Introduction to Computational Physics (PHYS105)

Lecture 1: Introducing Python and Jupyter

Carl Gwilliam
(C.Gwilliam@liverpool.ac.uk)
UNIVERSITY OF LIVERPOOL

- **Attendance code + start video**
- **Activate Poll and open webpage**
- Hello everyone and welcome to Introduction to Computational physics aka PHYS105
- My name is Carl Gwilliam - Discuss what I do briefly
- **Throughout the lectures, please feel free to interrupt at any time to ask questions by raising your hand**

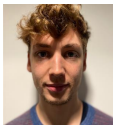
- Lectures
  - Tuesday @ 09:00-10:00
  - In Thomson-Yates Lecture Theatre (Rm 104)
- Computer Classes
  - Monday @ 09:00--11:00
  - In CTL PC teaching centre (PCTC)
- Assessment
  - 100% coursework (no exam)
  - Exercises in computing classes
    - Several formative / week
    - One summative / week (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)

Organisation

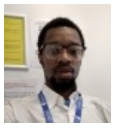
- Staff
  - Prof Carl Gwilliam (Rm 315)
    - Module Coordinator
    - C.Gwilliam@liverpool.ac.uk

  - 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

2

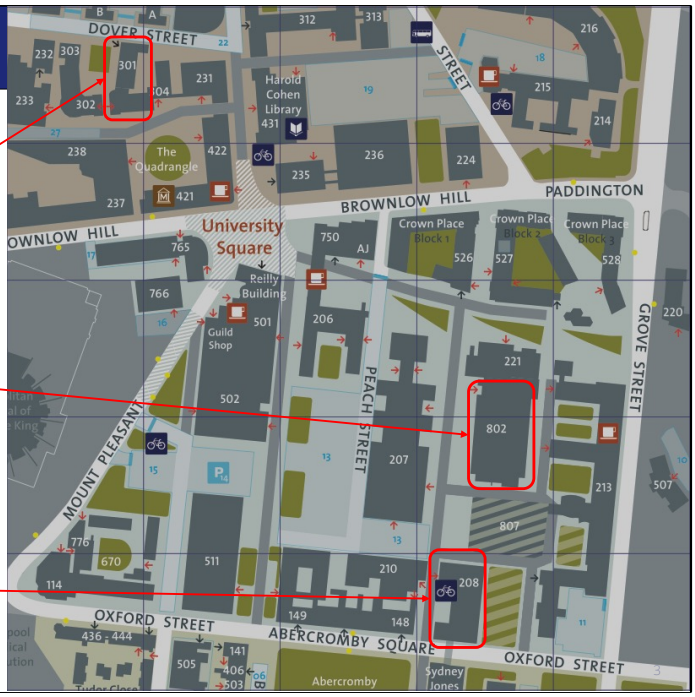
- I wanted to start with reminding you about the organisation of the course and some of the details that I presented in Welcome week
- Staff before assessment: will help you with questions/issues in problems classes
  - Have emails -> feel free to contact us
- Discuss what formative and summative are
  - Please don't skip formative – only way you can learn coding is by practice + we can give help on these
  - And how much help can have on each
    - Summative = minimal, just pointer in correct direction
- **Discuss when release, when hand in and when get feedback**
- Penalties and extensions
  - PLEASE NOTE
  - After that will be marked as zero

Locations

Lectures
- Thompson Yates Lecture Theatre
- Room 104

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

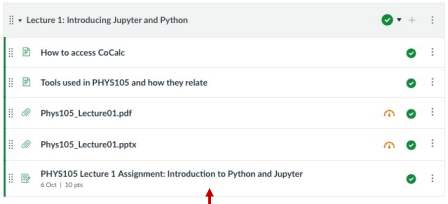
Academic Offices
- Oliver Lodge Laboratory
- Room 315 & 310 (3rd floor)



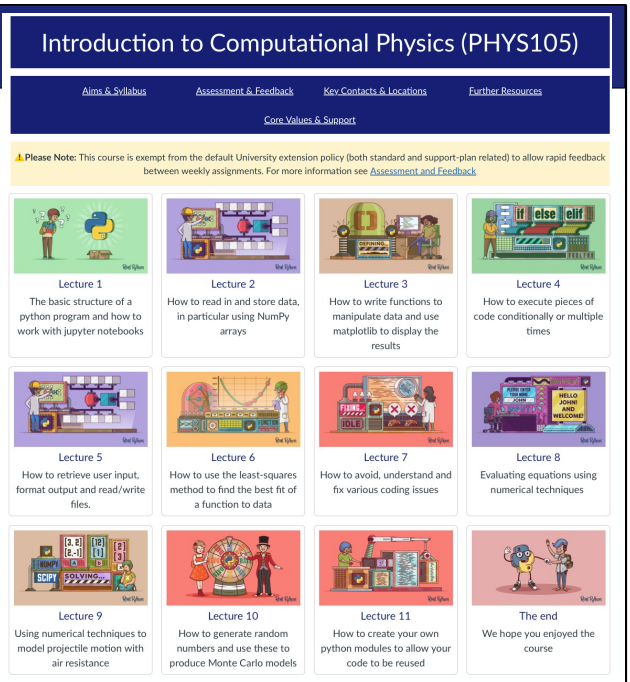
- PCTC on first floor
- Open door policy so welcome to come and see us at any point.

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



- All this info and more is summarised on canvas [show]
  - Top -> info -> please read esp. assessment part
  - **Module layout**
  - First lecture -> copy of lecture, will put up recording, first assignment
  - All assignments up with deadlines and will open before Tuesday's lecture
  - Please take a look

Aims and outcomes

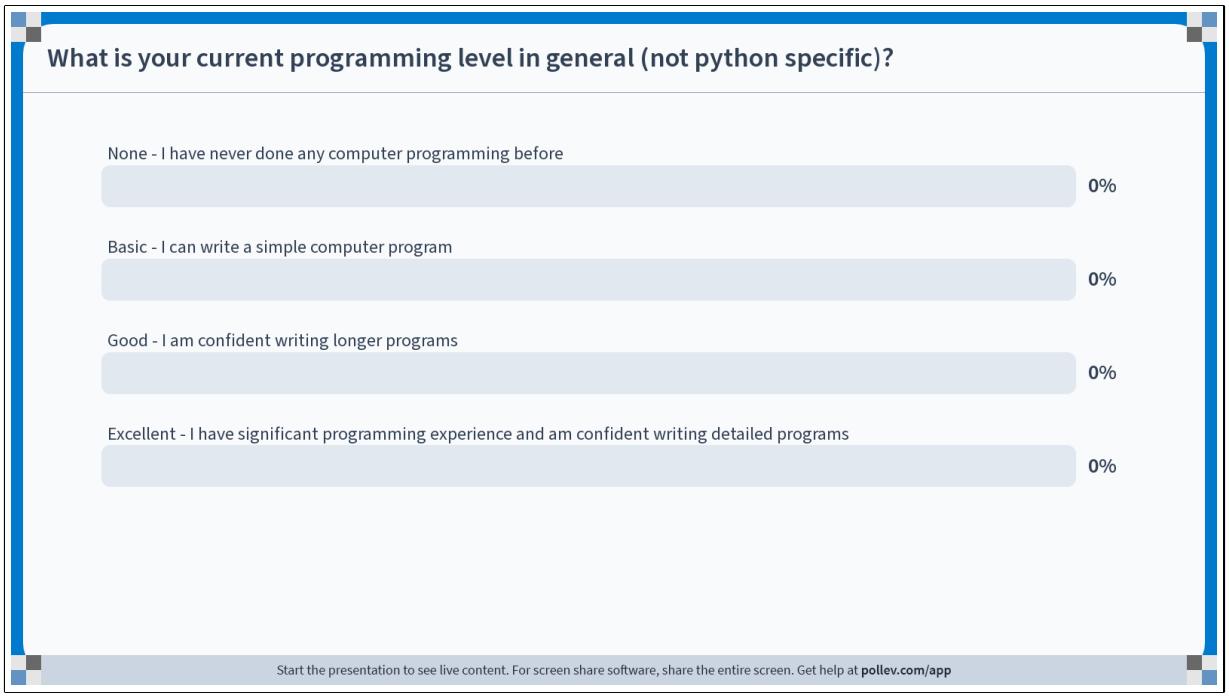
- The main aim of the course is to be able to create and run computer algorithms to solve real physical problems, breaking the problems down into steps amenable to such methods
  - Illustrating the insight into physics which can be obtained using computational methods
- In doing so, we will introduce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

5

- As mentioned in Welcome Week …
- Don't worry if you don't understand some of the words right now, they will be introduced throughout the course
- Since there will be a wide range of abilities - nothing to GCSE/A-level CS - we will not assume any prior coding knowledge

- It is useful to know the prior knowledge distribution so have a poll.
  - Go to web address or scan QR code and fill in now
  - Discuss results + will be added to canvas



Do not modify the notes in this section to avoid tampering with the Poll Everywhere activity.
More info at polleverywhere.com/support

What is your current programming level in general (not python specific)?
https://www.polleverywhere.com/multiple_choice_polls/RD1Hm21xsrAL9fVXsPqd4?state=opened&flow=Default&onscreen=persist

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

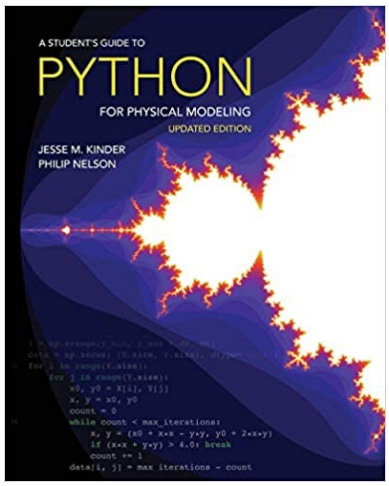
7

- Link fitting to lab course
- Complex program -> step back -> good practice
- How do calculations, such as solving equations, calculating differentials and integrals etc, numerically (where no algebraic solution)
  - Build up steps in analysing physics problem
- In Lecture 9 we will look at physics problem that requires a numeric solution in more detail, namely projectile motion, using the techniques we have learnt
- In Lecture 12 -> python module/library can take forward and use in labs

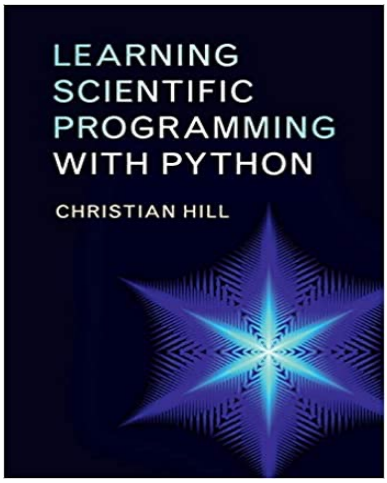
- **The code snippets shown in each lecture will be released as a python notebook, along with the assignment in CoCalc, before the lecture so you can look at in parallel**

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



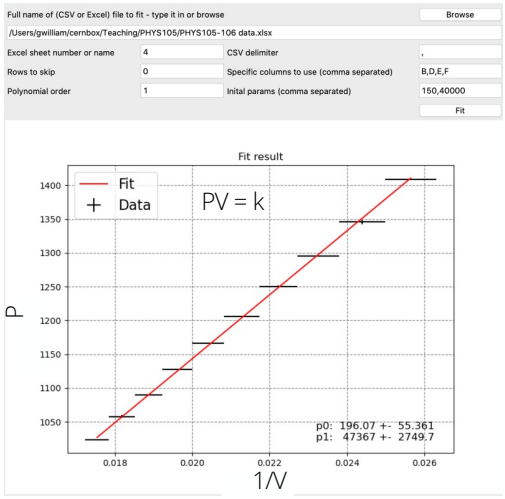
8

- Updated edition or newer
- These are also available in library
- All the details are on canvas
- **Mention online resources [show in canvas] + google**
- **ANY QUESTIONS about the structure of the course?**

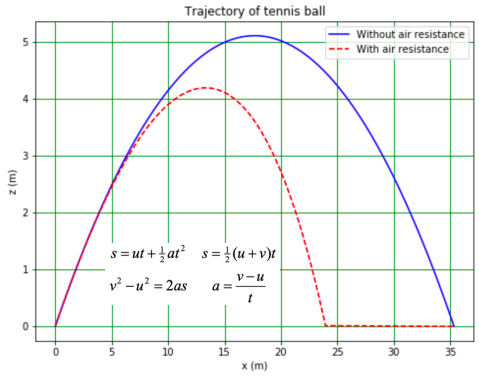
Why do I need to program?

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



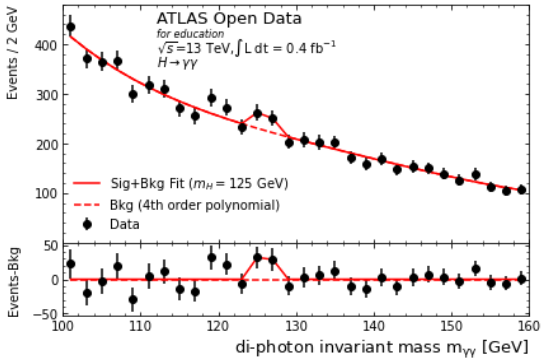
9

- Since you signed up to a physics degree, you might ask yourself why you need to program
  - Went over this in welcome week but worth repeating
- In fact, there are many cases where this will be essential in your physics degree
- The most obvious is analysing experimental data taken in your lab module
  - Pressure * volume = constant
  - Plot results, fit with a straight line to extract proportionality constant, k
- GUI that will use in lab
  - Reproduce as final module w/o GUI part
- Another is to calculate results that have no exact analytic solution
  - **Look at projectile motion in Lecture 9 as mentioned**

Why do I need to program?

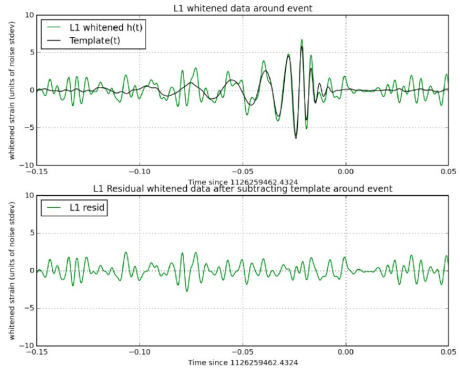
Used in almost all real physics (and wider scientific) research

- 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

10

Beyond your immediate degree, programming is used in almost all physics research

And more generally in most scientific research areas

- Look for Higgs, discovered in 2012, which I was involved in
  - Plots shows the mass of the two photons
  - Peak in mass yy at 125 -> existence of higgs boson
  - Can do with public data using tools learn over the course
- Another example was the amazing discovery of GW from BH merge
  - See the displacement vs time where the oscillations increase as two black holes approach
  - Again, can do publicly with tools learnt in this course

Why do I need to program?

Used in many other careers outside science too

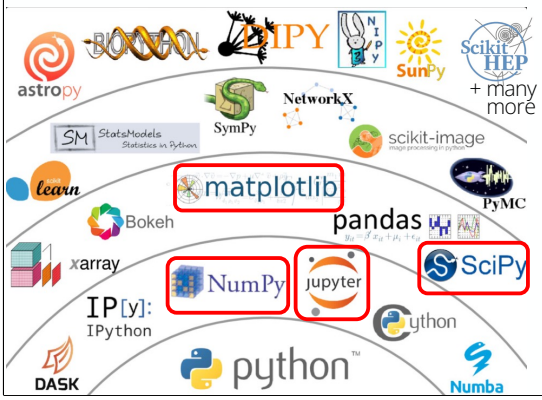
- https://data-flair.training/blogs/python-career-opportunities/

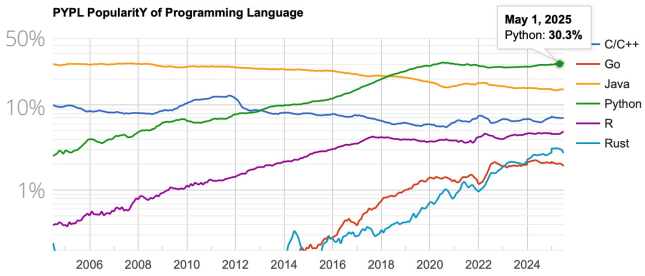
11

- Point out some of the varied companies, from the tech giants to Netflix, both programming the platform itself and also analysis of viewer data to things you might not think of eg. pandora jewellery
- [20 mins]

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





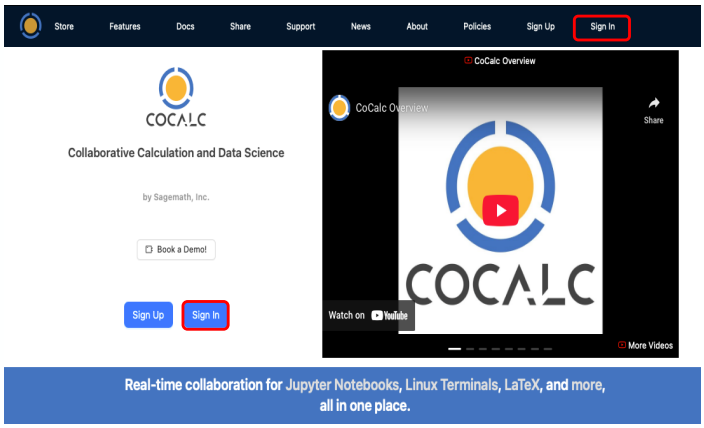
- Large and mature library of tools
  - Both built-in and external
- Vast scientific ecosystem
  - Storage and manipulation of large data
  - Data processing and visualisation
  - Physics-specific packages

12

- Hopefully, I have convinced you that programming is an invaluable skill during your degree and whatever you go on to do after
- But why python?
- Most popular programming language: generally and in physics
- Go through diagram pointing out ones we will use
- Physics-specific: astro and HEP (point)
- **ANY QUESTIONS on why we are learning to program in python?**

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



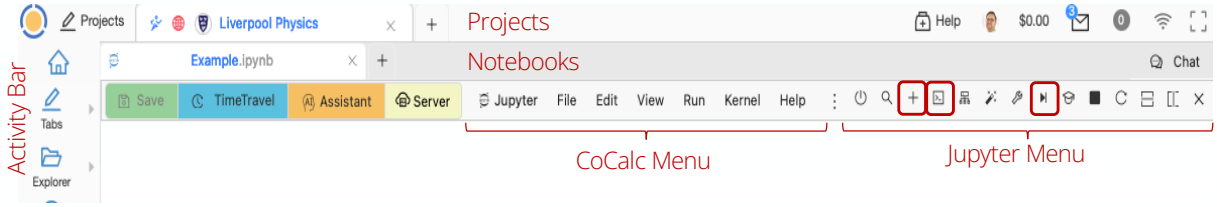
Note: If you can't see the folder/notebook you may need to click "Restart project" button and wait a bit first

13

- As discussed in Welcome Week … [show]
- Should have signed up in welcome week or yesterday – if not please make sure you do before Monday's PC class classes
- Go through to sign up
- **First notebook has now been released so should be able to find it [show]**

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

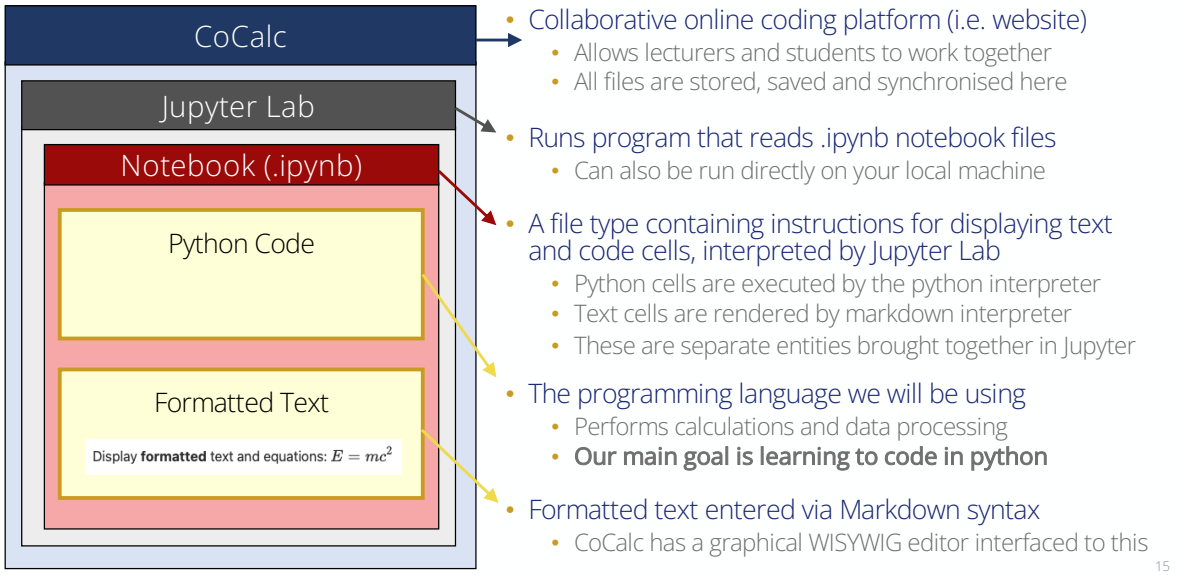


- The basic unit of a notebook is a cell, which can be created by clicking the + icon (or `Esc+b`)
  - To delete the cell, from the menu use Edit → Delete Cells (or `Esc+d+d`)
- A cell can contain either python computer code (default) or Markdown-formatted text
  - To change to text, select it (by clicking 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`)

14

- Once on cocalc we will be using jupyter notebooks [show]
  - Mention the tabs for projects and notebooks along with the two menus
- Go through to basic cell unit then go back to show menus on cocalc and how to create/delete cells
  - Go through changing and running

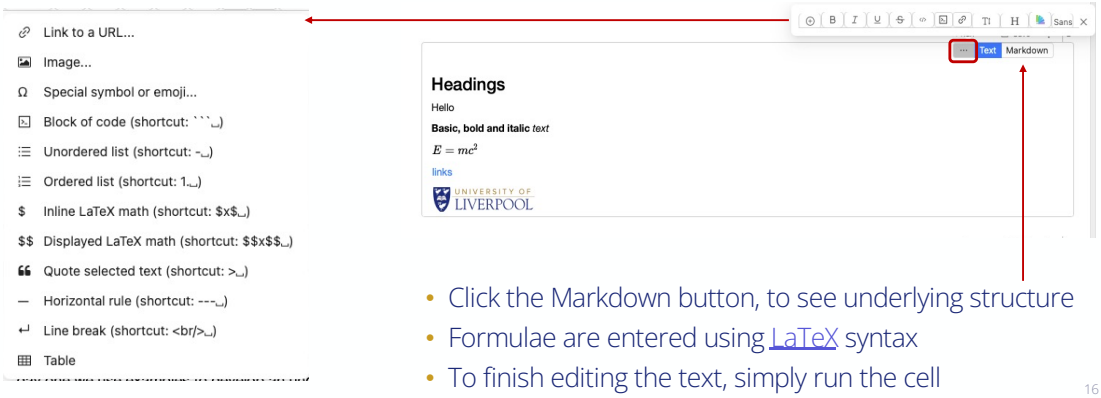
Breakdown of components



- Since there are lots of tools, I want to remind you how these fit together to make sure we are on the same page …
- **Local machine**
  - **If prefer to work like that there are instructions on how to setup jupyter lab in the first notebook + can ask me**

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



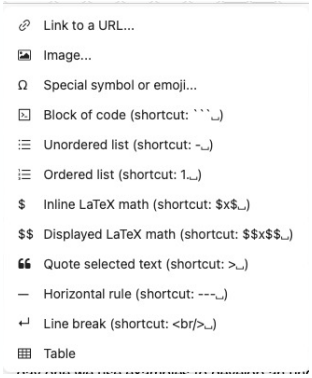
- Click the Markdown button, to see underlying structure
- Formulae are entered using LaTeX syntax
- To finish editing the text, simply run the cell

16

- **Although our main goal is to learn python, let's start by looking briefly at text cells …**
- Go through in notebook [show]
  - **Double-click to edit**
  - Cover headings, bold/italic
  - Running and editing again
  - Equations, links and images via + icon
- Reminder that must run
- **If you want to see the underlying markdown just click on the "markdown" button at the top right [change]**
- **ALL lecture snippets will be available as a notebook in that week's CoCalc directory for you to follow along or play with after**
- **ANY questions on text cells?**

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

17

- **Although our main goal is to learn python, let's start by looking briefly at text cells …**
- Go through in notebook [show]
  - **Double-click to edit**
  - Cover headings, bold/italic
  - Running and editing again
  - Equations, links and images via + icon
- Reminder that must run
- **If you want to see the underlying markdown just click on the "markdown" button at the top right**
- **ALL lecture snippets will be available as a notebook in that week's CoCalc directory for you to follow along or play with after**
- **ANY questions on text cells?**

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

18

- Go through first bullet and then enter/show code interactively in notebook [show]
  - Mention that see it is code by "In []"
  - Also that when you run you see a number which increments each time you run a cell
    - Can see the order in which they are run
- Come back to last bullet

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
```

19

- Line numbers!
- Can refer to value by variable name afterwards
  - Quotes mean a string, which is just a sequence of characters
- Functions e.g. print -> comma-separate set of argument
  - Can be entered directly or using a variable
- What happens if execute -> see results "inline"
  - Show interactively [show]
- Please add to-the-point comments to your code
  - So you can understand when you come back
  - So we know what you are doing
  - Can be on a line of their own or at the end of a line of code (but not in the middle)
- When run this program prints Hello World
  - Note how the variable "name" has been replaced by its actual value

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

20

- Will see other operators in the notebook
- Go through interactively, especially how trig example works
- … And even create our modules own by the end
- **ANY QUESTIONS on the basics of python coding?**
- **[40 mins]**

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

Code

```
In [3]:
1 num = 10
2 den = 0
3 result = num/den
4 print(result)
```

Error Message

```
Out[3]:  Help me fix this...
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
/tmp/ipykernel_878/1277757653.py in <cell line: 3>()
      1 num = 10
      2 den = 0
----> 3 result = num/den
      4 print(result)
ZeroDivisionError: division by zero
```

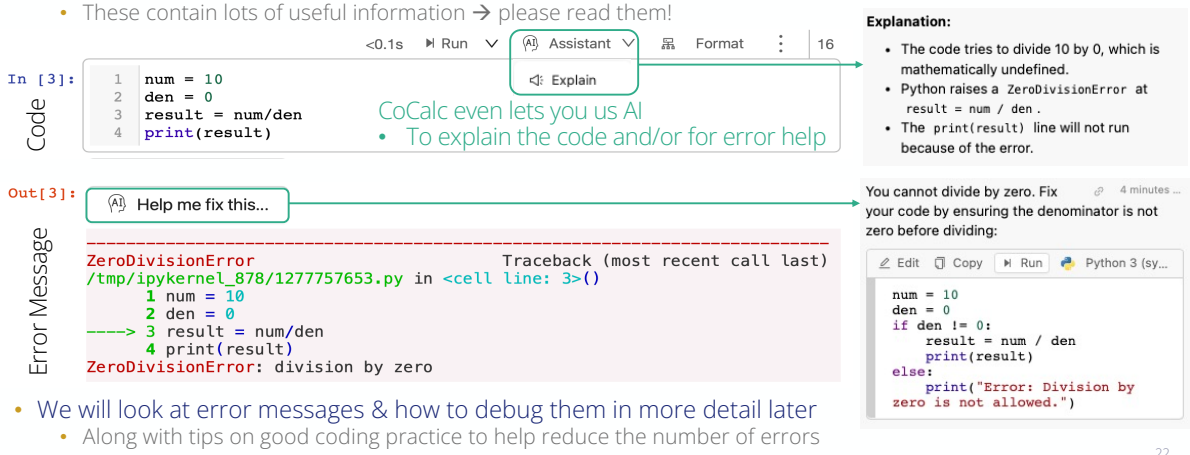
- We will look at error messages & how to debug them in more detail later
  - Along with tips on good coding practice to help reduce the number of errors

21

- Show interactively [show]
  - Go through what order to look and what it means, including line numbers

Errors

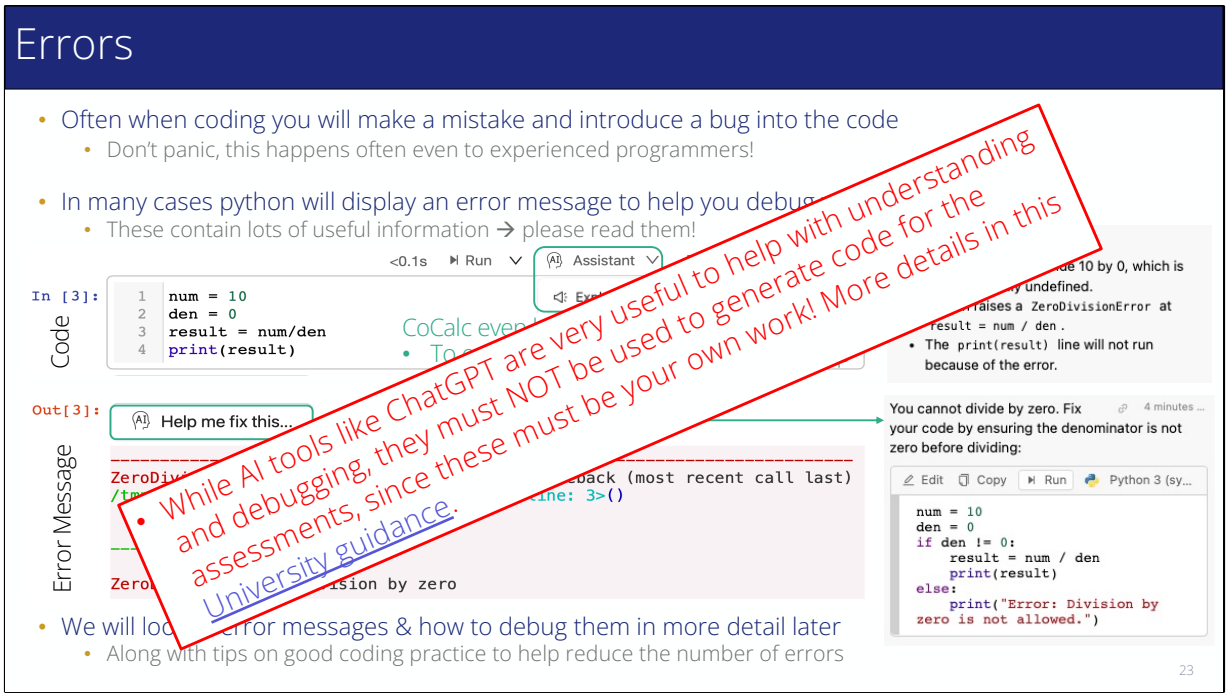
- Often when coding you will make a mistake and introduce a bug into the code
  - Don't panic, this happens often even to experienced programmers!
- In many cases python will display an error message to help you debug what is wrong
  - These contain lots of useful information → please read them!



- 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

22

- CoCalc even has integration with the ChatGPT AI assistant [show]
  - At the top right of each code cell you can ask it to explain the code to you
  - In addition, if you get an error it can help you fix it
    - You may not understand all the suggestions at this point



- CoCalc even has integration with the ChatGPT AI assistant [show]
  - At the top right of each code cell you can ask it to explain the code to you
  - In addition, if you get an error it can help you fix it
    - You may not understand all the suggestions at this point

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!)

24

- Go through up to put into practice
- **Show notebook for lecture 1 again [show]**
  - Mention that goes over what was in lecture + expands
  - Don't just read, execute the cells and play with the markdown/code
  - Important to try the formative exercises as practice is the only way to learn to program
  - Demonstrators, both staff and postgrad, there to help
  - **Can start already**
  - Summative -> print to pdf [show] -> **download [show]** -> hand in on canvas
  - **Deadline + extensions**
  - **When get feedback/marks:**
    - **General solutions + feedback on formative in next lecture**
    - **Individual feedback + mark on Canvas within 2 weeks**
- **ANY FINAL QUESTIONS?**
- [50 mins]