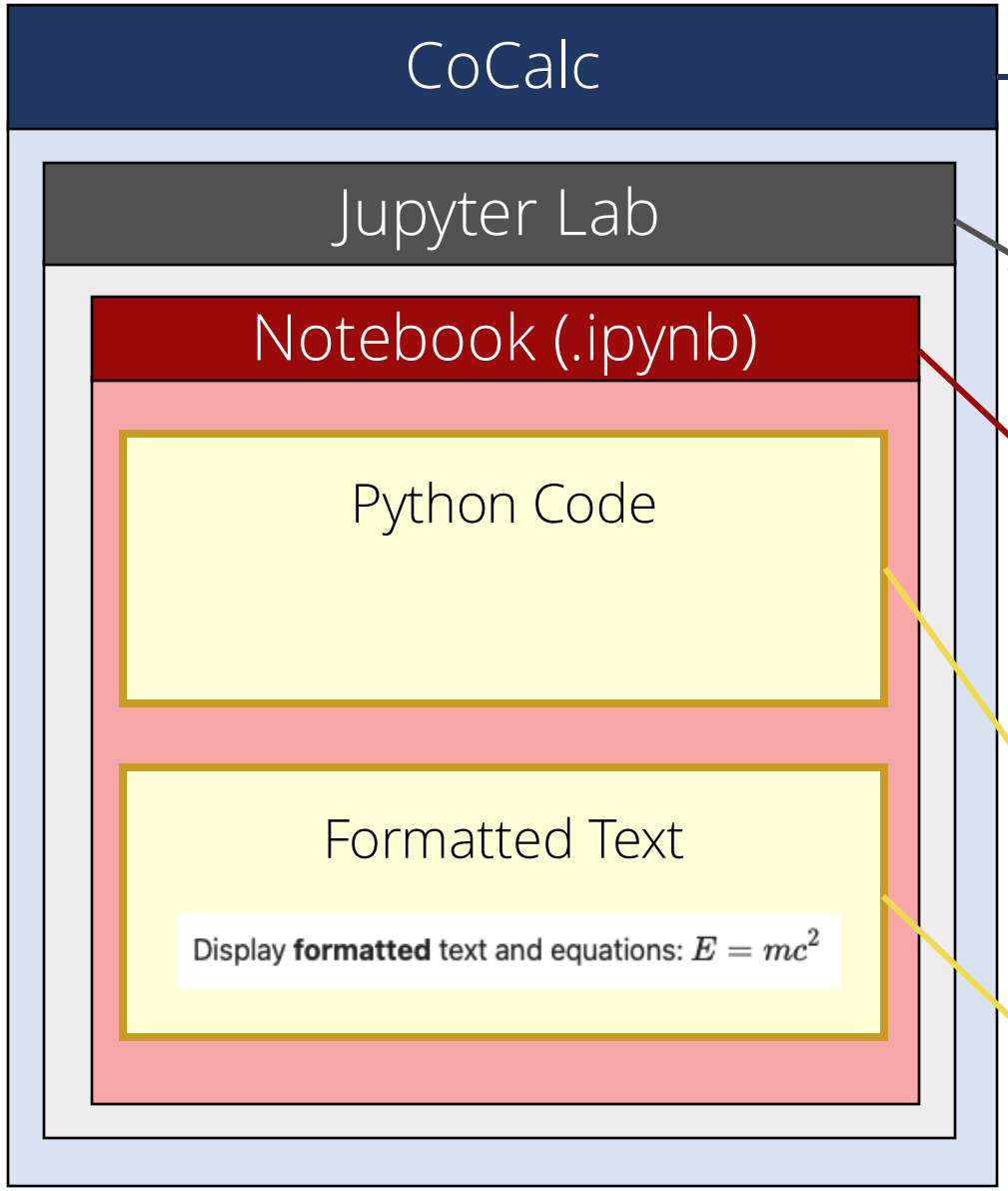


- The basic unit of a notebook is a cell, which can be created by clicking the + icon (or `Esc+b`)
  - To delete the cell, from the menu use Edit → Delete Cells (or `Esc+d+d`)

- A cell can contain either python computer code (default) or Markdown-formatted text
  - To change to text, select it (by clicking on it) then click on icon → Change Cell to Markdown (or `Esc+m`)
  - To change it back to code, select it and use Cell Type icon→ Change Cell to Code (or `Esc+y`)

- To run a cell (and move to the next) click the ▶| icon in cell or jupyter menu (or `shift + enter`)

Breakdown of components

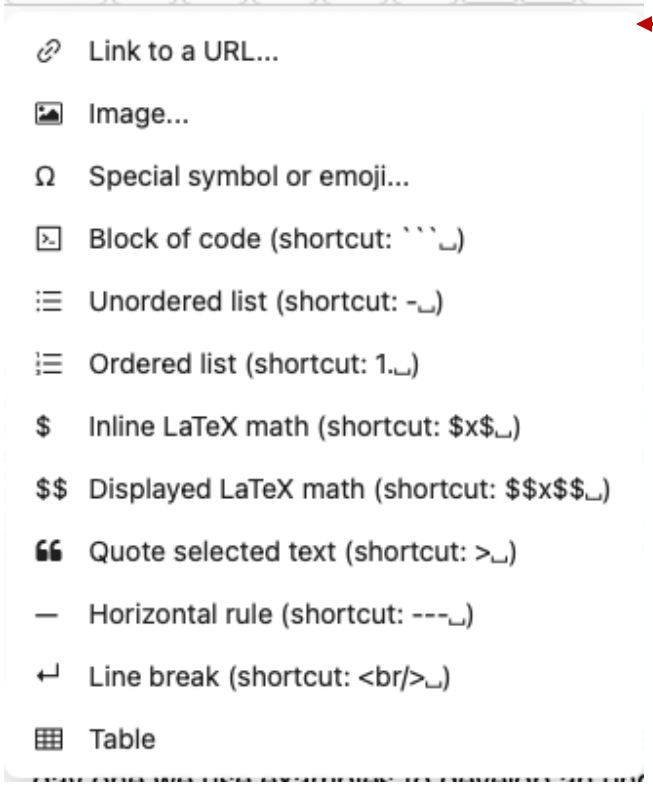


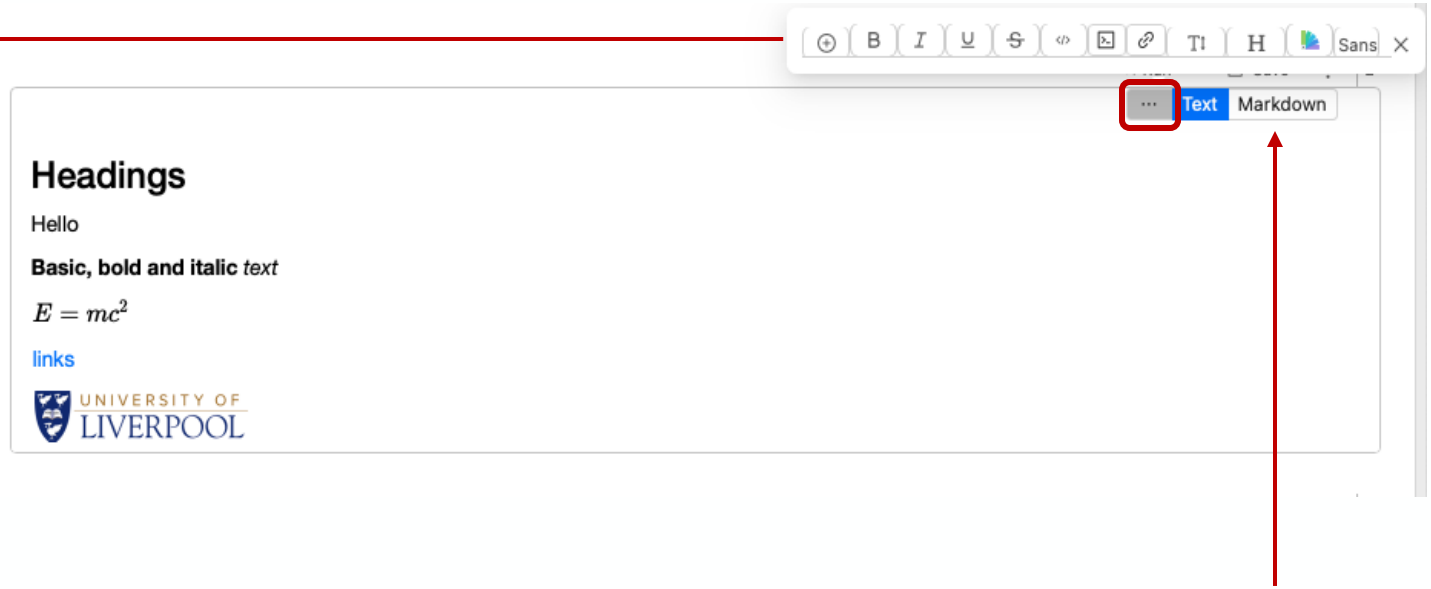
- Collaborative online coding platform (i.e. website)
  - Allows lecturers and students to work together
  - All files are stored, saved and synchronised here
- Runs program that reads .ipynb notebook files
  - Can also be run directly on your local machine
- A file type containing instructions for displaying text and code cells, interpreted by Jupyter Lab
  - Python cells are executed by the python interpreter
  - Text cells are rendered by markdown interpreter
  - These are separate entities brought together in Jupyter
- The programming language we will be using
  - Performs calculations and data processing
  - Our main goal is learning to code in python
- Formatted text entered via Markdown syntax
  - CoCalc has a graphical WISYWIG editor interfaced to this

Jupyter notebooks: Text cells

- Text cells allow you to enter formatted text along with formulae, hyperlinks, images and much more
  - These are formatted using a specification known as Markdown

- CoCalc has a helpful visual WISYWIG editor to avoid having to enter markdown directly (via ··· icon)
  - Simple to use but does lack some of the more detailed power of Markdown



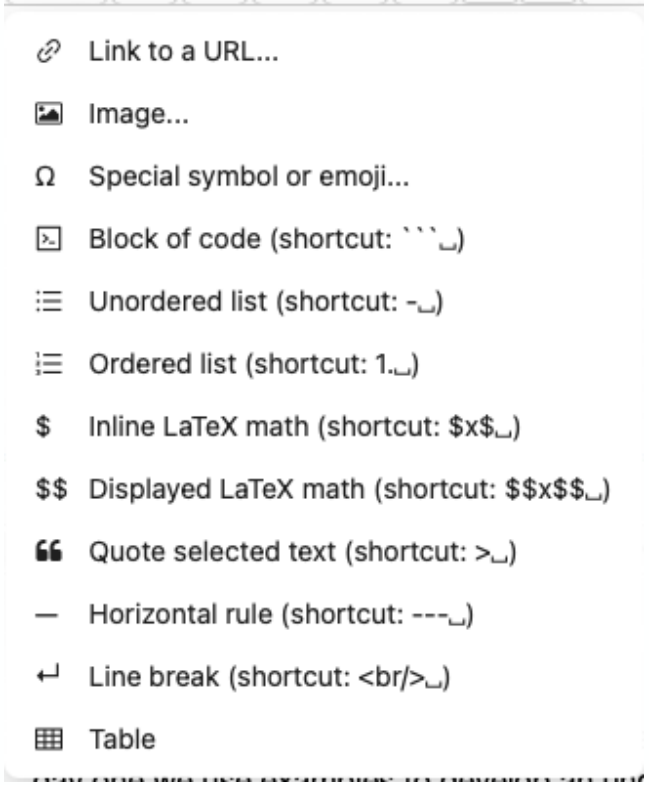


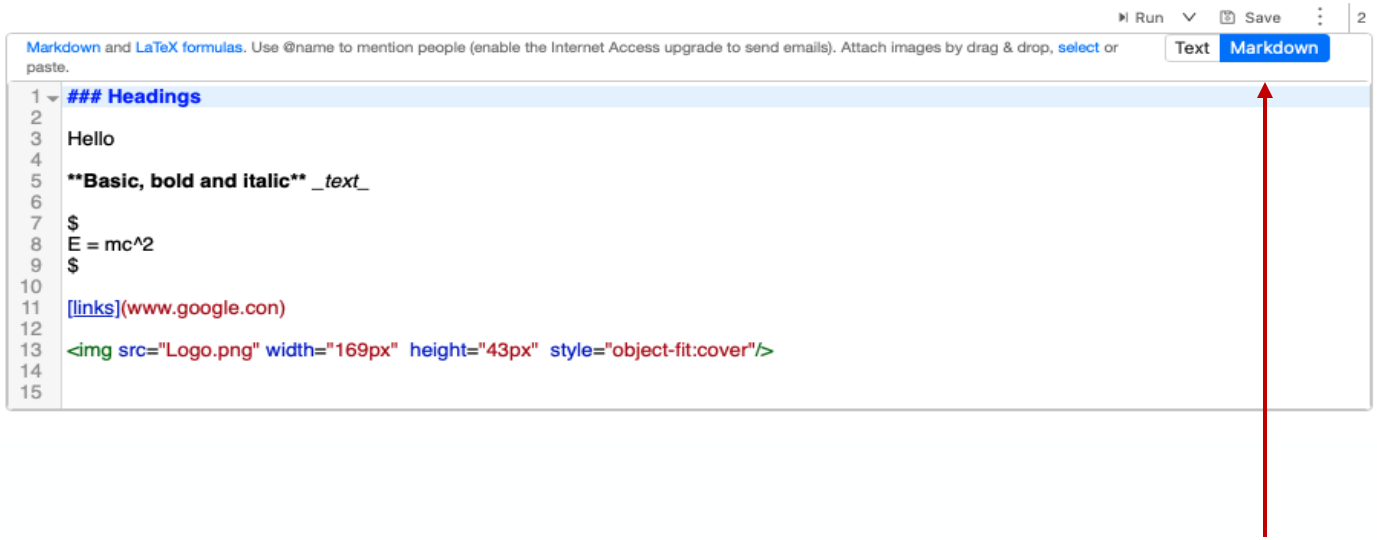
- Click the Markdown button, to see underlying structure
- Formulae are entered using LaTeX syntax
- To finish editing the text, simply run the cell

Jupyter notebooks: Text cells

- Text cells allow you to enter formatted text along with formulae, hyperlinks, images and much more
  - These are formatted using a specification known as Markdown

- CoCalc has a helpful visual WISYWIG editor to avoid having to enter markdown directly (via ··· icon)
  - Simple to use but does lack some of the more detailed power of Markdown





- Click the Markdown button, to see underlying structure
- Formulae are entered using LaTeX syntax
- To finish editing the text, simply run the cell

Jupyter notebooks: Code cells

- Text cells allow us to write attractive documents quickly and efficiently, but the power of notebooks is that they allow these documents to be combined with computer code
  - We can describe a problem and work on the program to solve it in the same place
  - Output is displayed inline

- Code cells used to write python code e.g.

```
In [ ]: 3+4
```

- When the cell is run, code is executed
  - And result output inline

```
In [1]: 3+4
Out[1]: 7
```

- Can edit the cell and rerun

```
In [2]: 3+2
Out[2]: 5
```

- Together, the separate code cells in a notebook build up a program
  - The results of previous cells, once run, can be used in subsequent ones

```
1 result = 3+4
```

```
1 result = result + 2
2 result
```

```
9
```

- During the course, we will learn how to develop python programs from scratch
  - Starting with the basics in the lecture 1 notebook

Hello, World!

- The canonical first program in any computing course is "Hello, World!"
  - The following python program does just that, highlighting some of the basics features of python

```
In [1]:
1 # This program prints Hello, World!
2 name = "World"                  # Assign a variable
3 print("Hello", name, "!")  # Print the result

Out[1]: Hello World !
```

Note: If they are not displayed, you should turn on line numbers for all cells via the *View* menu!

- The basic building blocks of any program are variables
  - Store some value and provide a symbolic label to refer to it
  - Assigned a value, here "World", with the equals "=" sign

```
name = "World"
```

- The other main building blocks of programs are functions
  - Pieces of code that do something with the variables
  - Called using brackets, with input variables as arguments

```
print ("Hello,", name, "!")
```

- Finally, we can document the code with comments
  - Start with a # and don't actually do anything when code runs
  - You will lose marks for not writing appropriate comments where needed

```
# Assign a variable
```

Python as a calculator

- The power of any programming language is allowing us to perform calculations
  - Often those which are too difficult or time consuming to do by hand

- Python can perform all the basic arithmetic
  - Using the math operators you might expect
  - Although some are a bit more obscure

```
In [1]: 333+666
Out[1]: 999

In [2]: 666-333
Out[2]: 333

In [3]: 666/333
Out[3]: 2.0

In [4]: 666*333
Out[4]: 221778
```

- It can also do more complex mathematics
  - Such as trigonometric functions
  - These live in the math module

```
In [1]: from math import cos, pi
        cosine = cos(pi)
        print(cosine)
        cosine = cos(2*pi)
        print(cosine)

-1.0
1.0
```

  - Note, variables can be reassigned to new values

- A **module** is simply a library (i.e. collection) of python code, such as variables and functions
  - Code in a module has to be **import**ed before use
  - We will come across many useful modules as the course progresses

Errors

- Often when coding you will make a mistake and introduce a bug into the code
  - Don't panic, this happens often even to experienced programmers!
- In many cases python will display an error message to help you debug what is wrong
  - These contain lots of useful information → please read them!

<0.1s Run Assistant Format 16

In [3]: Code

```
1  num = 10
2  den = 0
3  result = num/den
4  print(result)
```

Out[3]: Error Message

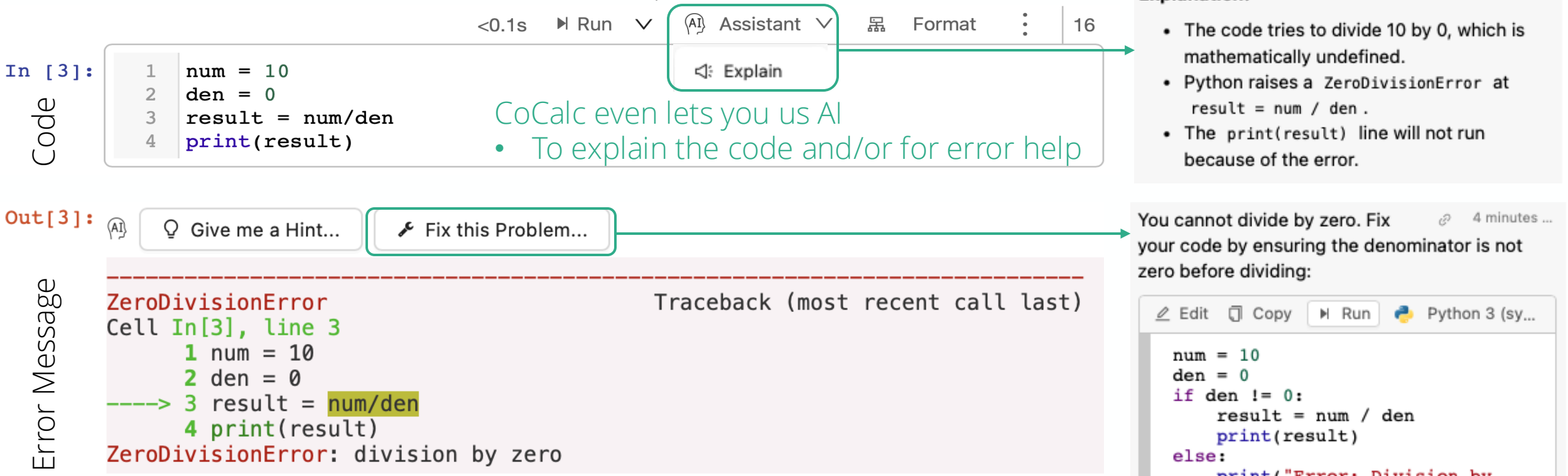
Give me a Hint... Fix this Problem...

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[3], line 3
      1 num = 10
      2 den = 0
----> 3 result = num/den
      4 print(result)
ZeroDivisionError: division by zero
```

- We will look at error messages & how to debug them in more detail later
  - Along with tips on good coding practice to help reduce the number of errors

Errors

- Often when coding you will make a mistake and introduce a bug into the code
  - Don't panic, this happens often even to experienced programmers!
- In many cases python will display an error message to help you debug what is wrong
  - These contain lots of useful information → please read them!

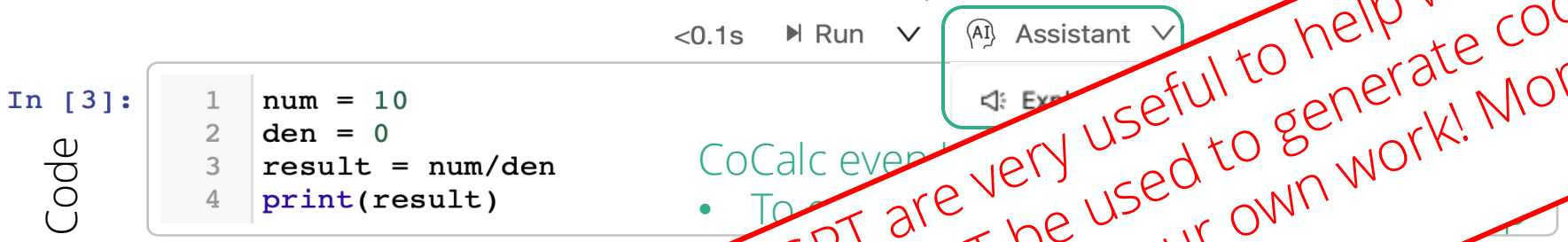


- We will look at error messages & how to debug them in more detail later
  - Along with tips on good coding practice to help reduce the number of errors

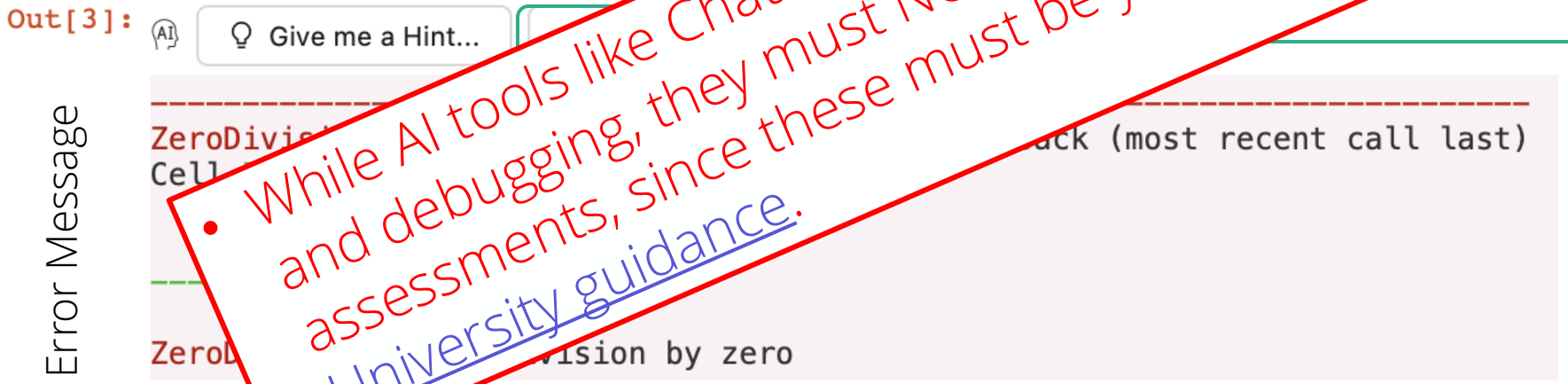
Errors

- Often when coding you will make a mistake and introduce a bug into the code
  - Don't panic, this happens often even to experienced programmers!
- In many cases python will display an error message to help you debug
  - These contain lots of useful information → please read them!

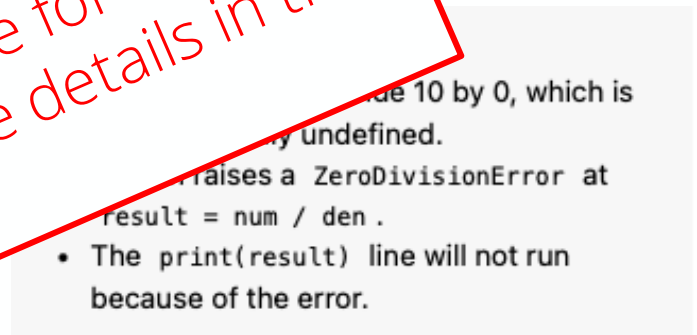
Code

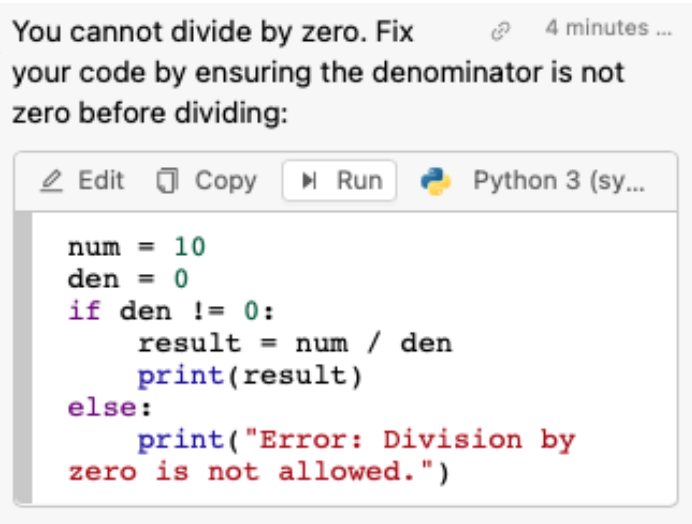


Error Message



- While AI tools like ChatGPT are very useful to help with understanding and debugging, they must NOT be used to generate code for the assessments, since these must be your own work! More details in this University guidance.





- We will look at error messages & how to debug them in more detail later
  - Along with tips on good coding practice to help reduce the number of errors

Summary

- Computer programming is an invaluable tool for many scientific problems
  - And python is by far the most popular language
- Over the course of PHYS105, you will learn how to program in python
  - Starting from scratch and by the end making your own module
- We will see how to use it to solve several real physics problems
  - Such as projectile motion, fitting experimental data, numerical solutions and MC methods
- We will see how to visualise and present the results
- The first computing classes next Monday will go over the basics of python and Jupyter from today
  - Will allow you to put into practice what we have discussed in the lecture
  - Demonstrators will help you to work through the material, including formative (practice) exercises
  - At the end you will have the first summative (counts to mark) assignment exercise
    - Which should be uploaded to canvas by the end of **Monday** (work submitted after extension period will score 0!)