Imperial College London

Computing - Lecture 1

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Dr. Aidan Crilly

a.crilly16@imperial.ac.uk

Lecture 1

1. Course Outline
2. Scientific Computing
   - What is it?
   - Why do we do it?
3. The tools that we will use
   - Python
   - Jupyter Notebooks
   - Spyder IDE
4. Tips for being an effective programmer

Dr. Brian Appelbe b.appelbe07@imperial.ac.uk

Lecture 1 Part 1

1. Course Outline
2. Scientific Computing
   - What is it?
   - Why do we do it?
3. The tools that we will use
   - Python
   - Jupyter Notebooks
   - Spyder IDE
4. Tips for being an effective programmer

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Course Objectives

- Introduce the concepts of programming & scientific computing.
- Learn the basics of the Python programming language.
- Learn how to use Python for analyzing data in Practical Labs.
- Learning how to independently improve your programming skills.

- **If this is your first time coding** - Don't worry! This course assumes that you have no prior programming experience.

- **If you have coded before** – There are lots of exercises to improve your skills. You will meet new ideas about data analysis and scientific computing.

Course Outline

- 6 Lectures (Recorded)

  Mondays: Oct 11th, 18th, 25th

  Nov 1st , 8th , 29th

- 4 Lab sessions – Blackett Computing Suite

  Weeks 2-5 (Oct 11th – Nov 5th)

  Groups D-F: Mon 9am-12pm

  Groups G-I: Tue 9am-12pm

  Groups J-L: Thur 9am-12pm

  Groups A-C: Fri 9am-12pm

- Computing Project

  Begins with Lecture on Monday Nov 8th

  Submission deadlines is Friday Nov 26th

  No scheduled Lab sessions but there will be drop-in sessions

Assessment & Feedback

- Computing Project is assessed - 10% of Practical Physics Module.
- You must pass the Computing Project to progress to Year 2.
- Computing Labs are not assessed but attendance is mandatory.
- Demonstrators will give you feedback on your progress in Computing Labs.

Computing Lab Sessions

- Each Computing Lab session will consist of 1 core worksheet and 1 advanced worksheet.
  - Worksheets will be in the form of Jupyter Notebooks.
  - They will be available on Blackboard.
- You must complete the core worksheet, in your own time if necessary.
  - The project will require elements from all core worksheets.
- The advanced worksheets will help you to develop advanced programming skills.
- Solutions to worksheets will be available 1 week after each Computing Lab session.

Computing Lab Sessions

- Computing Lab sessions will take place in the Blackett Computing Suite on Level 3.
- There will be many demonstrators available to provide guidance, answer your questions, etc.

Computing Notebook

- Your should keep a Computing Notebook to:
  - Keeping a record of the code that you wrote.
  - Recording results that you get from your code.
  - Planning code (e.g. Flow Charts) before you start writing it.
- Your Computing Notebook should be different to your Practical Lab Book.
- It can be a physical notebook or digital (e.g. OneNote).
- Your Computing Notebook will not be assessed, but you may sometimes need to show it to a demonstrator.

Y1 Computing Primer

Y1 Computing Primer

Dr. Brian D. Appelbe & Dr. Aidan J. Crilly

September 2021

1. Introduction

Welcome to Physics at Imperial College London. An essential tool for all Physicists is the art of computer programming. At Imperial we teach a programming language called Python. For some of you this will be the first time you have ever had to program. Don't worry, the Y1 Computing course assumes that you have no prior programming experience. By the end of the course you will be able to write your own programs in Python and use it for analysing experimental data.

This introductory guide contains information on Python, the software that you will use for writing Python code and some tips for setting this up.

2. What is Python?

Python is a programming language used universally in academia and industry. The official website

Lecture 1 Part 2

1. Course Outline
2. Scientific Computing
   - What is it?
   - Why do we do it?
3. The tools that we will use
   - Python
   - Jupyter Notebooks
   - Spyder IDE
4. Tips for being an effective programmer

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Why *scientific* computing?

Scientific Computing is the use of computers to solve problems in science.

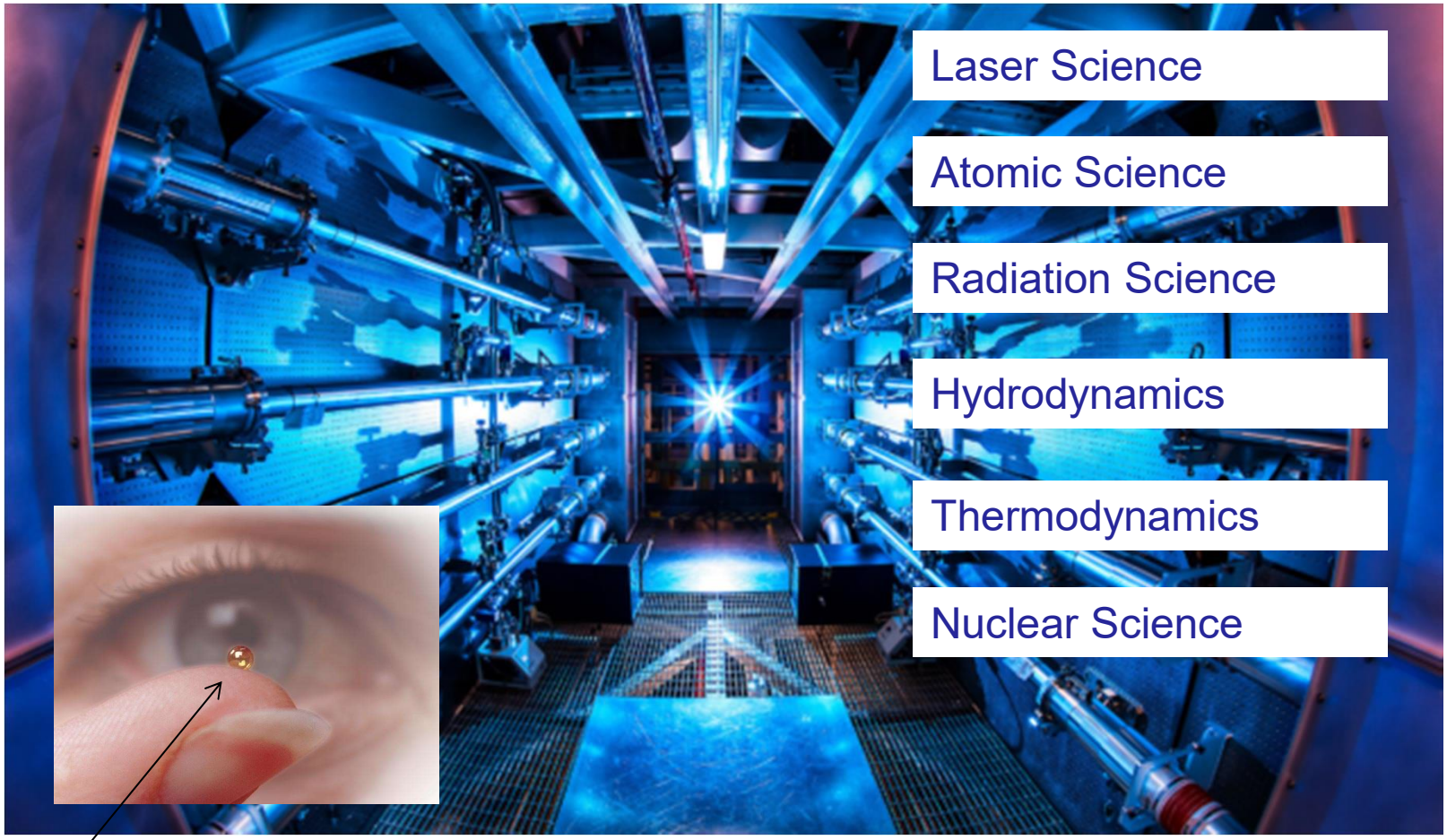
Complex calculations

- Numerical computation of equations
- Multiphysics computation

Repeated calculations

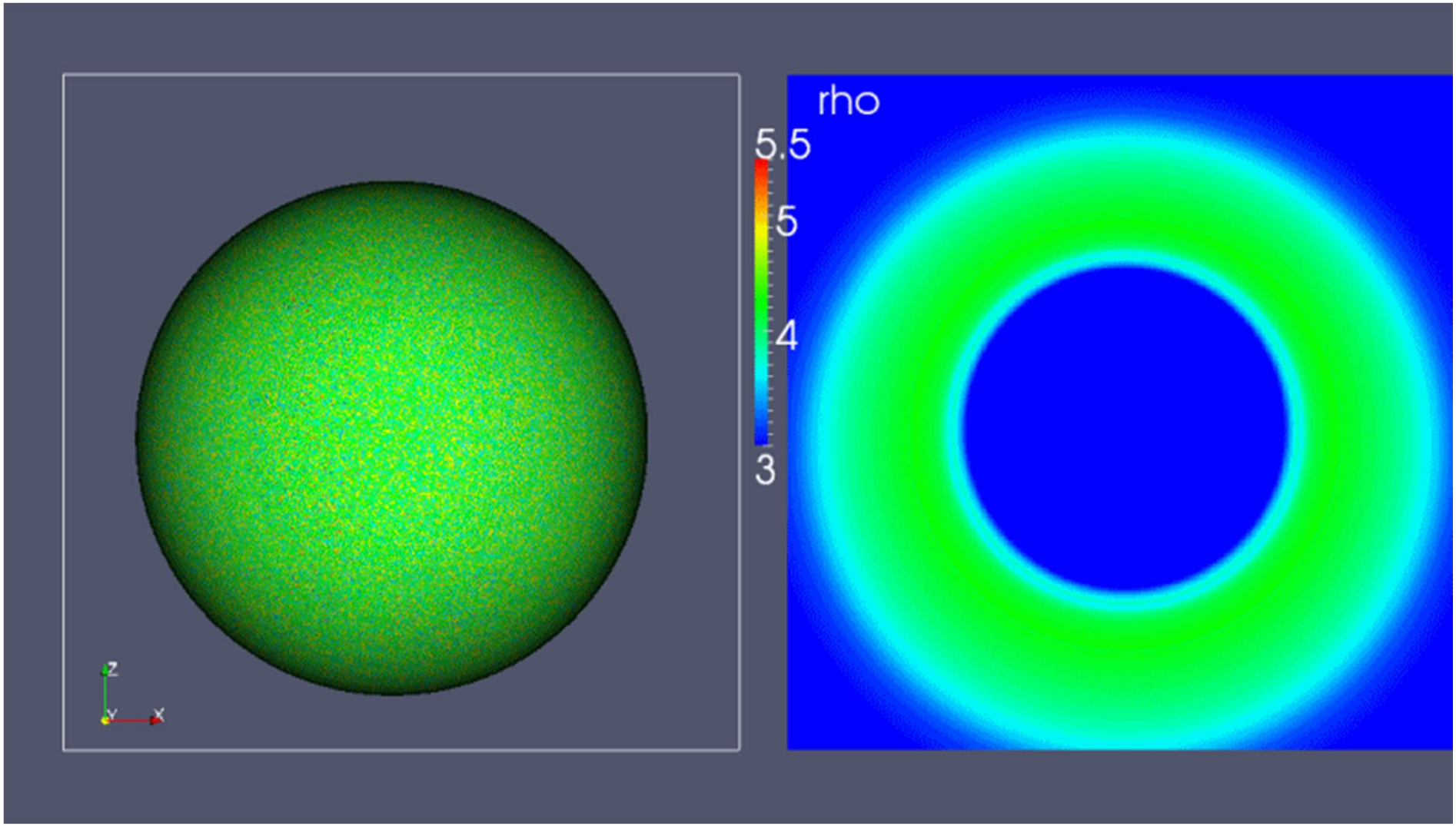
- Data analysis
- Simulations of theoretical models

An example of complex calculations – Inertial Confinement Fusion



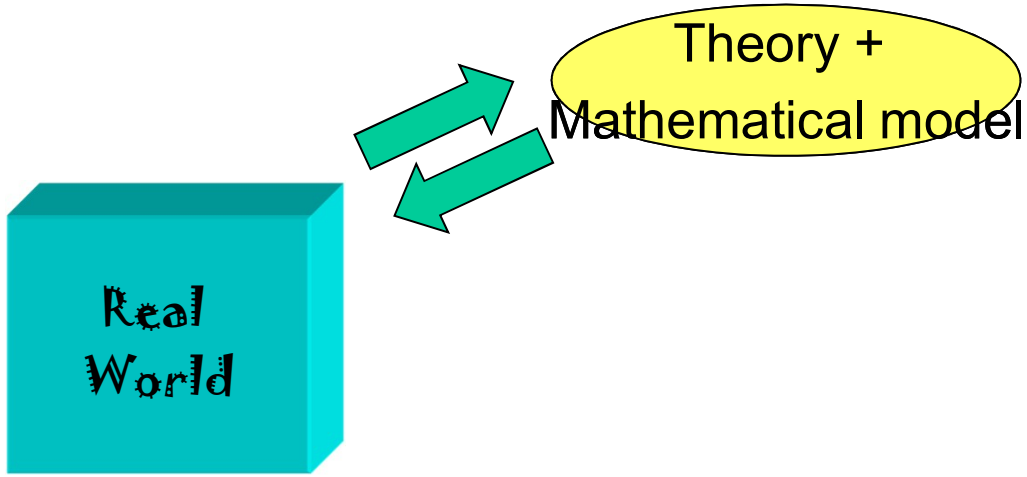
National Ignition Facility, California

Can we make this pellet of deuterium-tritium hotter than the centre of the Sun and > 100x water density?

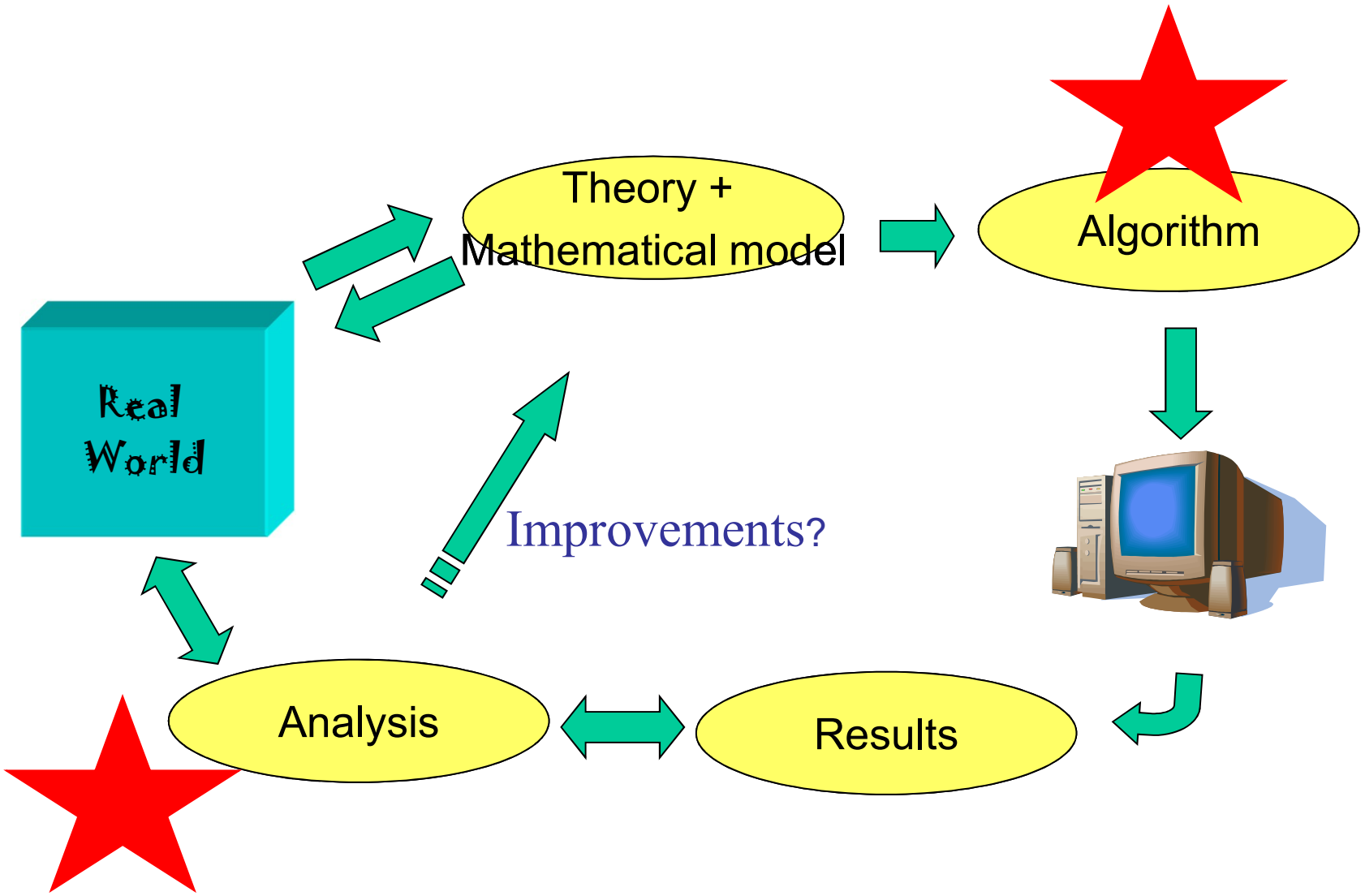


A single simulation takes thousands of computer cores running for weeks to produce the data for this gif

Bigger Picture



Bigger Picture



Year 1 Computing course: technical contents

Aim: to be able to use Python to analyse your lab data and simulate physical scenarios.

- Session 1: Manipulating arrays and plotting data
- Session 2: Error bars and fitting data
- Session 3: Logic statements, loops, and functions
- Session 4: Numerical modelling
- Computing Project: Using all your skills!

Lecture 1 Part 3

1. Course Outline
2. Scientific Computing
   - What is it?
   - Why do we do it?
3. The tools that we will use
   - Python
   - Jupyter Notebooks
   - Spyder IDE
4. Tips for being an effective programmer

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

What is Python?

Python...

- is a high-level programming language
- is a tool that interprets instructions you give a computer and makes the computer carry out those instructions

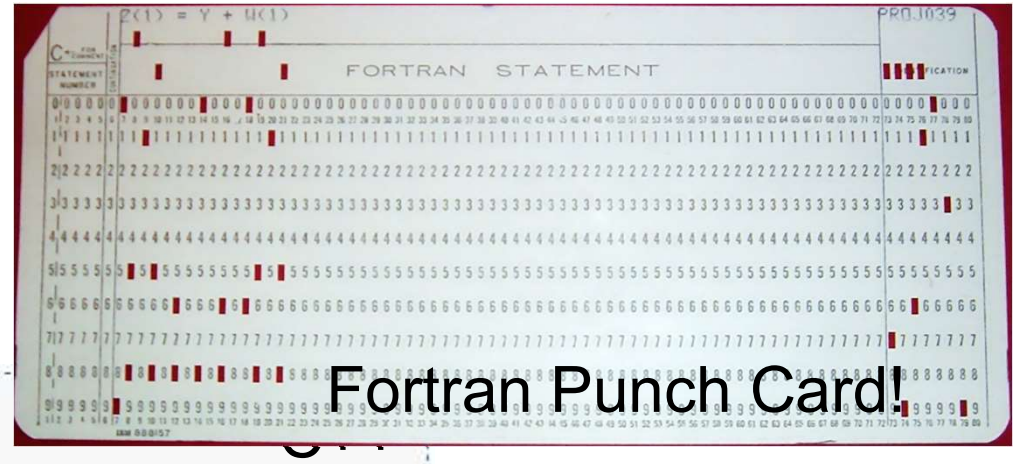
Why Python?

Python is:

- an industry standard
- easy and clear
- high level (many operations in few commands)
- very shallow learning curve (able to do a lot in little time)
- Applicable to all areas of STEM and beyond
- Free!

Hello World!

In various languages



Fortran Punch Card!

```
#include <iostream>

int main()
{
  std::cout << "Hello World!" << std::endl;
  return 0;
}
```

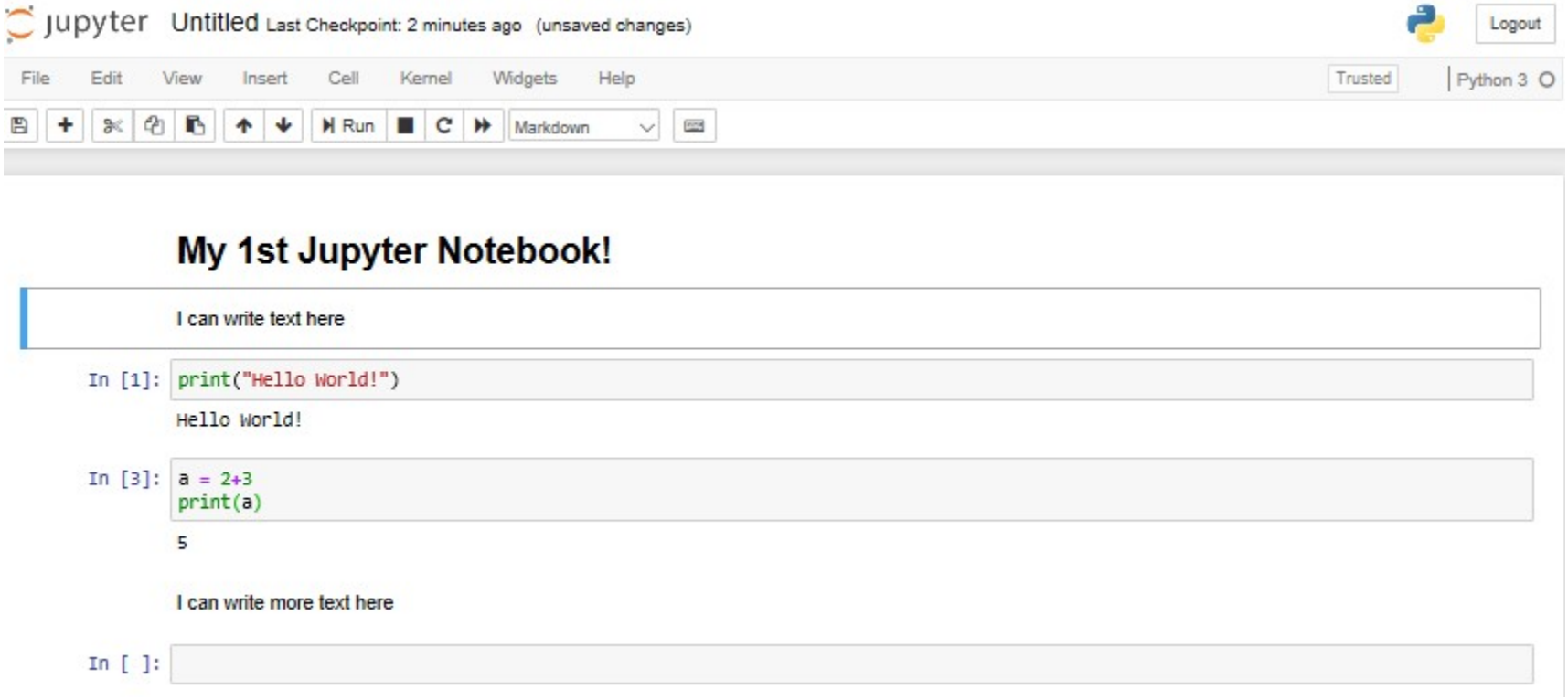
Java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

http://en.wikipedia.org/wiki/Hello_world_program_examples

Jupyter Notebooks – an Interactive lab script

- The worksheets are written in Jupyter Notebooks
- Jupyter Notebooks allow code to be mixed with text, images, equations, ...
- They are a great tool for creating a narrative



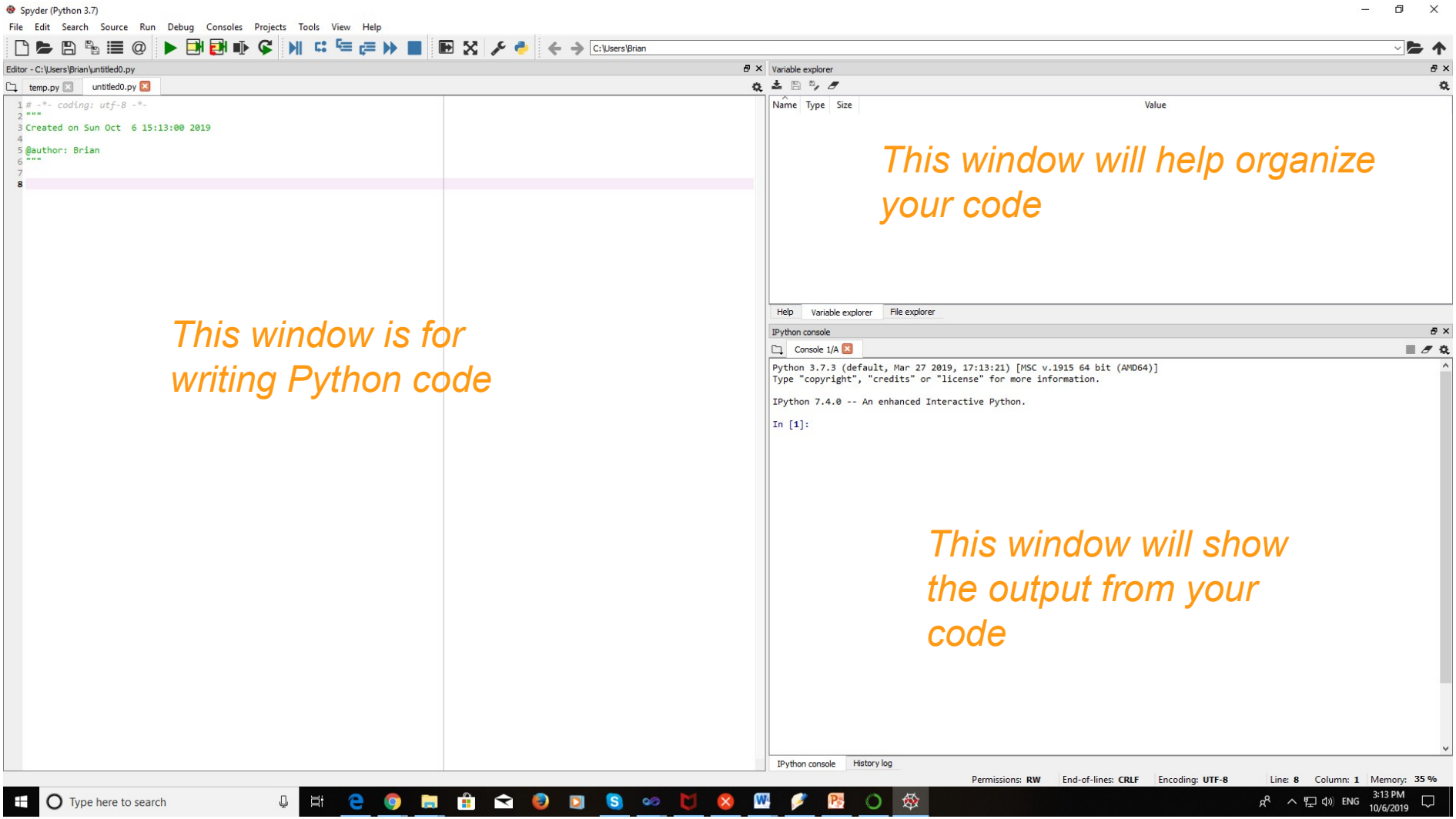
Jupyter Notebook Layout

- There are two notebooks per session – a core worksheet and an advanced worksheet
- You should ALWAYS start on the core worksheet first
- Exercises in core worksheet are split into two colours – green and yellow
- You should complete ALL green exercises in the worksheet before attempting the yellow ones

Spyder IDE – from Lab session 3 onwards

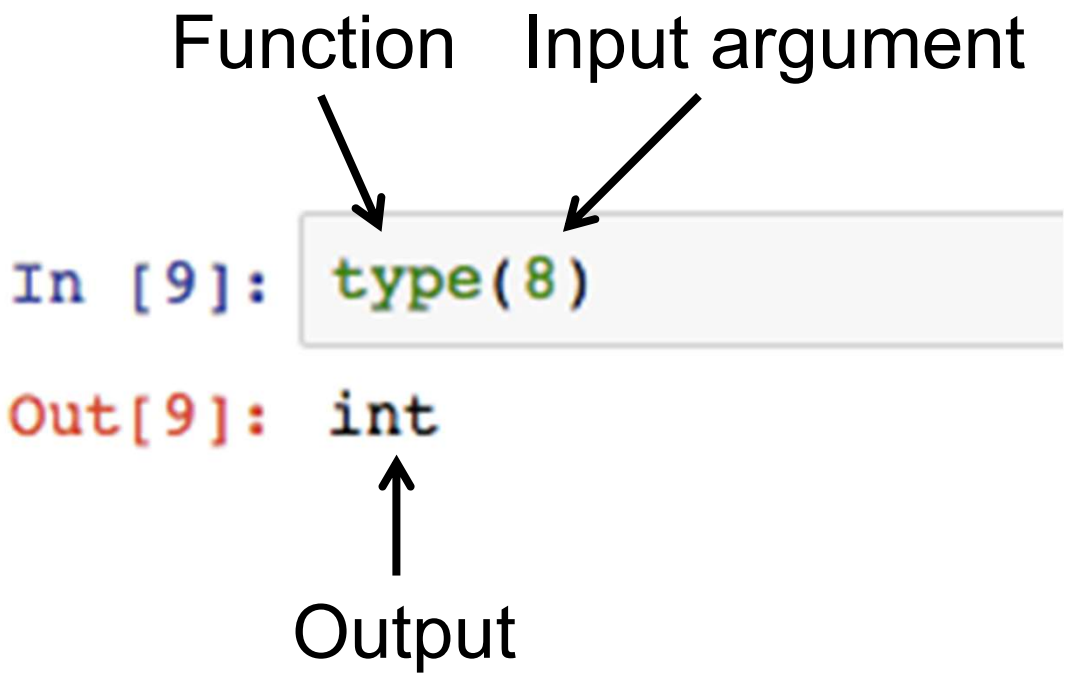
- Jupyter Notebooks are not suitable for writing long & complex code, or when we want to reuse pieces of code
- An IDE (Integrated Development Environment) makes it easy to build, modify, test, debug our code
- Many IDEs are available for Python – we will use Spyder

Spyder IDE – from Lab session 3 onwards



Functions

Functions are pre-built pieces of code that perform a specific task on given input and return output



Who builds these functions?

- Python developers (in-built functions)
- The community (packages)
- You! (Session 3)

Lecture 1 Part 4

1. Course Outline
2. Scientific Computing
   - What is it?
   - Why do we do it?
3. The tools that we will use
   - Python
   - Jupyter Notebooks
   - Spyder IDE
4. Tips for being an effective programmer

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

How to be an effective programmer

Experiential Learning – “Hands-on”: It is easiest to learn programming by writing code, trying new things, etc.

In Labs, demonstrators are there to answer your questions

However: you will also be *learning how to learn*

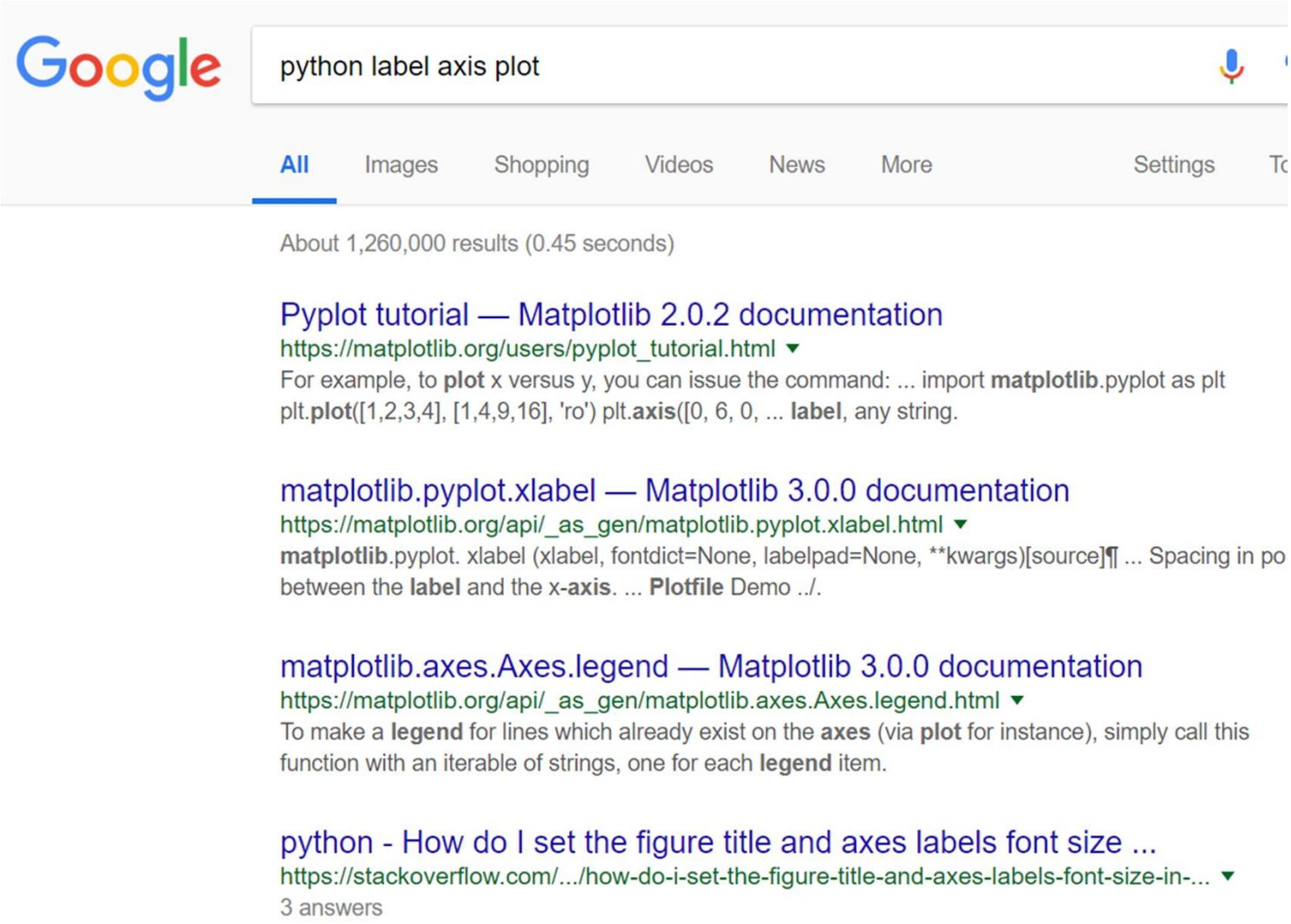
- Google
- In-built Python help
- Your peers
- ...

How to Google a problem

Effectively googling a problem is valuable skill to learn as a programmer

- Use keywords, not whole sentences
- Search for error messages if applicable
- Use quotations to keep words together
- Look at results from the documentations or websites such as stack overflow

Google Examples

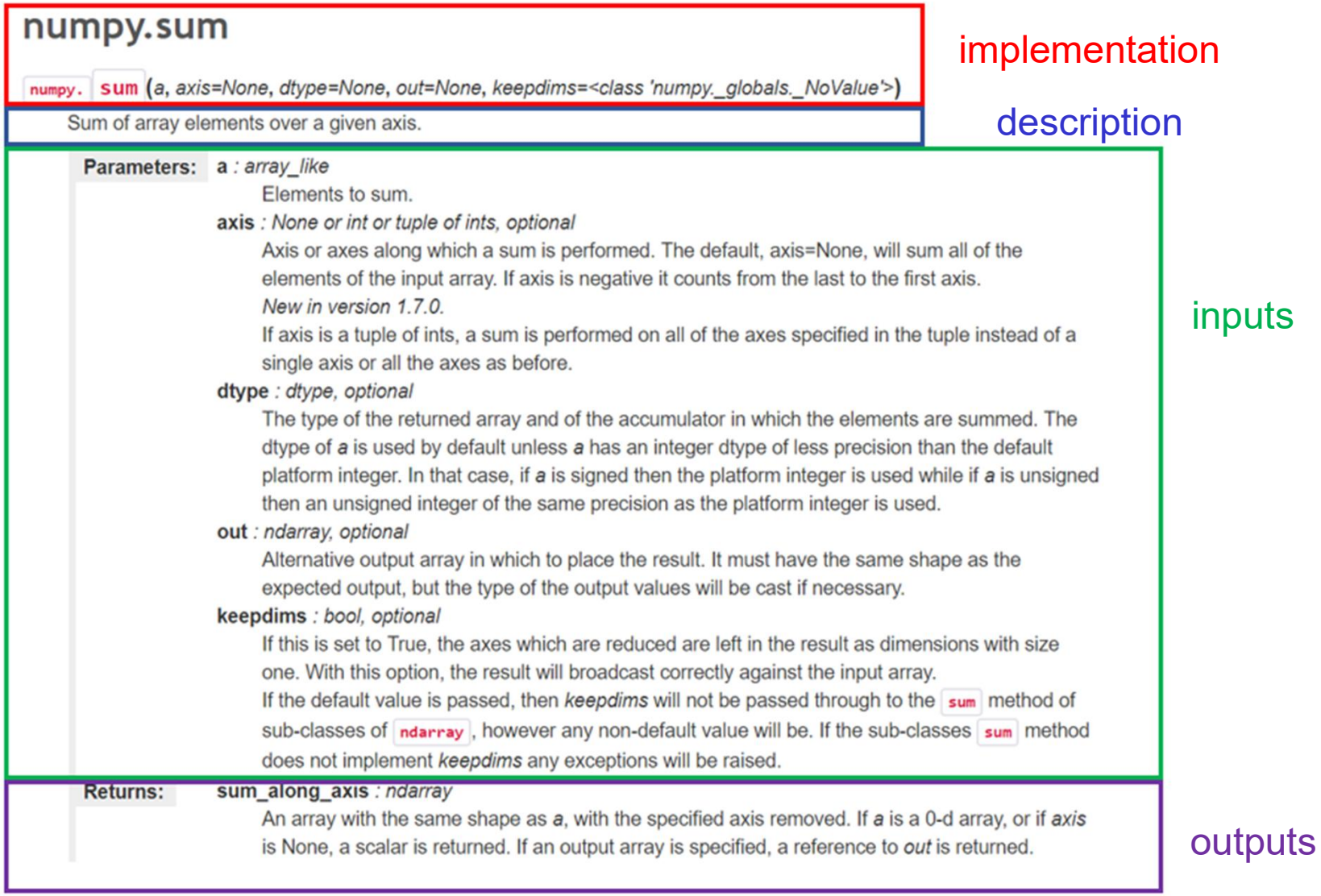


Python Help

All python functions comes with extensive documentation on how to use them

- You can access this directly by using the inbuilt help command in python – This will describe the functionality of what you are trying to use
- Alternatively you can go onto the documentation webpage to look at it in your browser

Documentation



Computing & Plagiarism

- A good programmer will re-use code that others have written . . .
- But beware! The plagiarism policy applies to code in the same way that it applies to other courses.
- If you use code written by your colleagues, or that you find from an online resource, then you should attribute it.
- Computing Projects will be checked for plagiarism.
- But remember – collaboration is great way to learn to code!

Questions?

Ask a demonstrator in Computing Lab sessions

Email:

Dr. Brian Appelbe **b.appelbe07@imperial.ac.uk**
Dr. Aidan Crilly **a.crilly16@imperial.ac.uk**

Have Fun!

Imperial College London

Computing - Lecture 2

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Dr. Aidan Crilly

a.crilly16@imperial.ac.uk

Lecture 2

1. Review of Computing Session 1
2. Some Programming Tips
   - Reading error messages
   - Commenting your code
3. Fitting data using Python

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Lecture 2 – Part 1

1. Review of Computing Session 1
2. Some Programming Tips
   - Reading error messages
   - Commenting your code
3. Fitting data using Python

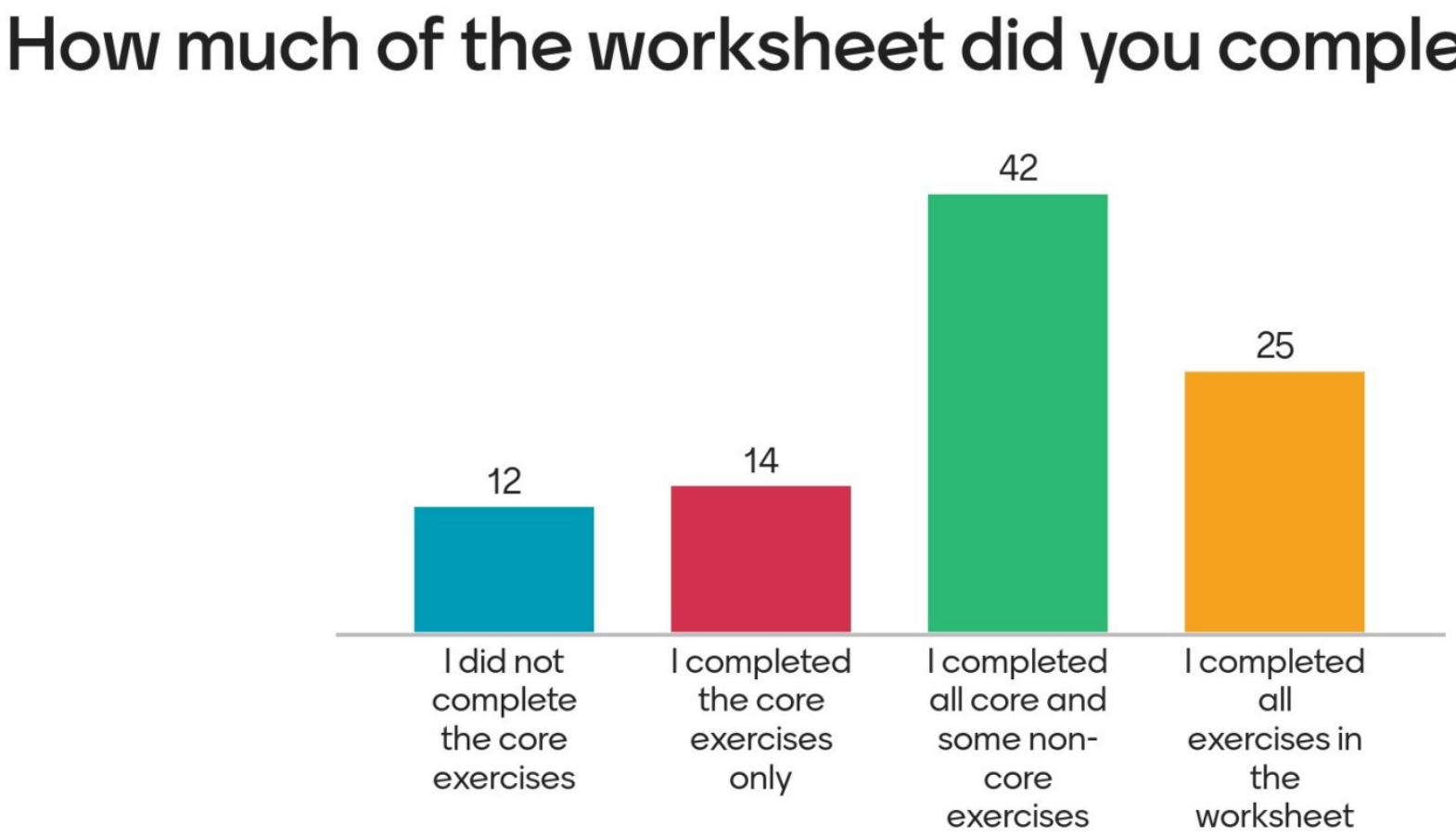
Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

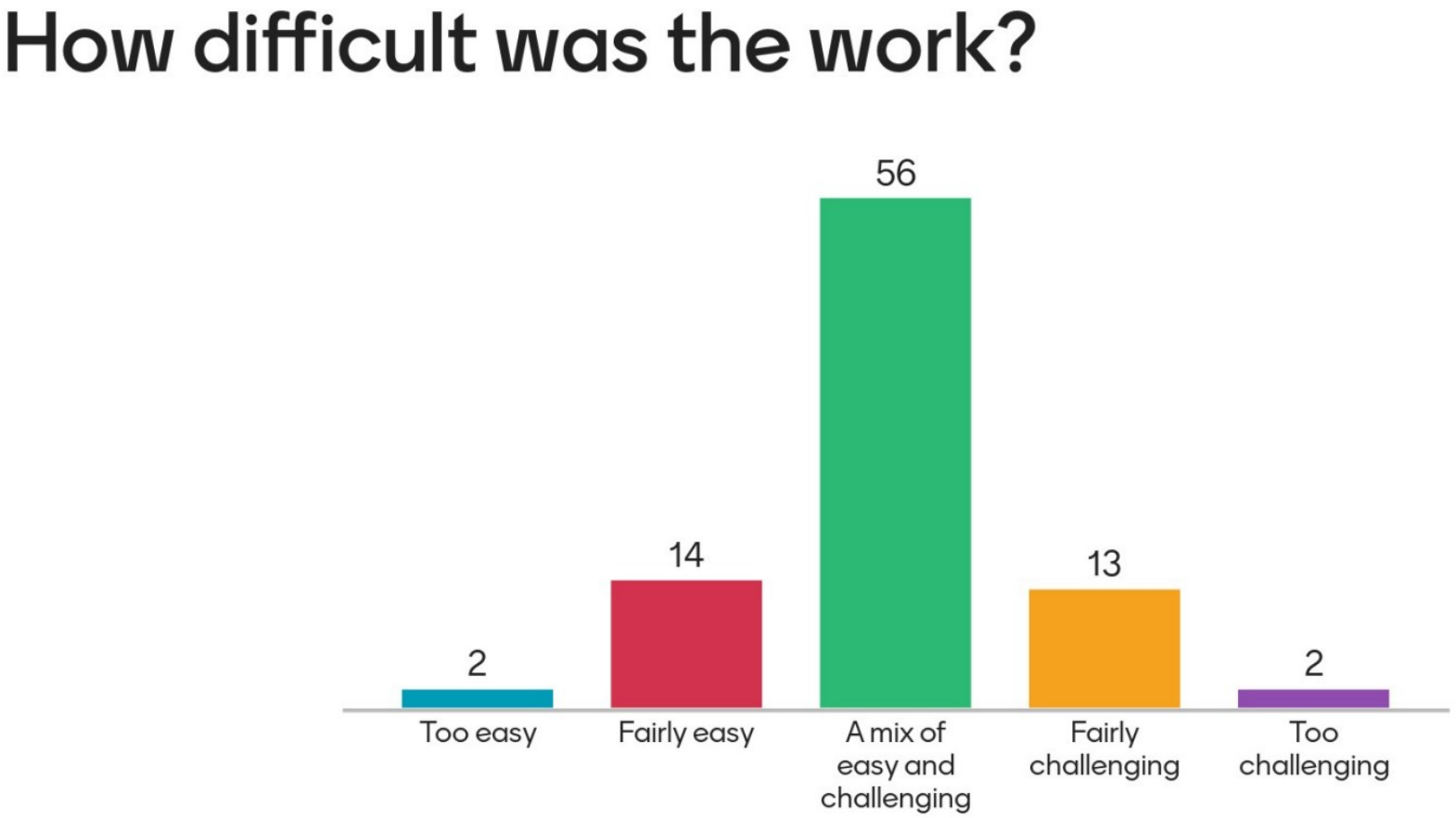
Announcements

- Please complete Mentimeter polls (your feedback is anonymous!).
- Take a break during the Computing session.
- You will need to do homework to maximize your learning.
- Example solutions for last week's lab sessions are now on blackboard.

Your Feedback - Mentimeter Result



Your Feedback - Mentimeter Result



Your Feedback - Mentimeter Result

- It’s normal to find some exercises difficult.
- It’s normal to have some exercises still to finish by the end of the session.
- The optional exercises will help with your labs and it is recommended that you look at them as well.
- Worksheets contain a lot of information – this is so that you can work on them independently and refer back to them later.

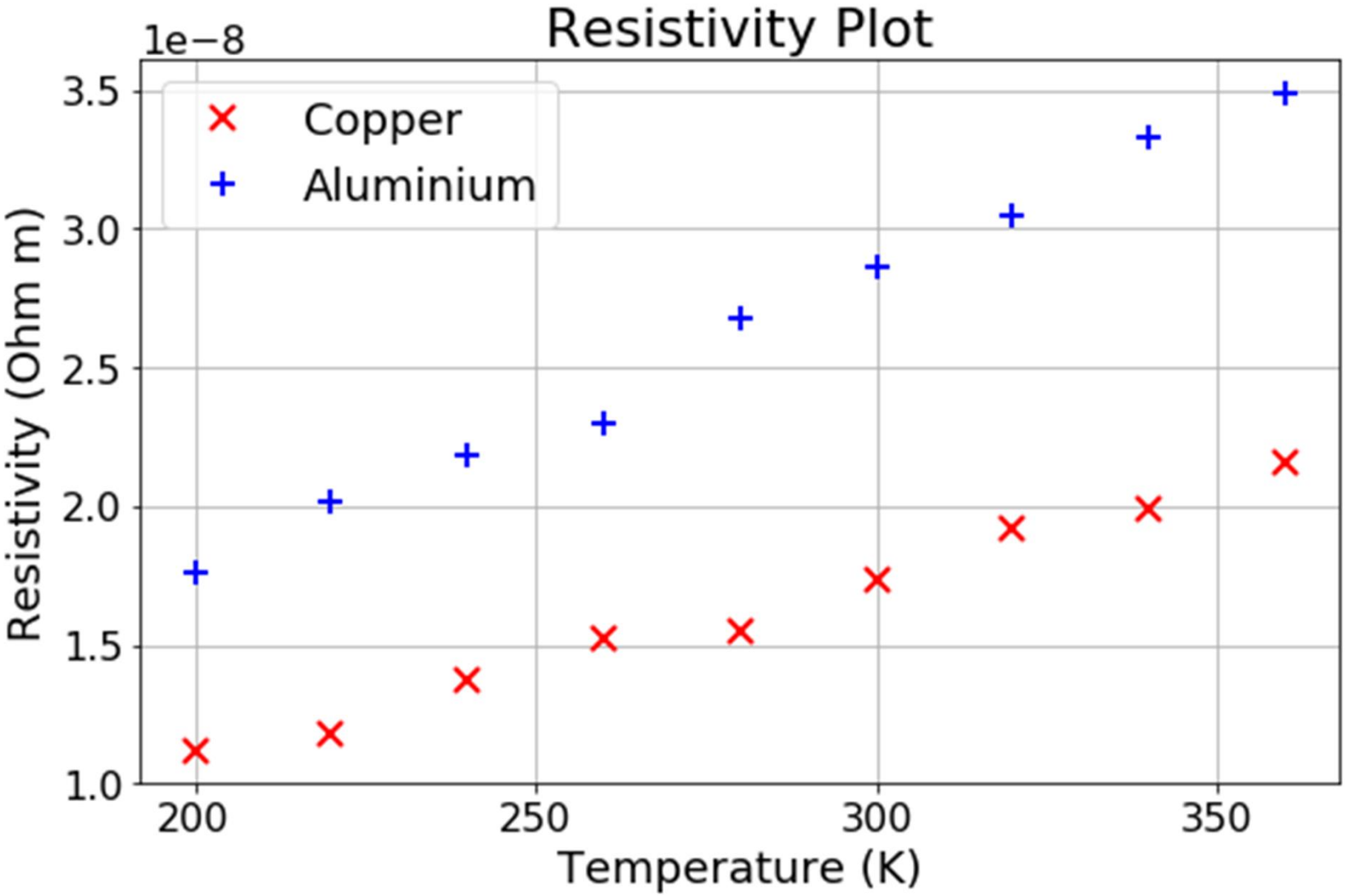
File Management

- It’s important to keep your Computing files organised.
- Have a separate folder for Computing and/or each Computing session.
- When using Jupyter Notebooks your files need to be in the Jupyter directory.
- For college computers this is your H:drive.
- On your own computer this is probably in your user directory.
- You can transfer your files using OneDrive, a memory key, emailing a copy to yourself, etc.

Session 1 Learning Objectives

- Create variables and carry out arithmetic operations on them.
- Import packages (e.g. NumPy, Matplotlib).
- Create and manipulate arrays, and do basic statistics on them.
- Read in a data file and save your data to an output file.
- Create line plots and scatter plots.
- Amend the plot format to create a figure of a publishable standard.
- Commenting your code.

Recap Session 1 – plotting your data



Year 1 Computing course: aims and structure

Aim: to be able to use Python to analyse your lab data and simulate physical scenarios.

- Session 1: Manipulating arrays and plotting data
- Session 2: Fitting Data and error bars
- Session 3: Logic statements, loops, and functions
- Session 4: Numerical modelling
- Project: Using all your skills!

Year 1 Computing course: aims and structure

Aim: to be able to use Python to analyse your lab data and simulate physical scenarios.

- Session 1: Manipulating arrays and plotting data
- Session 2: Fitting data and error bars
- Session 3: Logic statements, loops, and functions
- Session 4: Numerical modelling
- Project: Using all your skills!

Lecture 2 – Part 2

1. Review of Computing Session 1
2. Some Programming Tips
   - Reading error messages
   - Commenting your code
3. Fitting data using Python

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Error Message Interpretation

- When using functions in session 1, you may have encountered error messages.
- These error messages contain a lot of technical information – these can be difficult to understand at first.
- But there are a few key areas that you can focus on to understand what went wrong.

Let's write some code to load & print resistivity data:

Temperature	Resistivity Cu	Resistivity Al
K	Ohm m	Ohm m
200	1.12E-08	1.76E-08
220	1.18E-08	2.02E-08
240	1.37E-08	2.18E-08
260	1.52E-08	2.30E-08
280	1.55E-08	2.68E-08
300	1.73E-08	2.86E-08
320	1.92E-08	3.05E-08
340	1.99E-08	3.33E-08
360	2.16E-08	3.49E-08

```
In [4]: import numpy as np
        T, R_Cu, R_Al = np.loadtxt("Data/Resistivity.csv",skiprows=1,delimiter=',',unpack=True)
        print(T)
        print(R_Cu)
        print(R_Al)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-4-e485198b01f8> in <module>
      1 import numpy as np
----> 2 T, R_Cu, R_Al = np.loadtxt("Data/Resistivity.csv",skiprows=1,delimiter=',',unpack=True)
      3 print(T)
      4 print(R_Cu)
      5 print(R_Al)
```

Where it occurred in your code

```
C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\npyio.py in loadtxt(fname, dtype, comments, delimiter, converters, skiprow
s, usecols, unpack, ndmin, encoding, max_rows)
```

Temperature	Resistivity Cu	Resistivity Al
K	Ohm m	Ohm m
200	1.12E-08	1.76E-08
220	1.18E-08	2.02E-08
240	1.37E-08	2.18E-08
260	1.52E-08	2.30E-08
280	1.55E-08	2.68E-08
300	1.73E-08	2.86E-08
320	1.92E-08	3.05E-08
340	1.99E-08	3.33E-08
360	2.16E-08	3.49E-08

```
_data(chunk_size)

store
ers, vals)]

tcomp>(.0)

store
ers, vals)]

tconv(x)

    796     typ = dtype.type
```

Where it occurred in the function

```
ValueError: could not convert string to float: 'K'
```

The type of error that occurred

Commenting your Code

- The comment symbol in Python is #
- Python will not try to execute any code that appears on a line after #

Comments are useful for:

1. Making the code easier to understand
2. Removing executable lines of code for testing

Making Code Easier to Understand

- Describe what code is supposed to do
- Define variables
- For Physics: units!

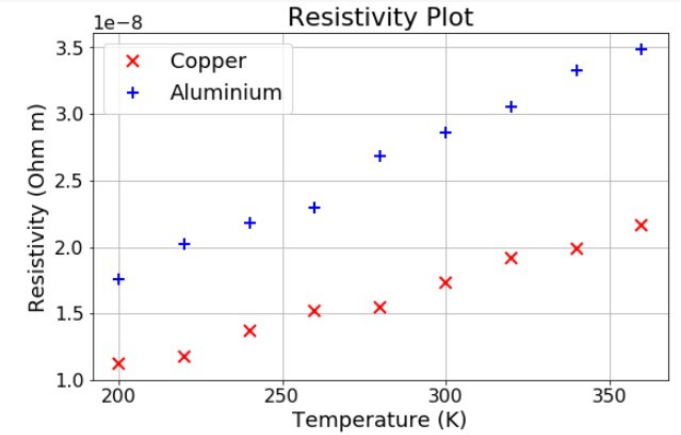
In [1]:
```
#This snippet of code calculates the hypotenuse of a triangle,given the height and width
a=3 #Height in cm
b=4 #Width in cm
c=sp.sqrt(a**2+b**2) # Calculate length of hypotenuse (in cm) using Pythagoras theorem
# Print the result:
print("The hypotenuse of a triangle with width ", a, " cm and height ", b, " cm is ", c, " cm ")
```

- Short descriptions should be included as comments in code
- Longer descriptions (& Flow Charts!) should be in your Computing Notebook

Testing Code

- Use comments instead of deleting code

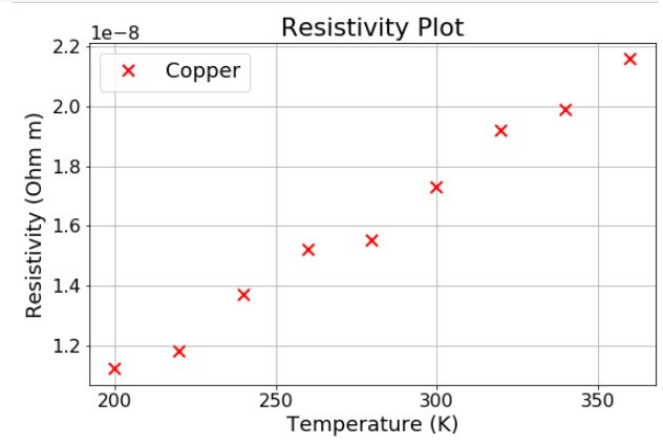
In [11]:
```
# Plot parameters: experiment with different values
params = {
   'axes.labelsize': 18,# Set to size 18
   'font.size': 18,# Set to size 18
   'font.family': 'sans-serif', # Select font group
   'font.serif': 'Arial', # Select font
   'legend.fontsize': 18,# Set to size 18
   'xtick.labelsize': 16,# Set to size 16
   'ytick.labelsize': 16, # Set to size 16
   'figure.figsize': [8.8, 8.8/1.618] # Set to [6,4] to start with
}
plt.rcParams.update(params)

plt.grid() # Add grid to plot
plt.plot(T,R_Cu, 'x', mew=2, ms=10, color='red') # Plot the Copper resistivity data
plt.plot(T,R_Al, '+', mew=2, ms=10, color='blue') # Plot the Aluminum resistivity data
plt.xlabel("Temperature (K)") # x axis is Temperature in Kelvin
plt.ylabel("Resistivity (Ohm m)") # y axis is Resistivity in Ohm m
plt.title("Resistivity Plot") # Title of plot
plt.legend(["Copper", "Aluminium"]); # Legend
plt.xticks(sp.arange(200, 400, 50)) # Location of x ticks
plt.savefig("Output/Resistivity_plot_improved.png") # save plot to my output folder
```



Testing Code

- Use comments instead of deleting code

In [12]:
```
# Plot parameters: experiment with different values
params = {
   'axes.labelsize': 18,# Set to size 18
   'font.size': 18,# Set to size 18
   'font.family': 'sans-serif', # Select font group
   'font.serif': 'Arial', # Select font
   'legend.fontsize': 18,# Set to size 18
   'xtick.labelsize': 16,# Set to size 16
   'ytick.labelsize': 16, # Set to size 16
   'figure.figsize': [8.8, 8.8/1.618] # Set to [6,4] to start with
}
plt.rcParams.update(params)

plt.grid() # Add grid to plot
plt.plot(T,R_Cu, 'x', mew=2, ms=10, color='red') # Plot the Copper resistivity data
#plt.plot(T,R_Al, '+', mew=2, ms=10, color='blue') # Plot the Aluminum resistivity data
plt.xlabel("Temperature (K)") # x axis is Temperature in Kelvin
plt.ylabel("Resistivity (Ohm m)") # y axis is Resistivity in Ohm m
plt.title("Resistivity Plot") # Title of plot
#plt.legend(["Copper", "Aluminium"]); # Legend
plt.legend(["Copper"]); # Legend
plt.xticks(sp.arange(200, 400, 50)) # Location of x ticks
plt.savefig("Output/Resistivity_plot_improved.png") # save plot to my output folder
```



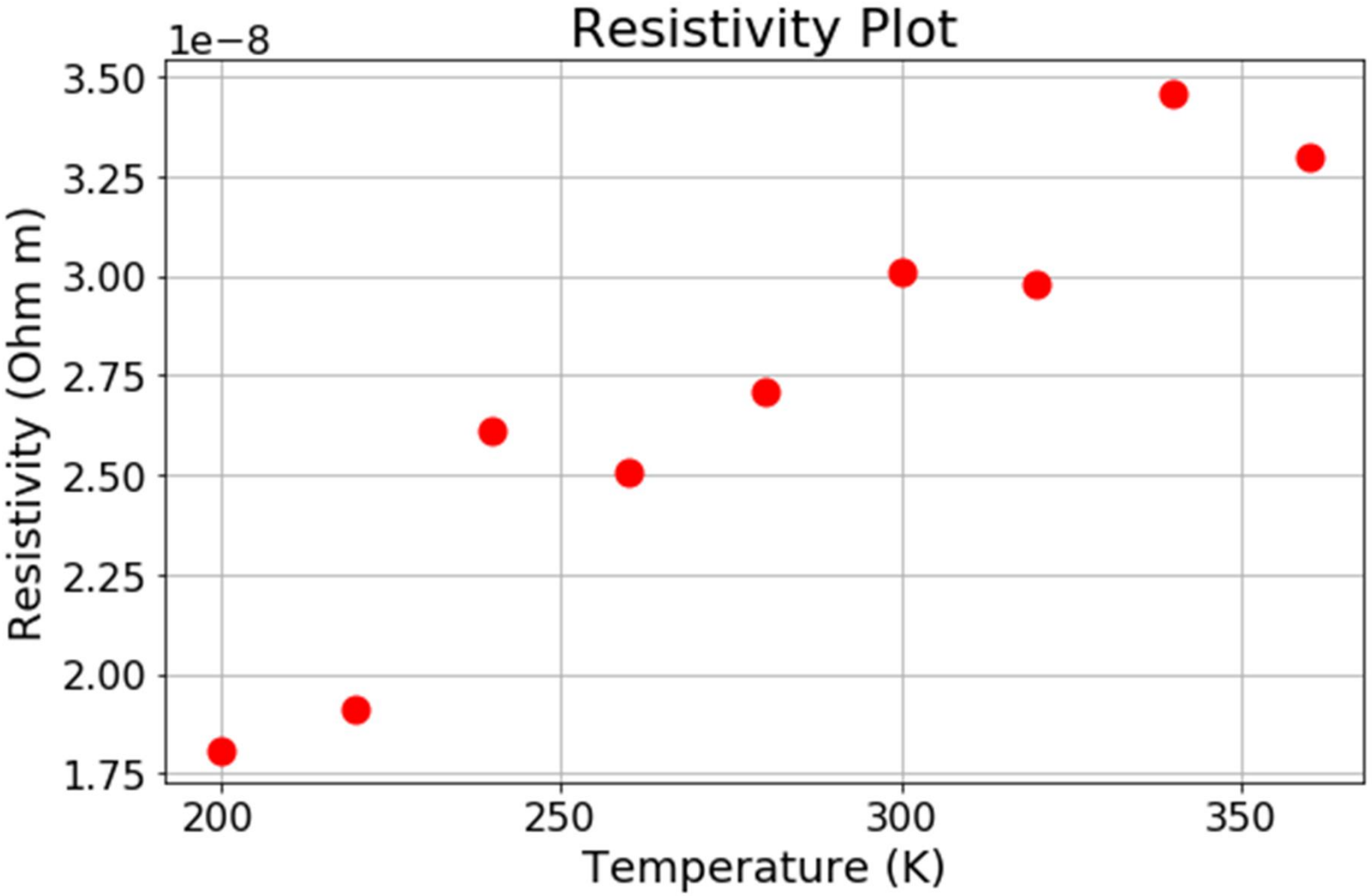
Lecture 2 – Part 3

1. Review of Computing Session 1
2. Some Programming Tips
   - Reading error messages
   - Commenting your code
3. Fitting data using Python

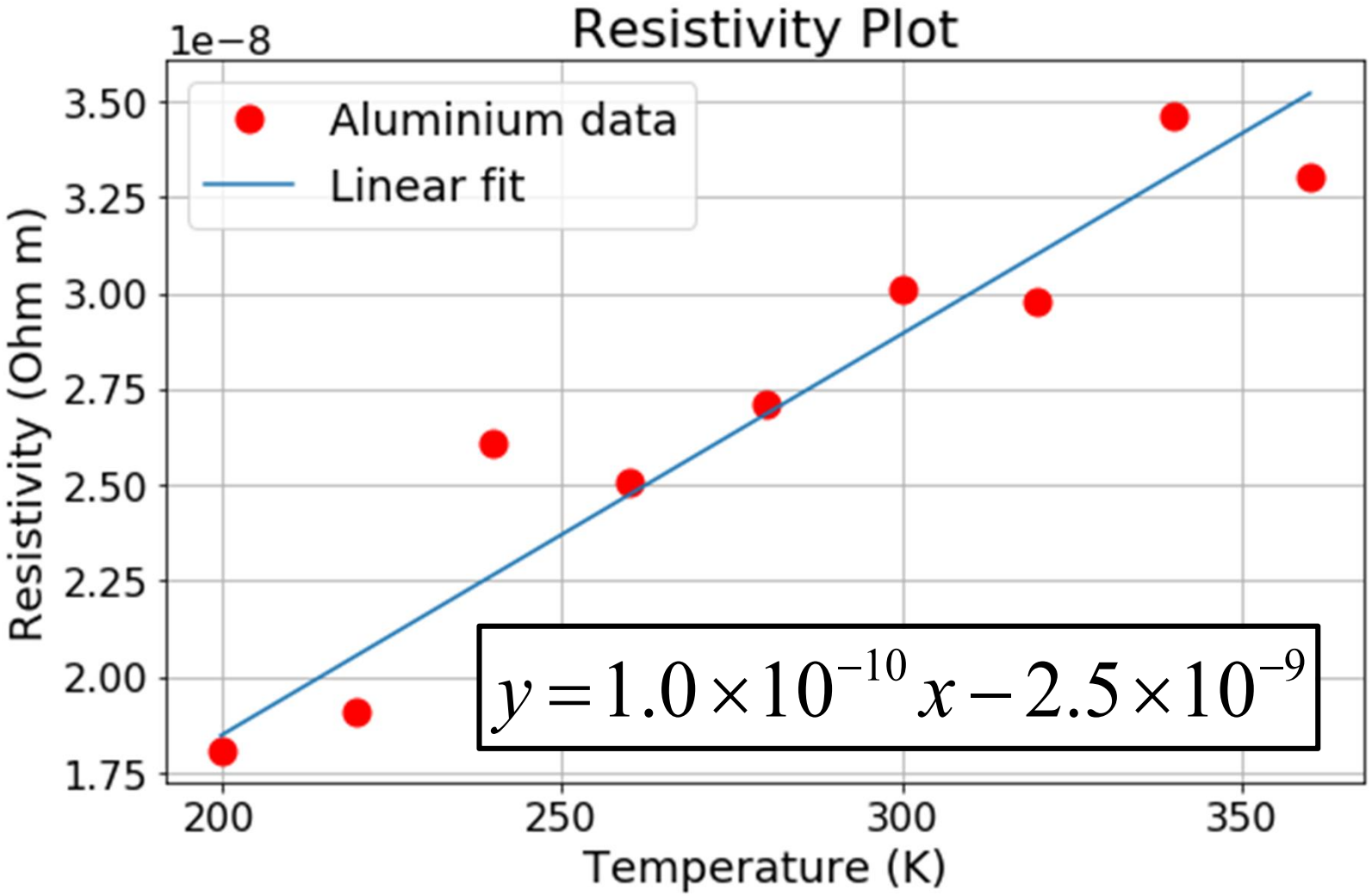
Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

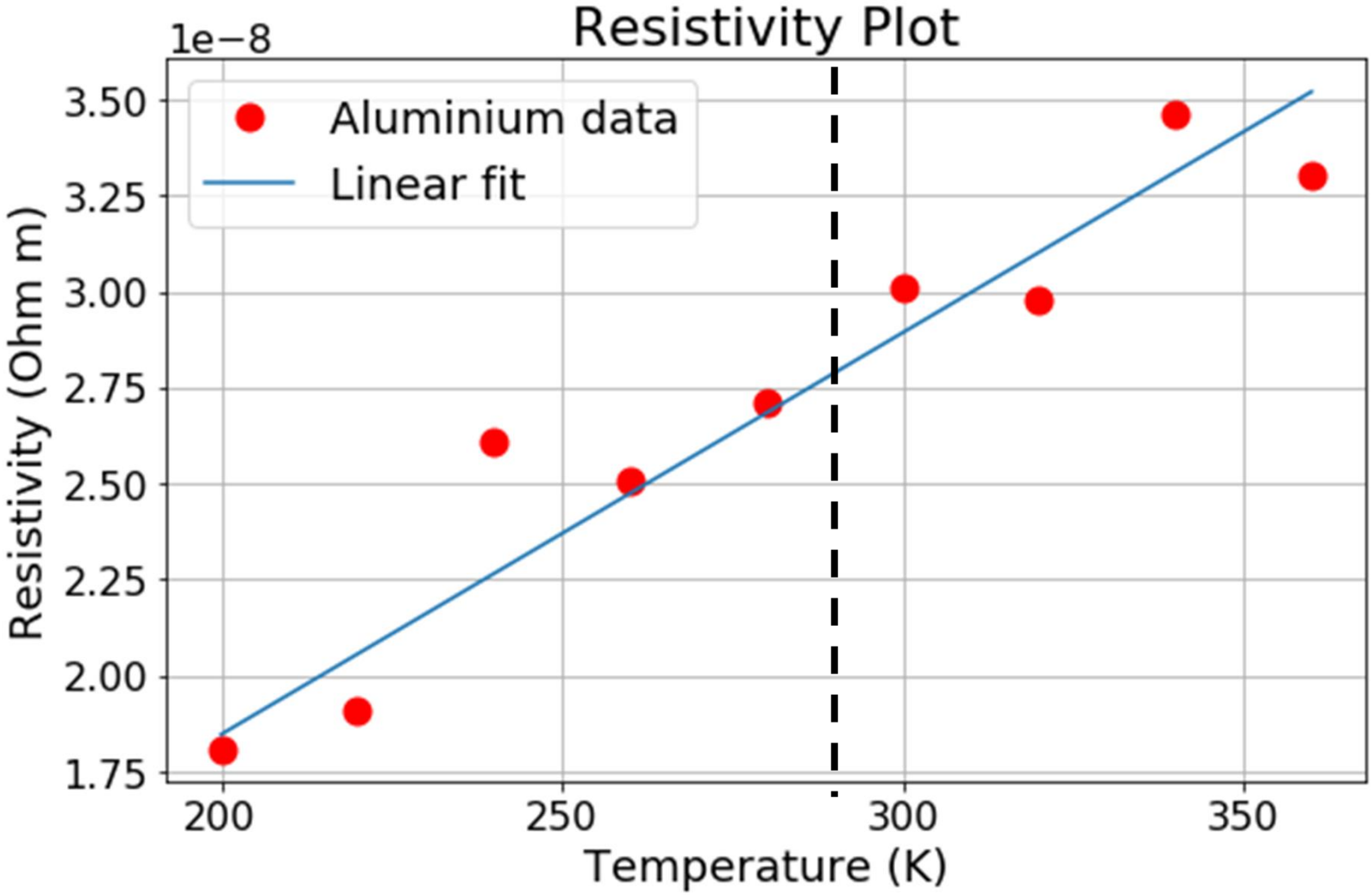
Session 2 – Fitting your data



Fitting your data with a theoretical function



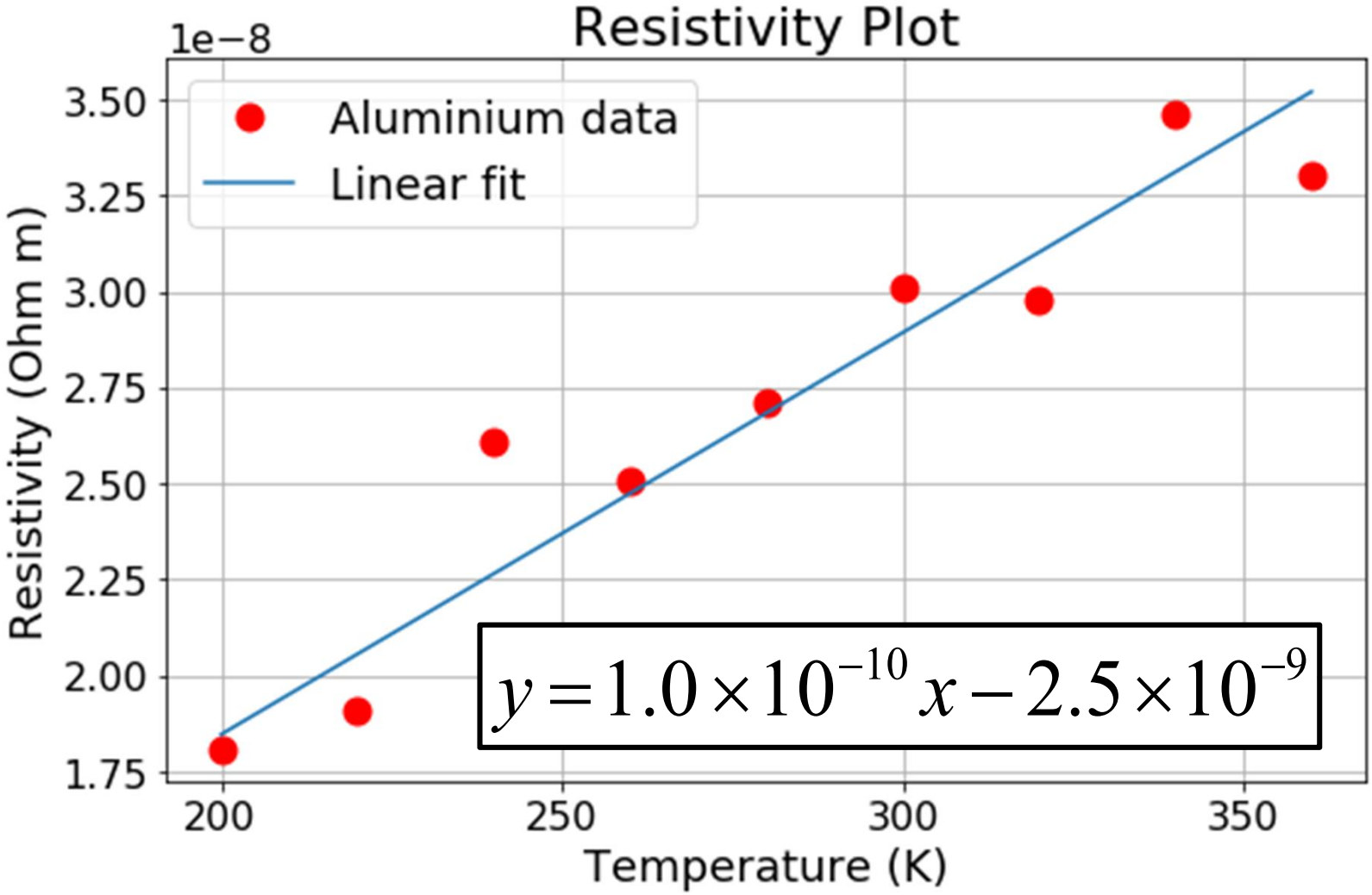
Why do data Fitting?
1. Prediction/interpolation/extrapolation



450 k

Why do data Fitting?

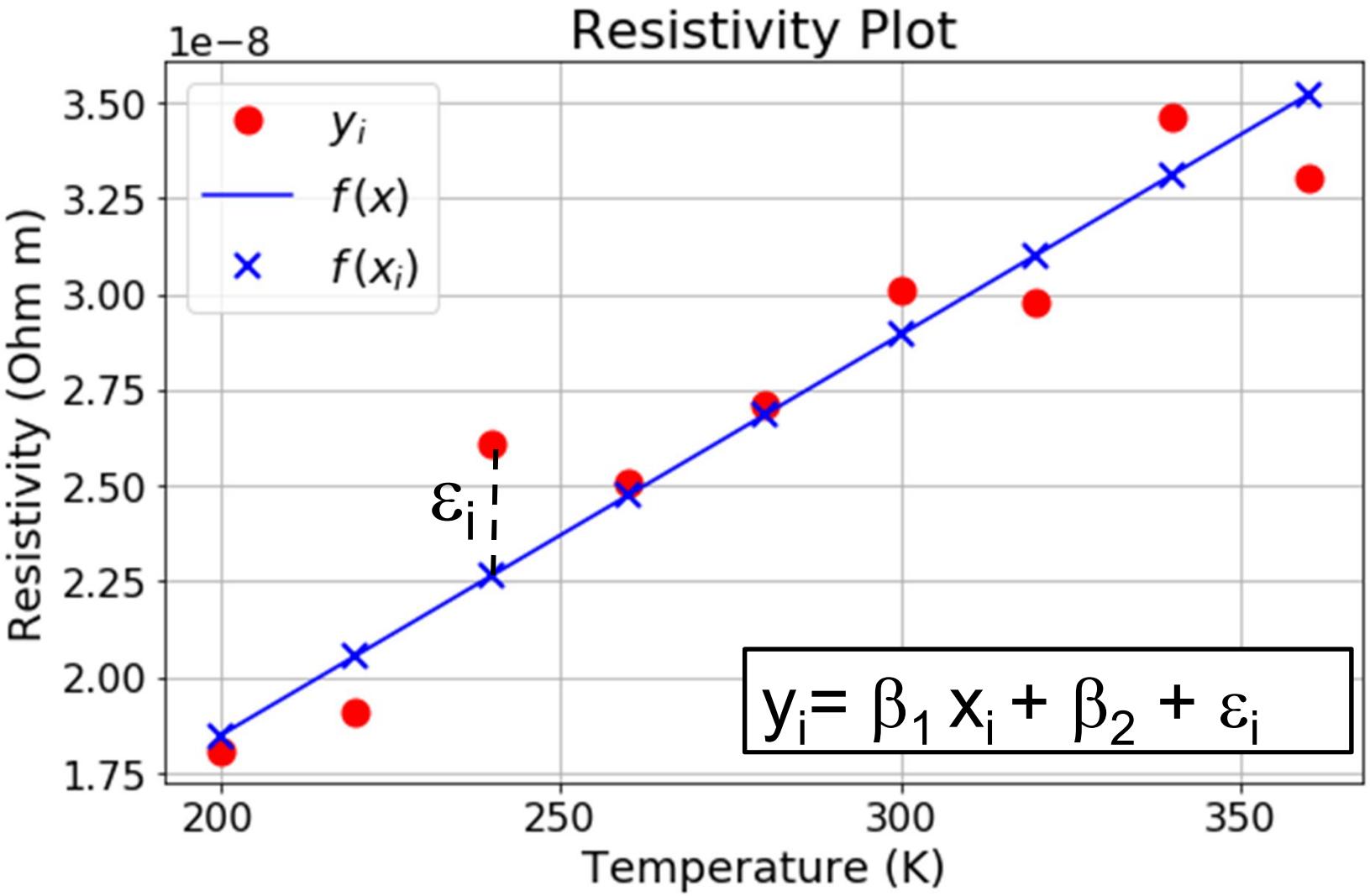
2. Establishing a relationship between dependent and independent variables



How does fitting work?

- For linear relation: $f(x) = \beta_1 x + \beta_2$

How does fitting work?



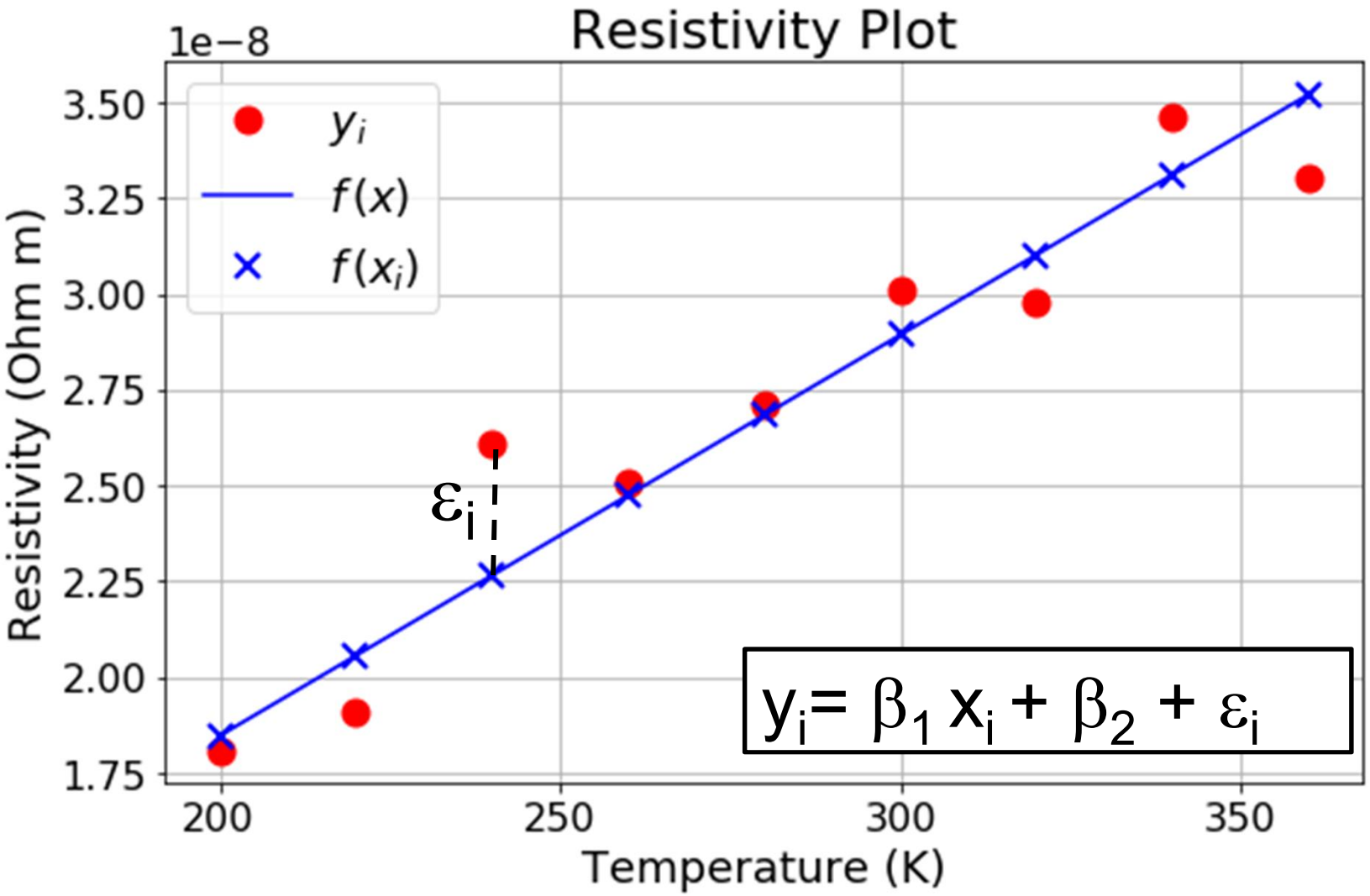
How does fitting work?

- For linear relation: $f(x) = \beta_1 x + \beta_2$
- Data points: x_i and y_i: $y_i = \beta_1 x_i + \beta_2 + \varepsilon_i$

Coefficients Residuals

- Least-squares method minimizes $SS_{err} = \sum_i \varepsilon_i^2$

How does fitting work?



Fitting a polynomial in Python: polyfit()

Boolean data type:
True, 1
False, 0

```
import numpy as np

T, R_Al = np.loadtxt('noisy_data.txt',unpack=1)
errors = 0.05*R_Al

fit,cov = np.polyfit(T,R_Al,1,w=1/errors,cov=True)
```

Covariant matrix

Fit parameters

Cov matrix yes/no

Weights

Order of the polynomial

Dependent variable (y)

Independent variable (x)

numpy.polyfit

numpy.**polyfit**(*x, y, deg, rcond=None, full=False, w=None, cov=False*) [source]

Least squares polynomial fit.

Fit a polynomial `p(x) = p[0] * x**deg + ... + p[deg]` of degree *deg* to points *(x, y)*. Returns a vector of coefficients *p* that minimises the squared error.

Parameters:

x : *array_like, shape (M,)*

x-coordinates of the M sample points `(x[i], y[i])`.

y : *array_like, shape (M,) or (M, K)*

y-coordinates of the sample points. Several data sets of sample points sharing the same x-coordinates can be fitted at once by passing in a 2D-array that contains one dataset per column.

deg : *int*

Degree of the fitting polynomial

rcond : *float, optional*

Relative condition number of the fit. Singular values smaller than this relative to the largest singular value will be ignored. The default value is len(x)*eps, where eps is the relative precision of the float type, about 2e-16 in most cases.

full : *bool, optional*

Switch determining nature of return value. When it is False (the default) just the coefficients are returned, when True diagnostic information from the singular value decomposition is also returned.

w : *array_like, shape (M,), optional*

Weights to apply to the y-coordinates of the sample points. For gaussian uncertainties, use 1/sigma (not 1/sigma**2).

cov : *bool, optional*

Return the estimate and the covariance matrix of the estimate If full is True, then cov is not returned.

Returns:

p : *ndarray, shape (deg + 1,) or (deg + 1, K)*

Polynomial coefficients, highest power first. If *y* was 2-D, the coefficients for *k*-th data set are in `p[:,k]`.

residuals, rank, singular_values, rcond

Present only if **full** = True. Residuals of the least-squares fit, the effective rank of the scaled Vandermonde coefficient matrix, its singular values, and the specified value of *rcond*. For more details, see **linalg.lstsq**.

V : *ndarray, shape (M,M) or (M,M,K)*

Present only if **full** = False and *cov `=True. The covariance matrix of the polynomial coefficient estimates. The diagonal of this matrix are the variance estimates for each coefficient. If y is a 2-D array, then the covariance matrix for the `k*-th data set are in `V[:,:,k]`

Warns:

RankWarning

The rank of the coefficient matrix in the least-squares fit is deficient. The warning is only raised if **full** = False.
The warnings can be turned off by

```
>>> import warnings
>>> warnings.simplefilter('ignore', np.RankWarning)
```

>>>

The output of polyfit(): fit parameters + errors

- The fit parameters: f(x) = β_1 x + β_2

$$y = 1.0 \times 10^{-10} x - 2.5 \times 10^{-9}$$

- The covariant matrix :

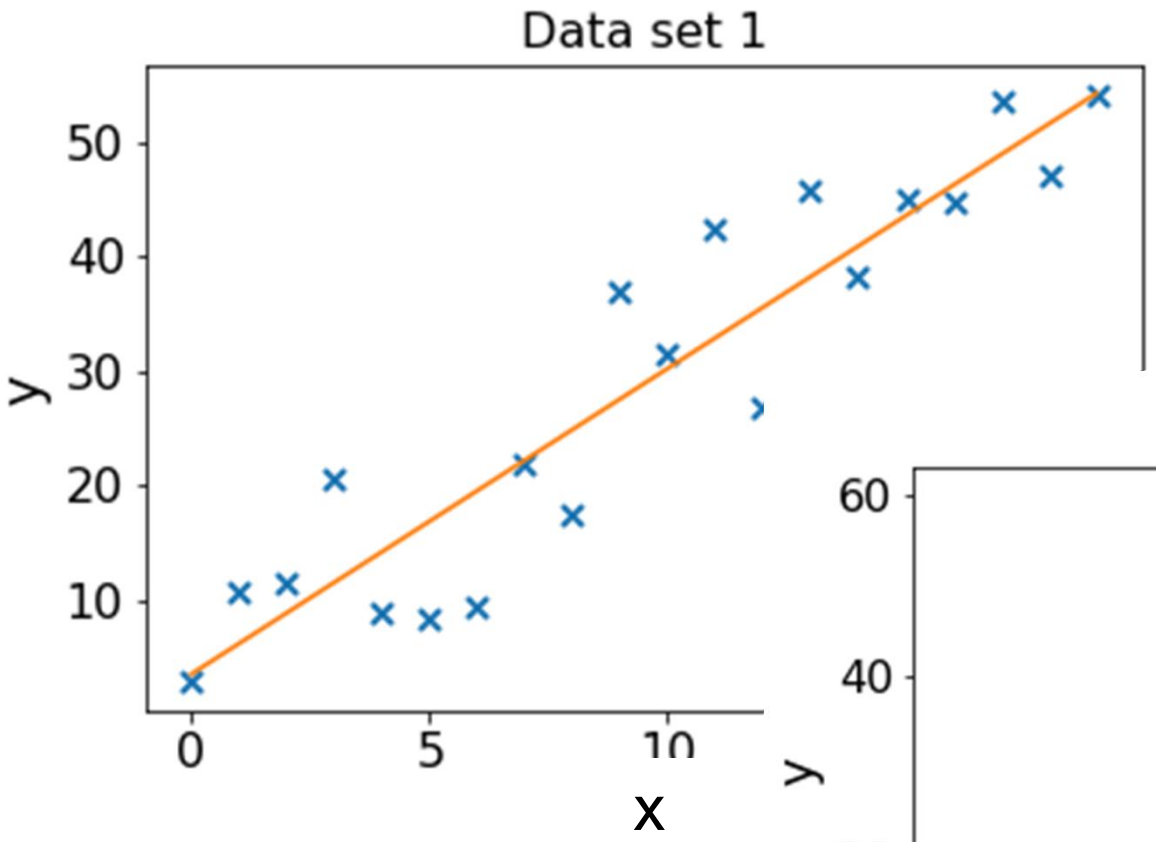
$$\begin{pmatrix} C_{00} & C_{10} \\ C_{01} & C_{11} \end{pmatrix}$$

$$\sigma_{\beta_1} = \sqrt{C_{00}}$$

$$\sigma_{\beta_2} = \sqrt{C_{11}}$$

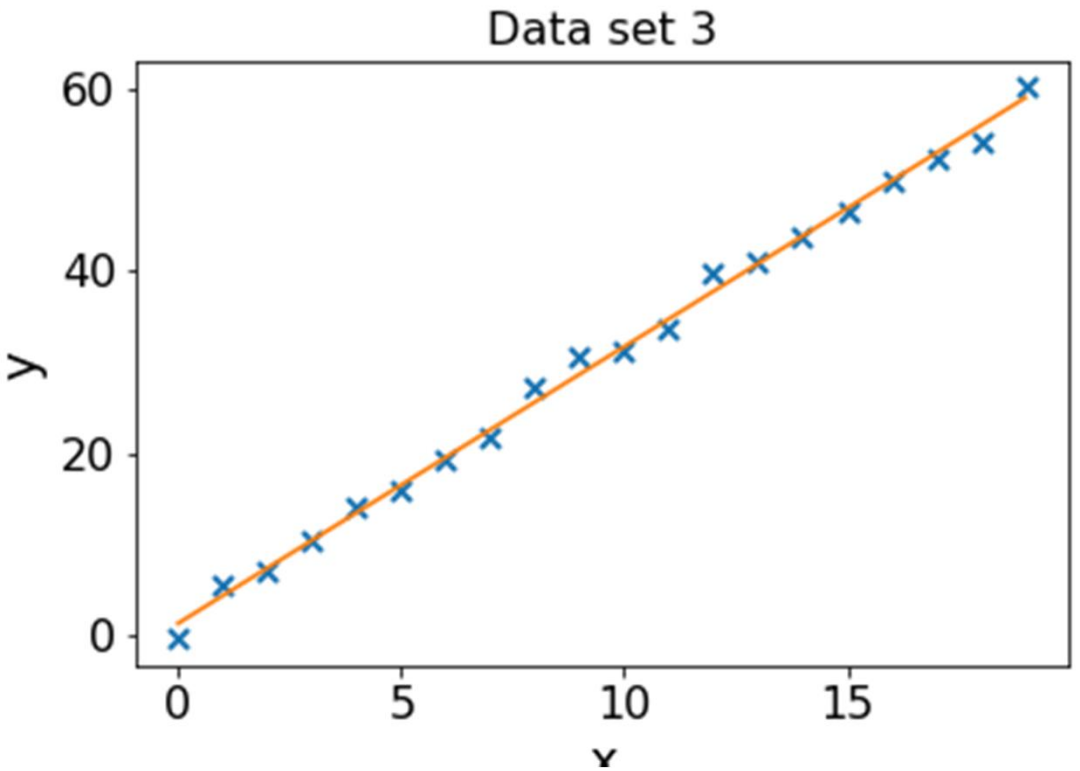
Uncertainties in the coefficients

The errors depend on the scatter in the data



$\beta_1 = 2.54 \pm 0.26$

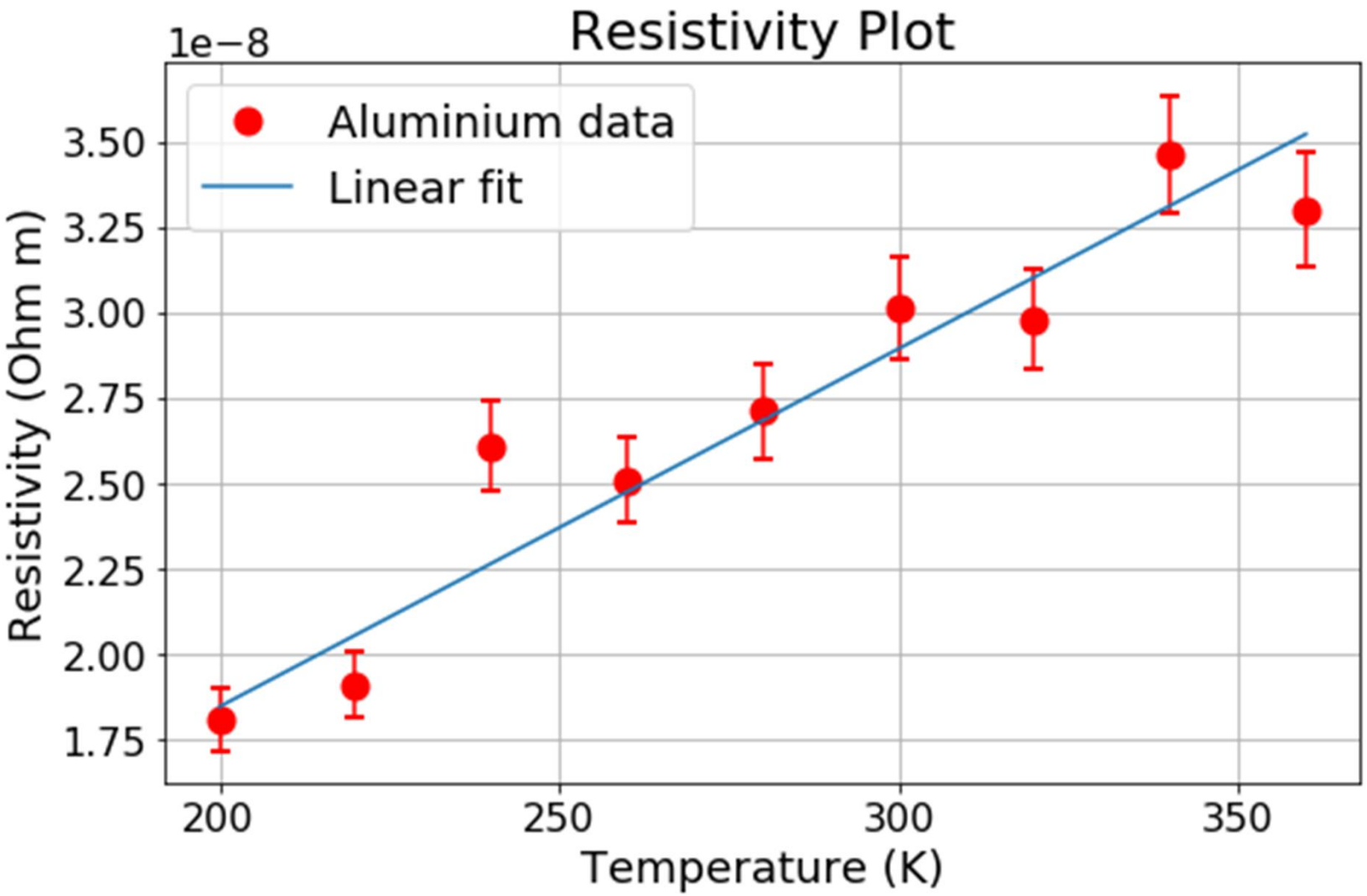
$\beta_2 = 3.56 \pm 2.84$



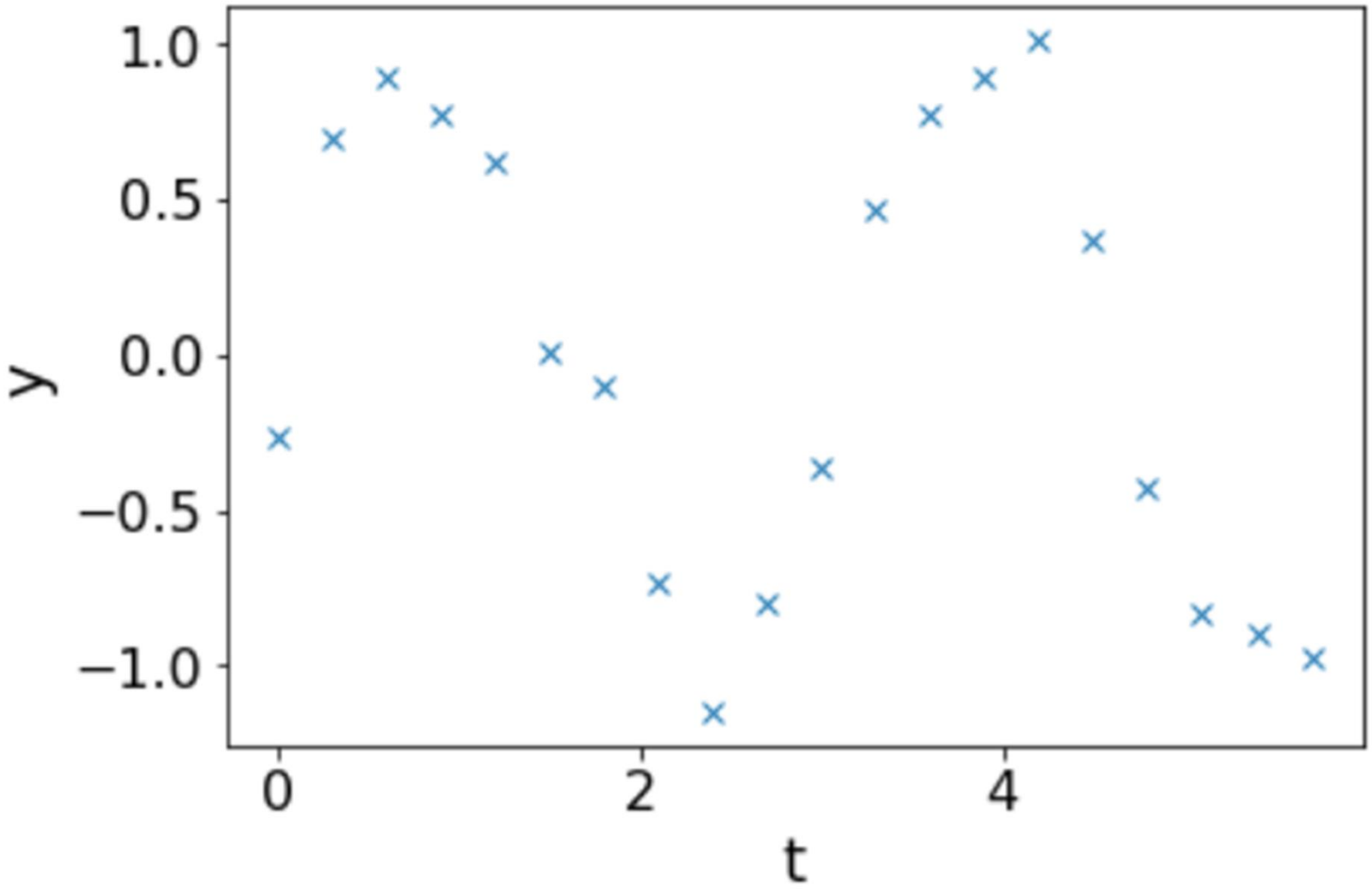
$\beta_1 = 3.03 \pm 0.05$

$\beta_2 = 1.9 \pm 0.5$

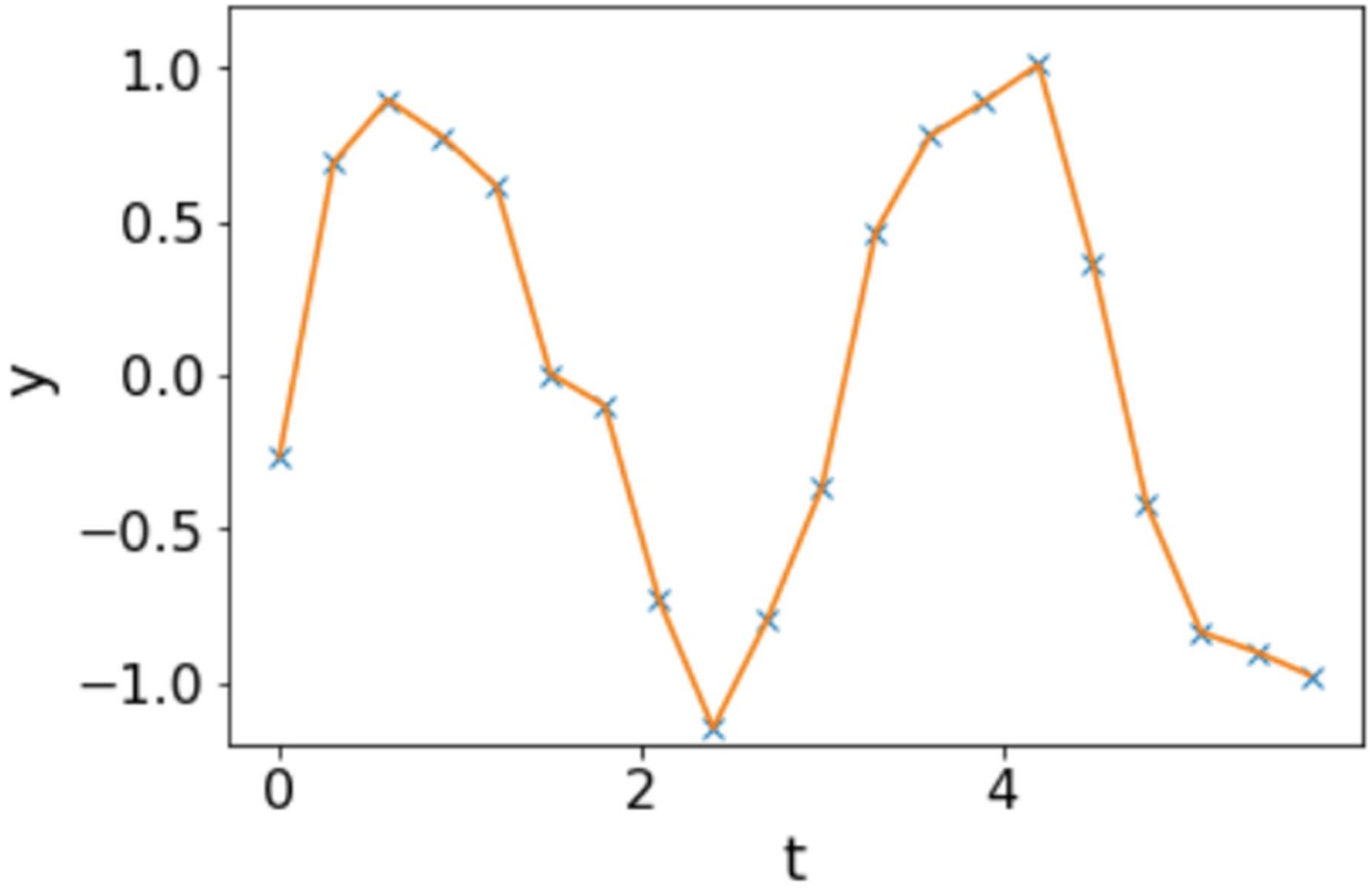
Put Error Bars on your data



Beware overfitting!

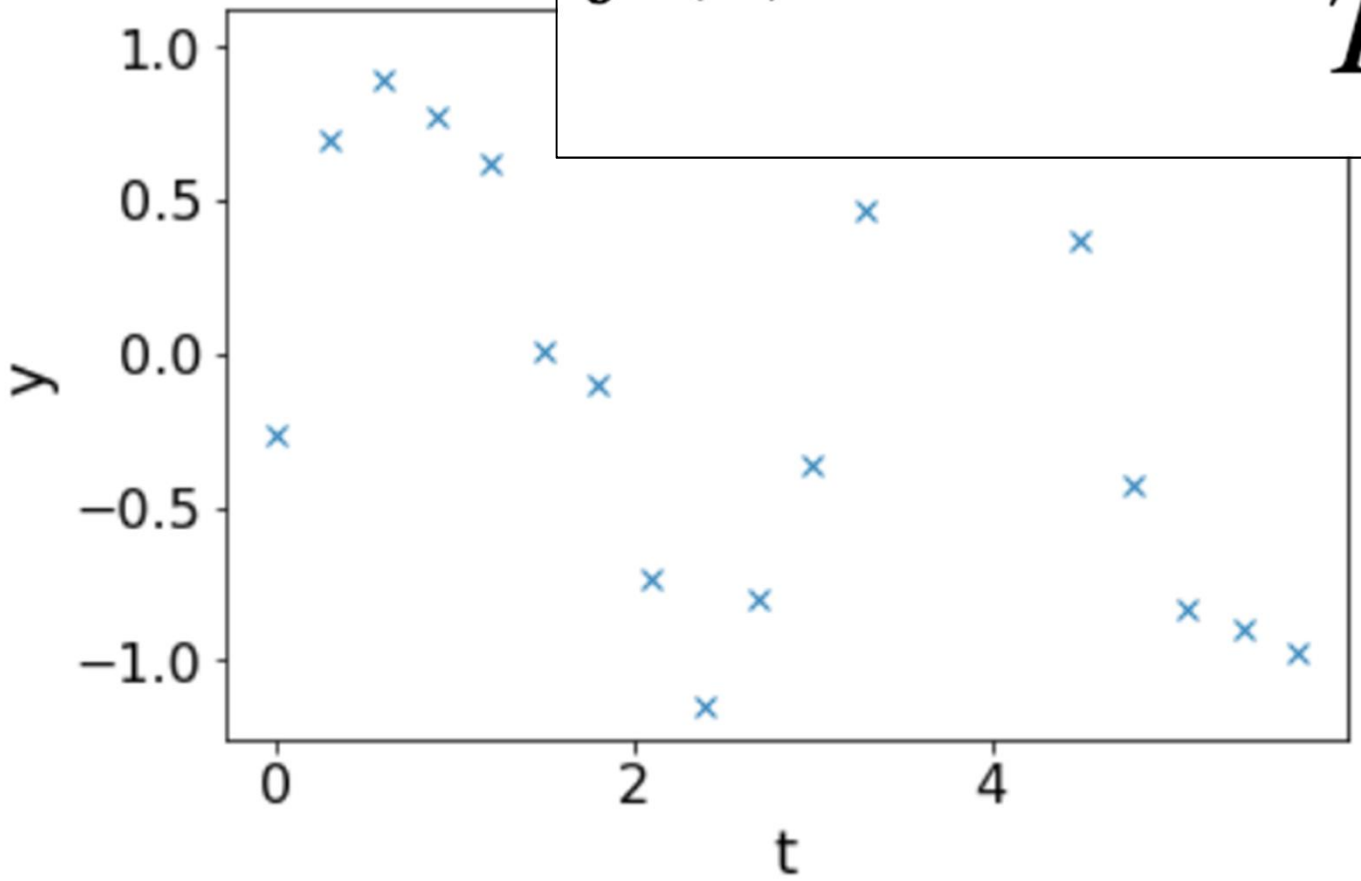


Beware overfitting – polynomial of order 25



Curve_fit: fitting non-linear functions (optional)

$$f(x) = A \sin \frac{2\pi}{T} t + \phi$$



Input: independent variable & fitting parameters

```
def my_sin(t, period, amplitude, phase):
    return amplitude*sp.sin(t * 2*sp.pi/period + phase)
```

Output: y-values of your function

Input: function name, independent variable, dependent variable, and first guess for fit parameters

```
fit = curve_fit(my_sin, t, y, p0=p0)
```

Output: optimized fit parameters

Imperial College London

Computing - Lecture 3

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Dr. Aidan Crilly

a.crilly16@imperial.ac.uk

Lecture 3

1. Review of Session 2 & The Spyder IDE
2. Control Flow of a Program – I
   - What is it?
   - Functions
3. Control Flow of a Program - II
   - Multiple Execution – “for loops”
   - Conditional Execution – “if statements”
   - Condition Controlled Multiple Execution – “while loops”

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Lecture 3 – Part 1

1. Review of Session 2 & The Spyder IDE
2. Control Flow of a Program – I
   - What is it?
   - Functions
3. Control Flow of a Program - II
   - Multiple Execution – “for loops”
   - Conditional Execution – “if statements”
   - Condition Controlled Multiple Execution – “while loops”

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Announcements

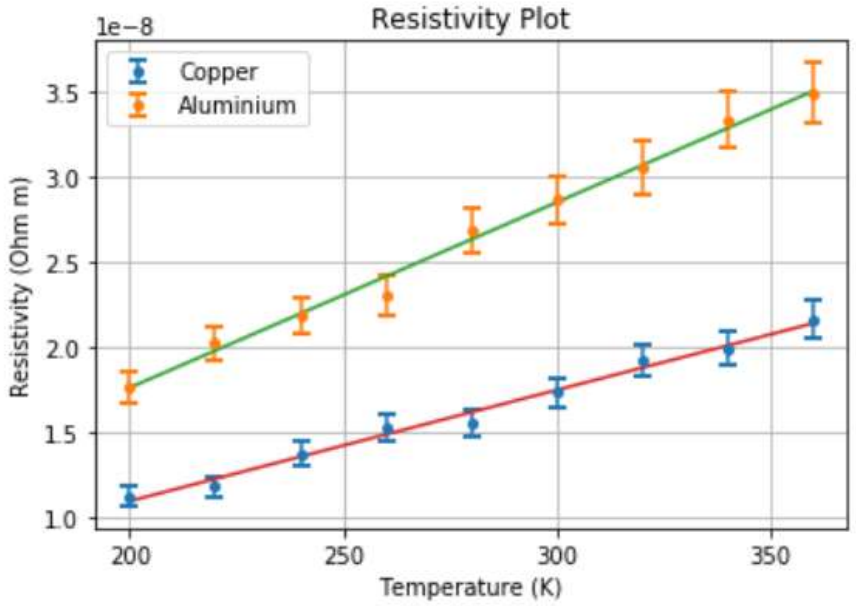
- Please complete Mentimeter poll after each Lab session.
- Example solutions for session 2 are on blackboard.
- Complete all green exercises before attempting ANY optional (yellow) exercises. This will involve initially skipping optional exercises.

This is an optional exercise

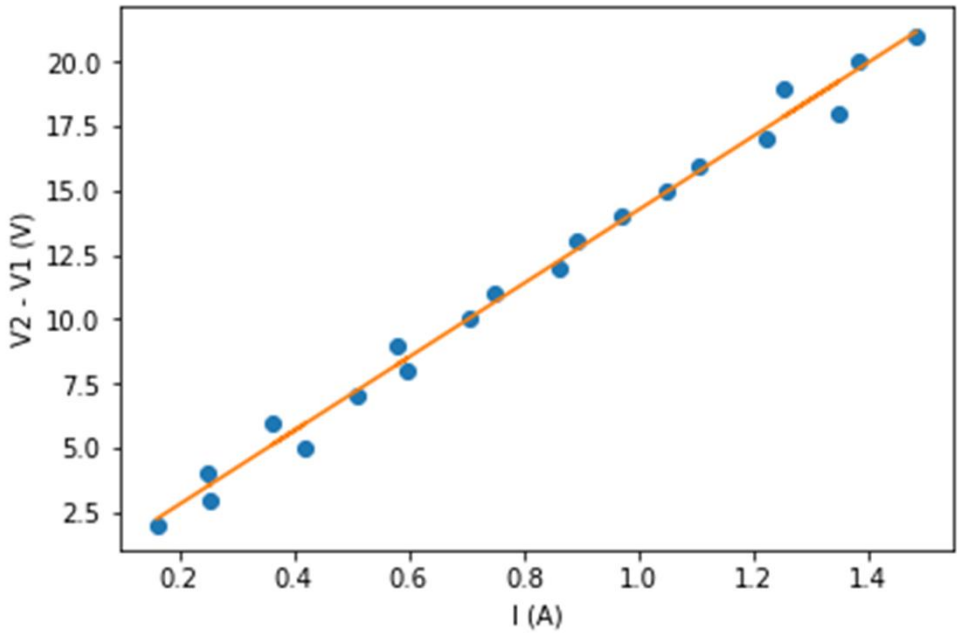
This is a core exercise

- If you have questions about plotting your experimental data using Python then ask a demonstrator.

Recap Session 2 – fitting your data




```
The resistance is 7.21 +/- 0.78 Ohm
```




```
Slope = 14.342 +/- 0.360
Intercept = -7.931e-02 +/- 3.241e-01
R = 7.17 +/- 0.18 Ohm
```

sp.polyfit(x,y,polynomial_order,covariance)

Fits a line for one variable and returns the best fit parameters (contained within Fit) and the uncertainty of those parameters (contained within cov).

sp.poly1d(Fit)

Returns an equation set by the parameters from Fit.

sp.optimize.curve_fit(fit_func,x,y,initial_guess,y_error)

Fits a user-defined function.

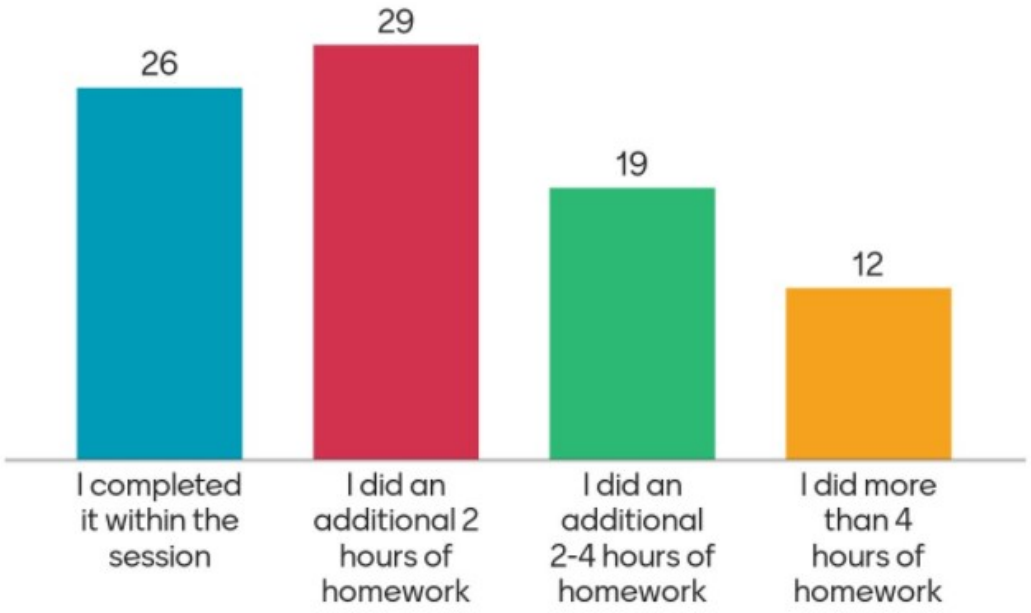
Session 2 Learning Objectives

- Plot a histogram of a 1D data set
- Plot error bars on scatter plots
- Fit a linear model to your data and plot both fit and data
- Find the uncertainties on your fit parameters

Your Feedback - Mentimeter Result

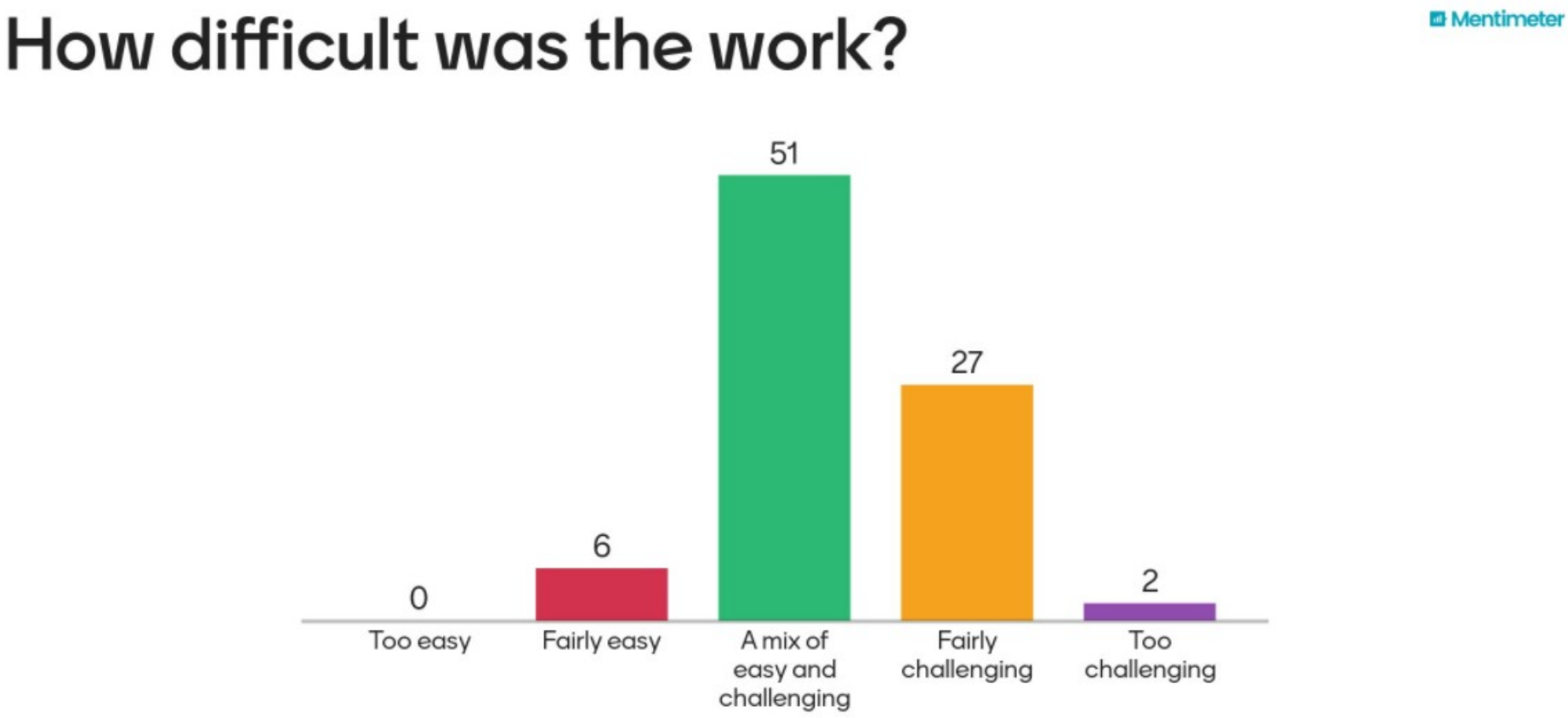
How much time did you spend on the Core Worksheet?

Mentimeter

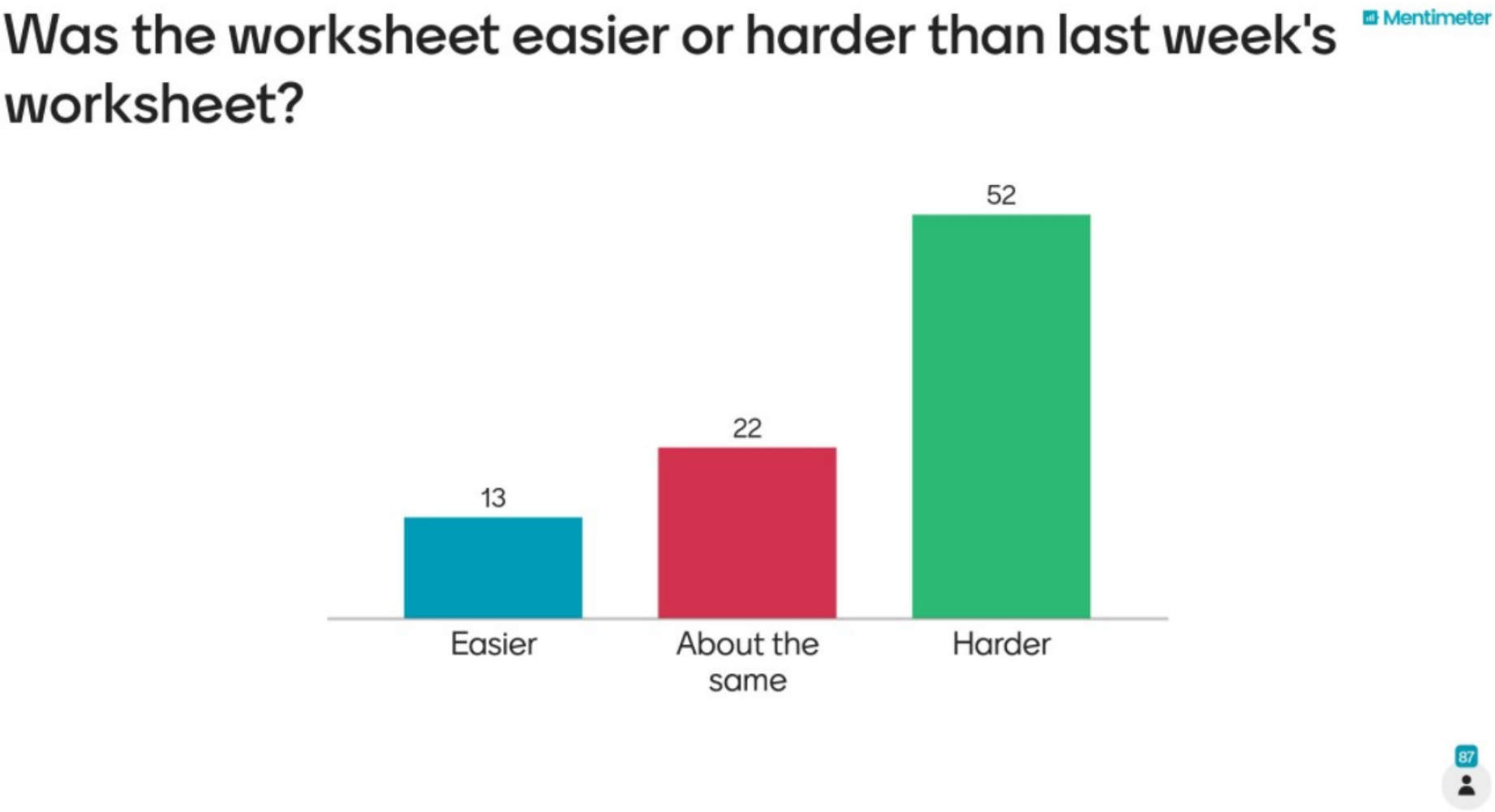


86

Your Feedback - Mentimeter Result



Your Feedback - Mentimeter Result



Year 1 Computing course: aims and structure

Aim: to be able to use Python to analyse your lab data and simulate physical scenarios.

- Session 1: Manipulating arrays and plotting data
- Session 2: Fitting data and error bars
- Session 3: Control Flow - Logic statements, loops, and functions
- Session 4: Numerical modelling
- Mini-Project: Using all your skills!

Year 1 Computing course: aims and structure

Aim: to be able to use Python to analyse your lab data and simulate physical scenarios.

- Session 1: Manipulating arrays and plotting data
- Session 2: Fitting data and error bars
- Session 3: Control Flow - Logic statements, loops, and functions
- Session 4: Numerical modelling
- Mini-Project: Using all your skills!

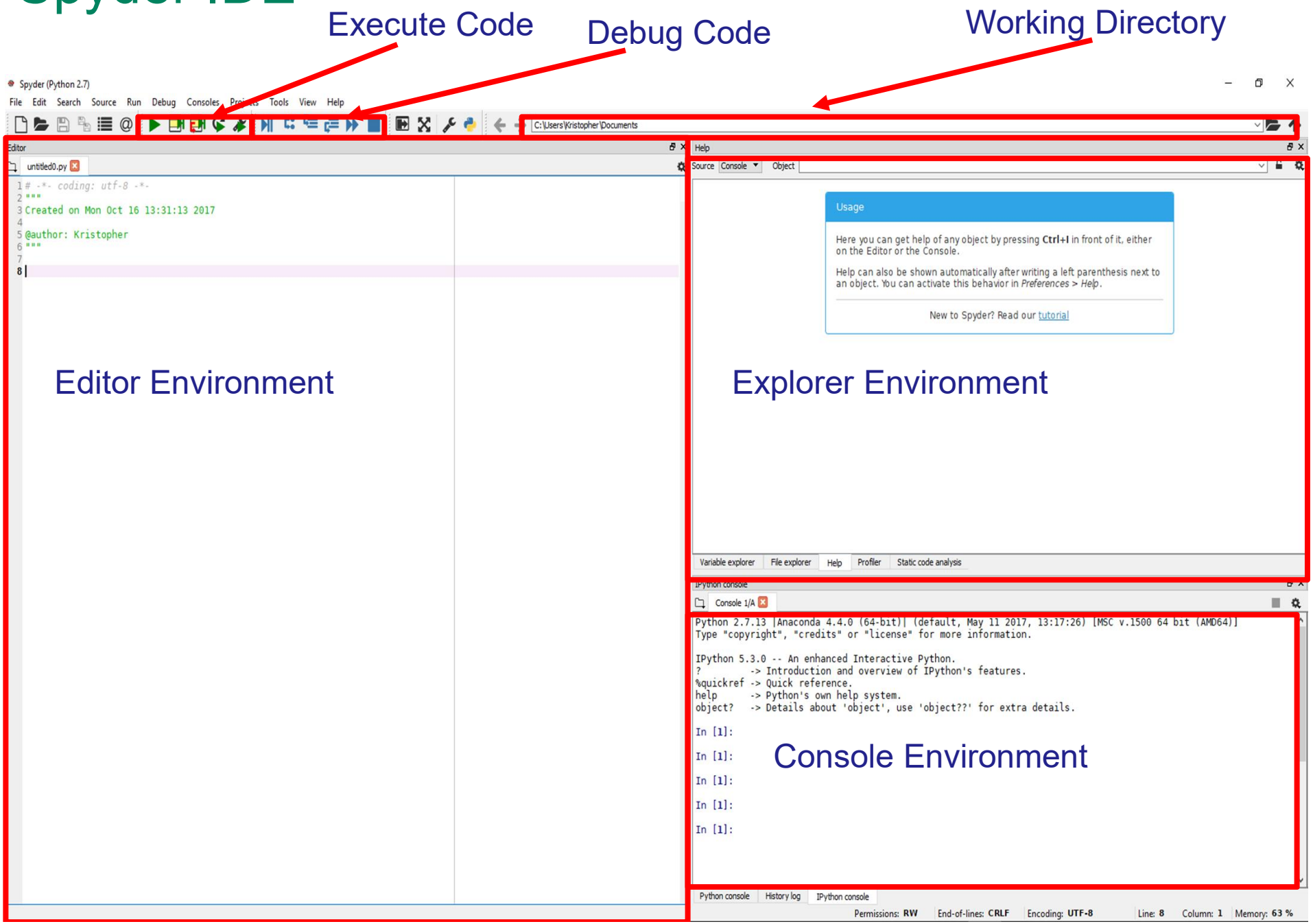
Spyder IDE

- We have used Jupyter Notebooks for the last two sessions to read, plot and fit data.
- Notebooks are great for small snippets of code.
- But they are not efficient for larger code blocks and write more complicated pieces of code.
- For this we can use an Integrated Development Environment (IDE).
- IDEs have many built-in tools for organising, testing & debugging code.
- For Python (in particular Anaconda) we use Spyder.

Accessing the Spyder IDE

- Here is how the Spyder Launch panel appears on Anaconda Navigator:

Spyder IDE


Execute Code
Debug Code
Working Directory
Spyder (Python 2.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Kristopher\Documents
Editor
untitled0.py
-*- coding: utf-8 -*-
"""
Created on Mon Oct 16 13:31:13 2017
@author: Kristopher
"""
Editor Environment
Help
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Variable explorer File explorer Help Profiler Static code analysis
Explorer Environment
IPython console
Console 1/A
Python 2.7.13 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:17:26) [MSC v.1500 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 5.3.0 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
In [1]:
Console Environment
Python console History log IPython console
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 8 Column: 1 Memory: 63 %

Getting started with Spyder

- Set the correct working directory by checking the upper right hand corner of your Spyder IDE.

- The Worksheets for Computing session 3 will still be written in Jupyter Notebooks but all your work should be done in Spyder.

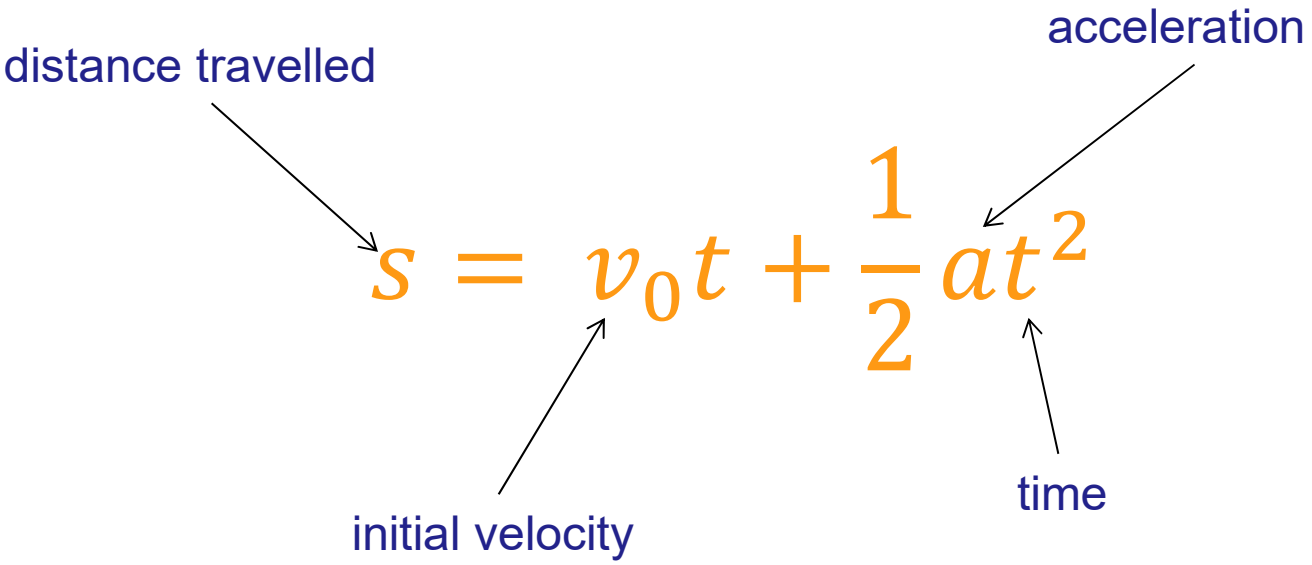
Lecture 3 – Part 2

1. Review of Session 2 & The Spyder IDE
2. Control Flow of a Program – I
   - What is it?
   - Functions
3. Control Flow of a Program - II
   - Multiple Execution – “for loops”
   - Conditional Execution – “if statements”
   - Condition Controlled Multiple Execution – “while loops”

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

An example – distance travelled by a car undergoing constant acceleration



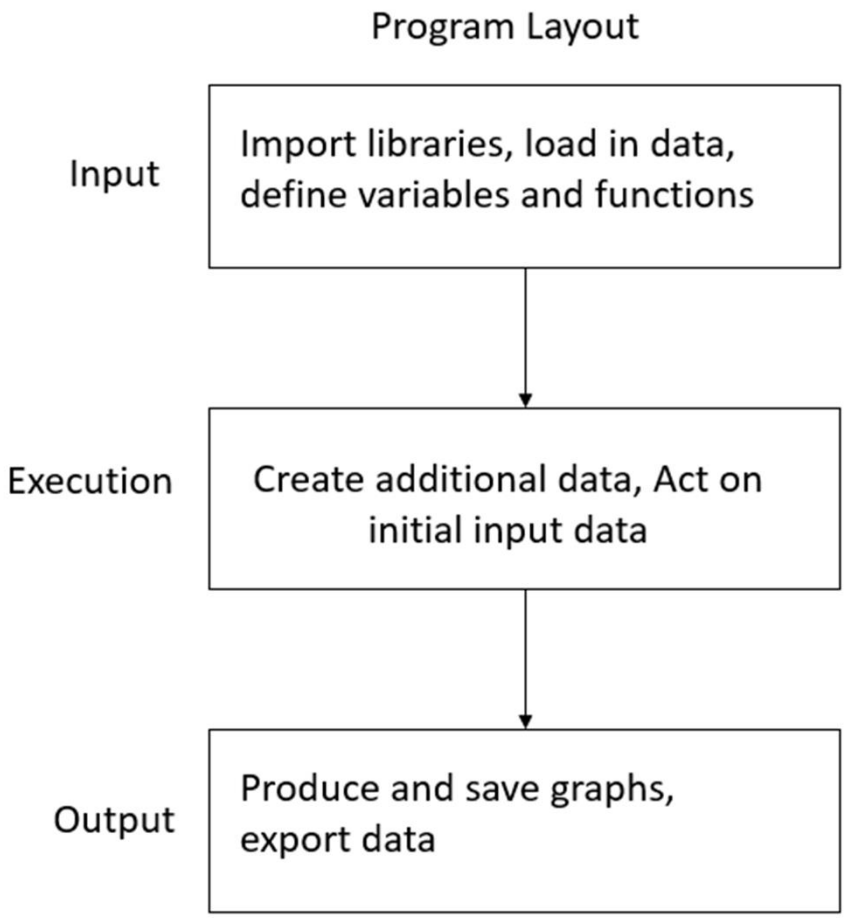
Plot distance travelled versus time for $v_0 = 10\ ms^{-1}$, $a = 0.8\ ms^{-2}$ and $t = 0 - 60\ s$

Control Flow

- Python executes statements one at time, from top to bottom of the program
- But, what if:
  - We want to execute the same statement multiple times?
  - We only want to execute a statement under certain conditions?

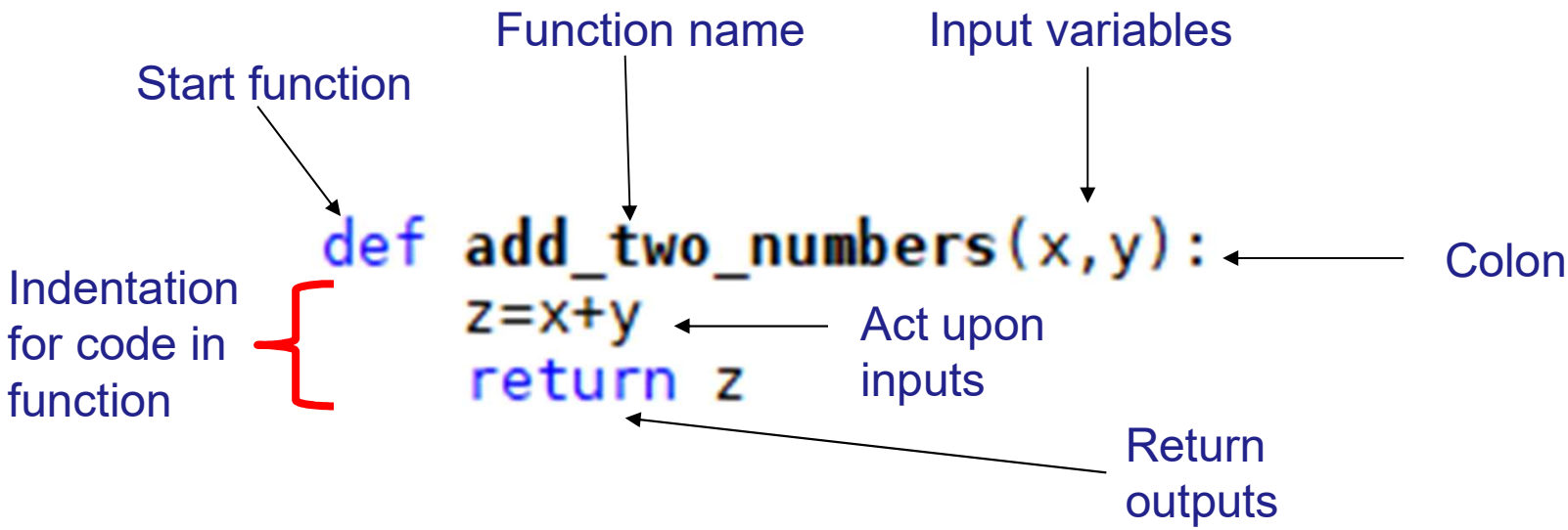
Control Flow is the order in which our lines of code are executed.

Use Block Diagrams to show Flow of Program



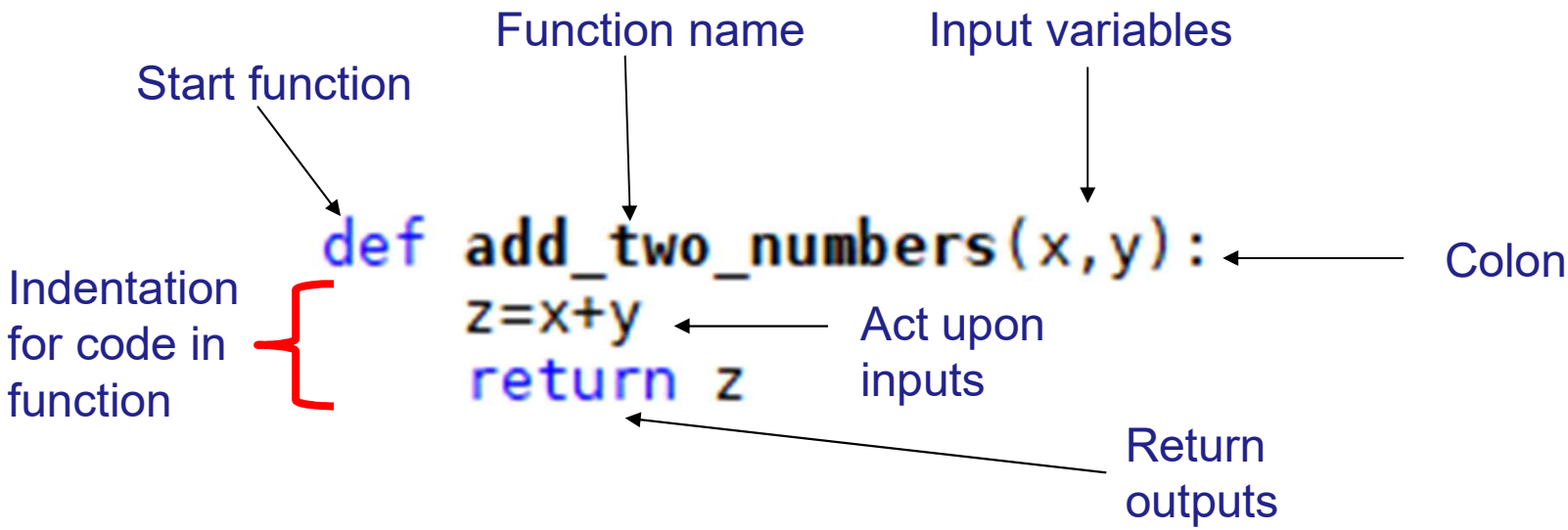
Functions

- In this session you will be expected to create your own custom functions to use
- You briefly saw a custom function in the optional material of session 2
- Here is a function which adds together two numbers



Functions

- Reduce duplicate code
- Decompose complex code into simpler steps
- Make it easier to reuse the code in other programs



Lecture 3 Part 3

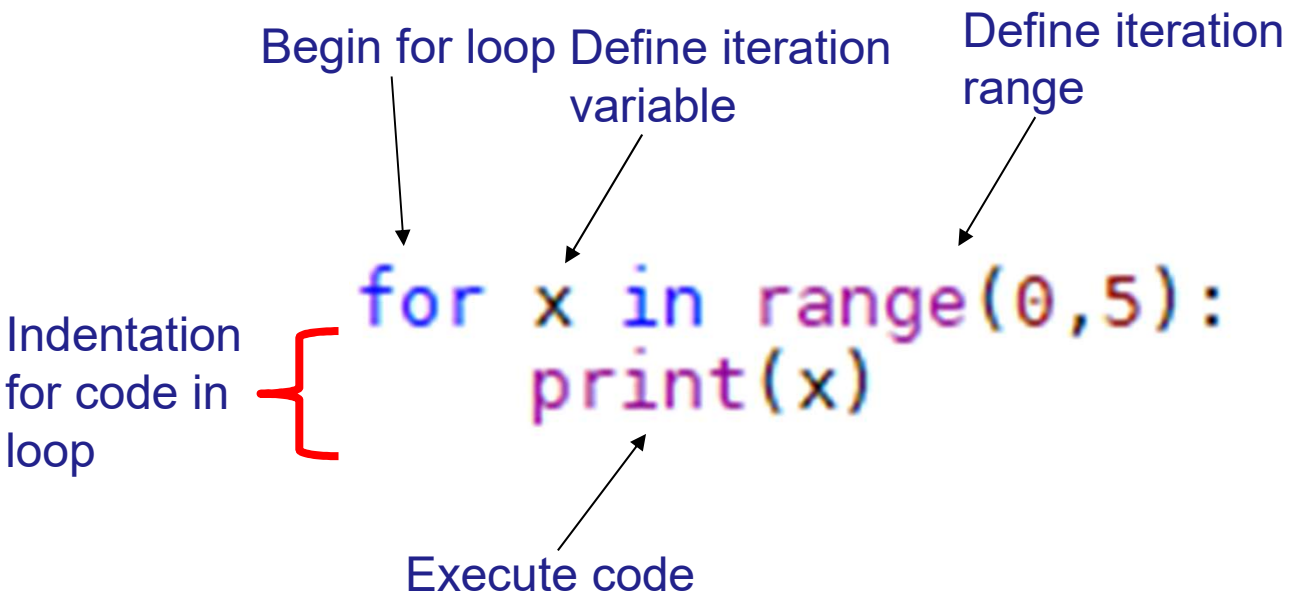
1. Review of Session 2 & The Spyder IDE
2. Control Flow of a Program – I
   - What is it?
   - Functions
3. Control Flow of a Program - II
   - Multiple Execution – “for loops”
   - Conditional Execution – “if statements”
   - Condition Controlled Multiple Execution – “while loops”

Dr. Brian Appelbe

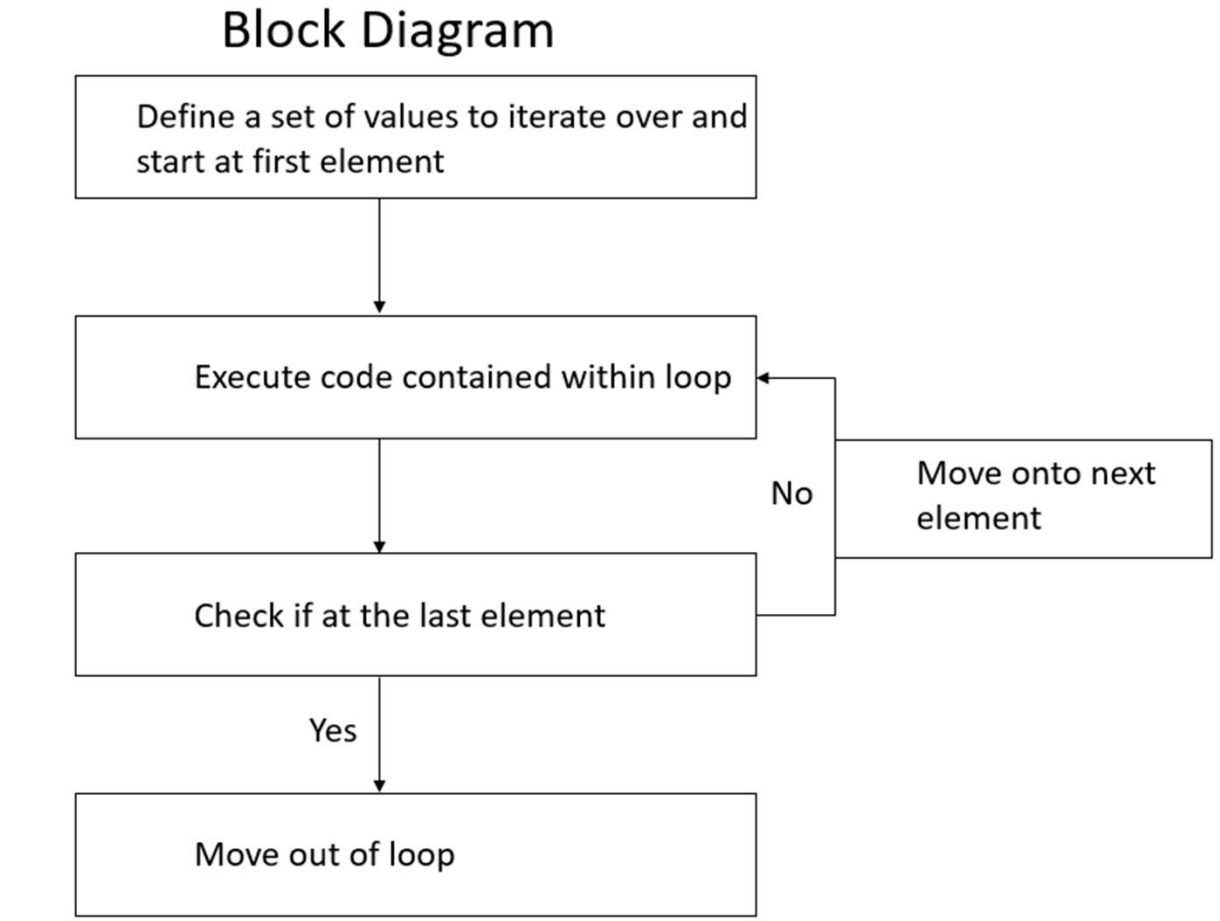
b.appelbe07@imperial.ac.uk

For Loops

- The most basic flow manipulator, for loops allow lines of code to be executed multiple times



For Loops - Diagram of Execution



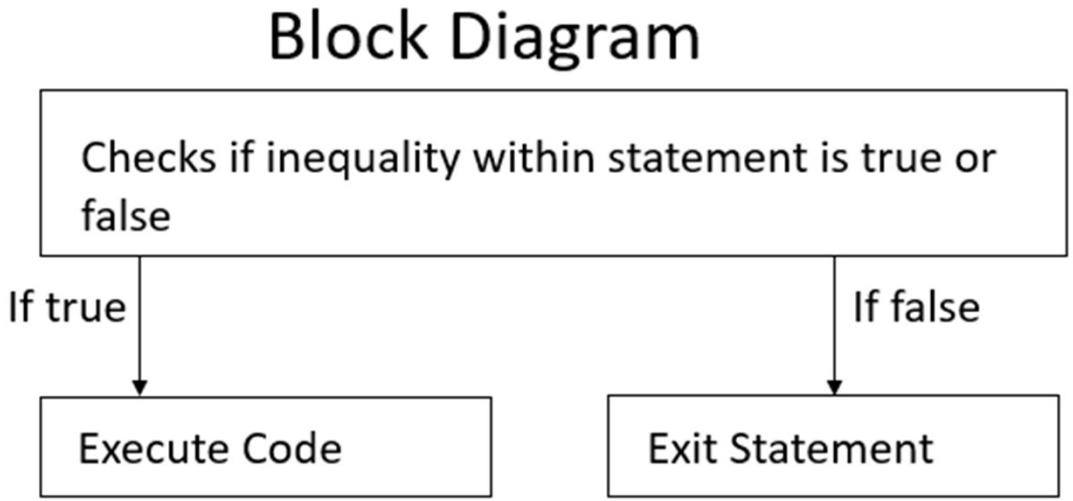

```
for x in range(0,5):
    print(x)
```

Mathematical Inequalities

- When we are controlling the workflow of programs, we often need to compare value of variables.
- This is achieved by using inequality statements.
- They are:
  - == Equal to
  - != Not Equal to
  - > Greater than
  - < Less then
  - >= Greater than or Equal to
  - <= Less than or Equal to

If Statements

- If statements are used to set a condition on a piece of code to only execute if the statement returns True.



If Statements - If, else and elif

- If statements do not need to be a binary yes/no statement. They can be extended by other associated keywords
- The else statements allows code to be executed when the associated if statements returns False.
- The elif (else-if) statements allows other statements to be compared within the same code block.

If Statement Example

- If statement:

```
Input_var=3

if(Input_var > 3):
    print("Your number is greater than 3")
```

- If else statement:

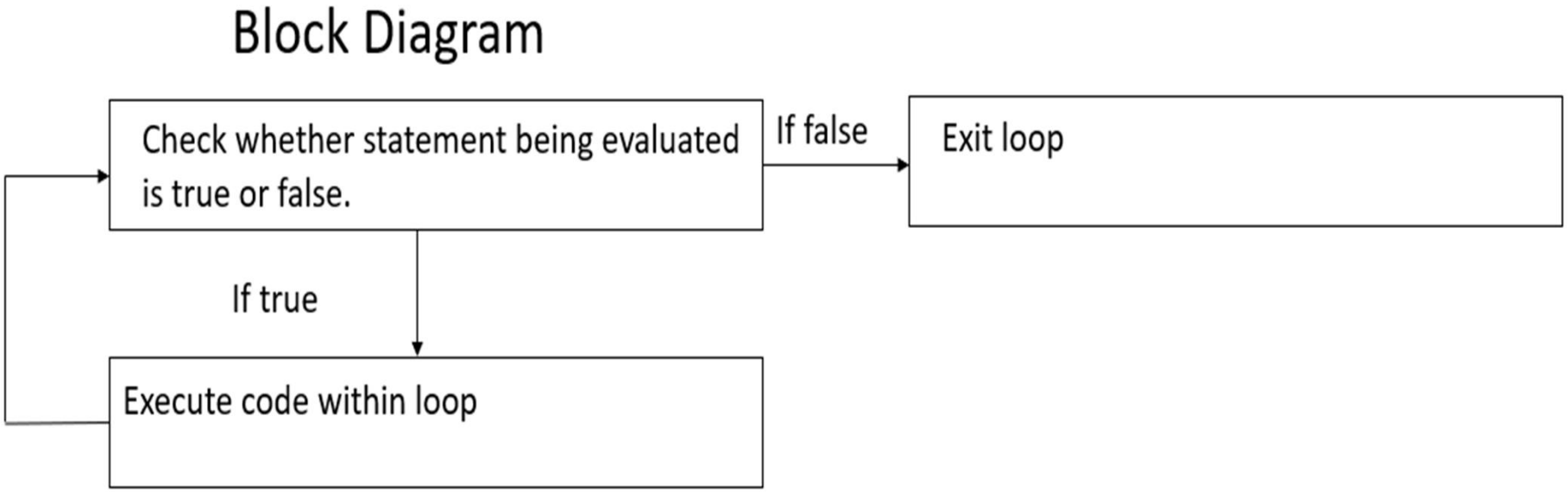
```
if(Input_var > 3):
    print("Your number is greater than 3")
else:
    print("Your number is less than 3")
```

- If elif else statement:

```
if(Input_var > 3):
    print("Your number is greater than 3")
elif(Input_var < 3):
    print("Your number is less than 3")
else:
    print("Your number is 3!")
```

While Statement

- While statements are used to set a condition on a piece of code to continue executing that code until it returns False.



While Statement

- A while statement will continue to execute until it encounters false when evaluating its inequality.
- If you do not give a way to do this, you will end up in an infinite loop with no way to exit.
- Always check if you have given your statement a way to finish its execution!

Final Notes about Session

- This session will introduce concepts and ideas that you will not have encountered before without prior programming experience.
- As such this will be a challenging session for most of you.
- If you have problems with how to tackle a problem consult the bottom of the Session 1 notebook on how to approach situations like this.

Imperial College London

Computing - Lecture 4

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Dr. Aidan Crilly

a.crilly16@imperial.ac.uk

Lecture 4

1. Review of Session 3
2. Planning your code & Debugging using Spyder
3. Session 4: Numerical Methods

Dr. Brian Appelbe b.appelbe07@imperial.ac.uk

Lecture 4 – Part 1

1. Review of Session 3
2. Planning your code & Debugging using Spyder
3. Session 4: Numerical Methods

Dr. Brian Appelbe b.appelbe07@imperial.ac.uk

Announcements

- Example solutions for Session 3 are now on blackboard.
- Session 4 is the final regular Computing session.
- Project:
  - Begins on Mon 8th Nov.
  - Next lecture will describe it in detail.
  - Submission deadline is Fri 26th Nov.
  - Submission is mandatory (10% of Prac Phys module).
  - No scheduled sessions but there will be drop-in sessions if you have questions.

Announcements

- A set of Physics problems to solve using Python has been made just for Imperial Physics students!
- This is extra-curricular, but is interesting and could help improve your Python skills and Physics knowledge.
- Find it here: pyproblems.github.io/book
- Introductory video: youtu.be/0sWDM7R_5XA

These links are at the bottom of Core Worksheet 4

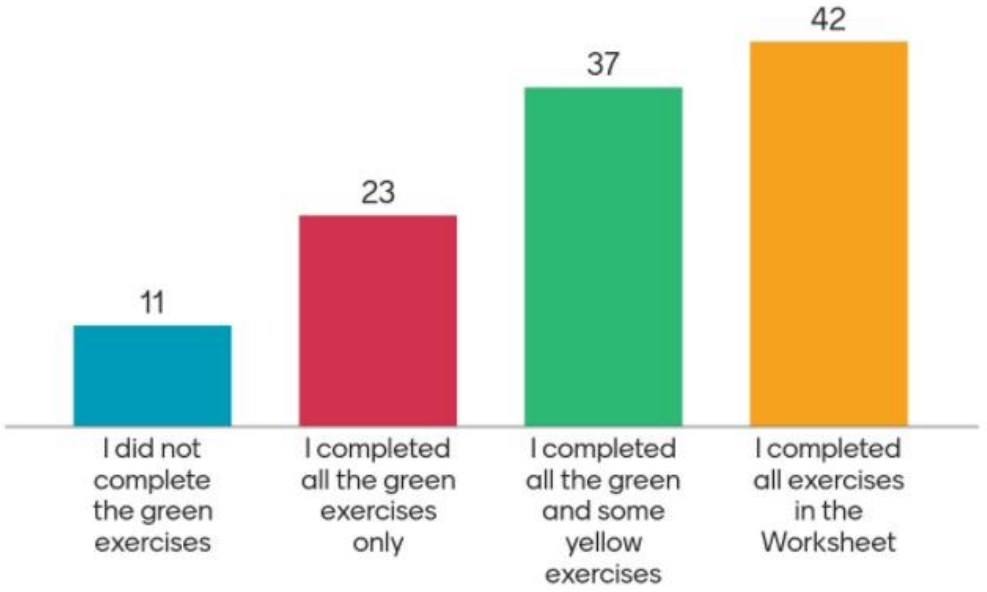
Session 3 Learning Objectives

- Using the Spyder IDE
- Writing functions
- Using for, if and while statements to control program flow
- Writing longer programs to achieve a specific task

Mentimeter Session 3 Result

Mentimeter

How much of the Core Worksheet did you complete?

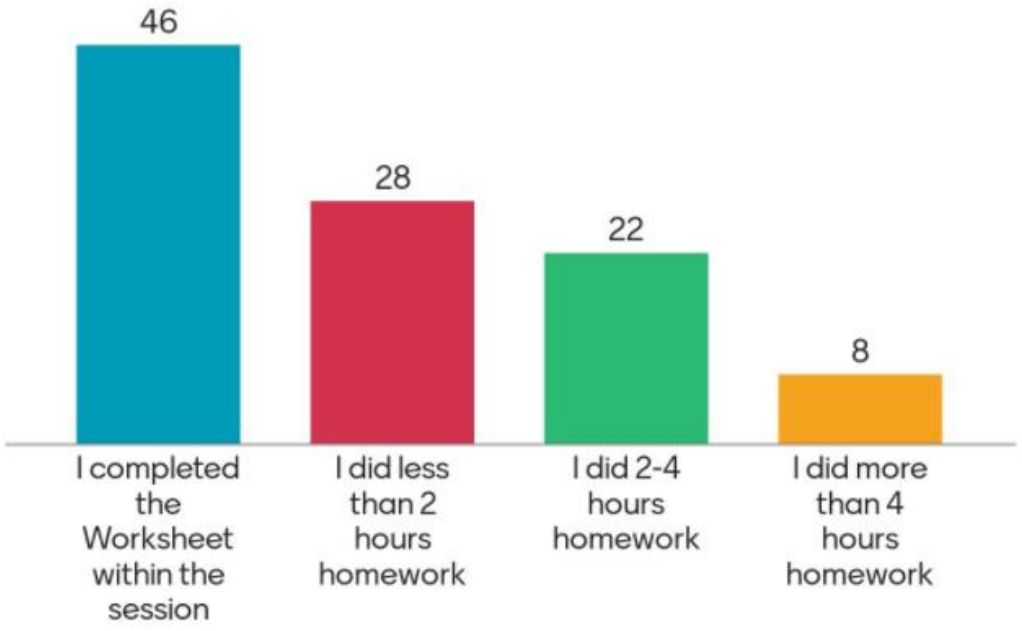


113

Mentimeter Session 3 Result

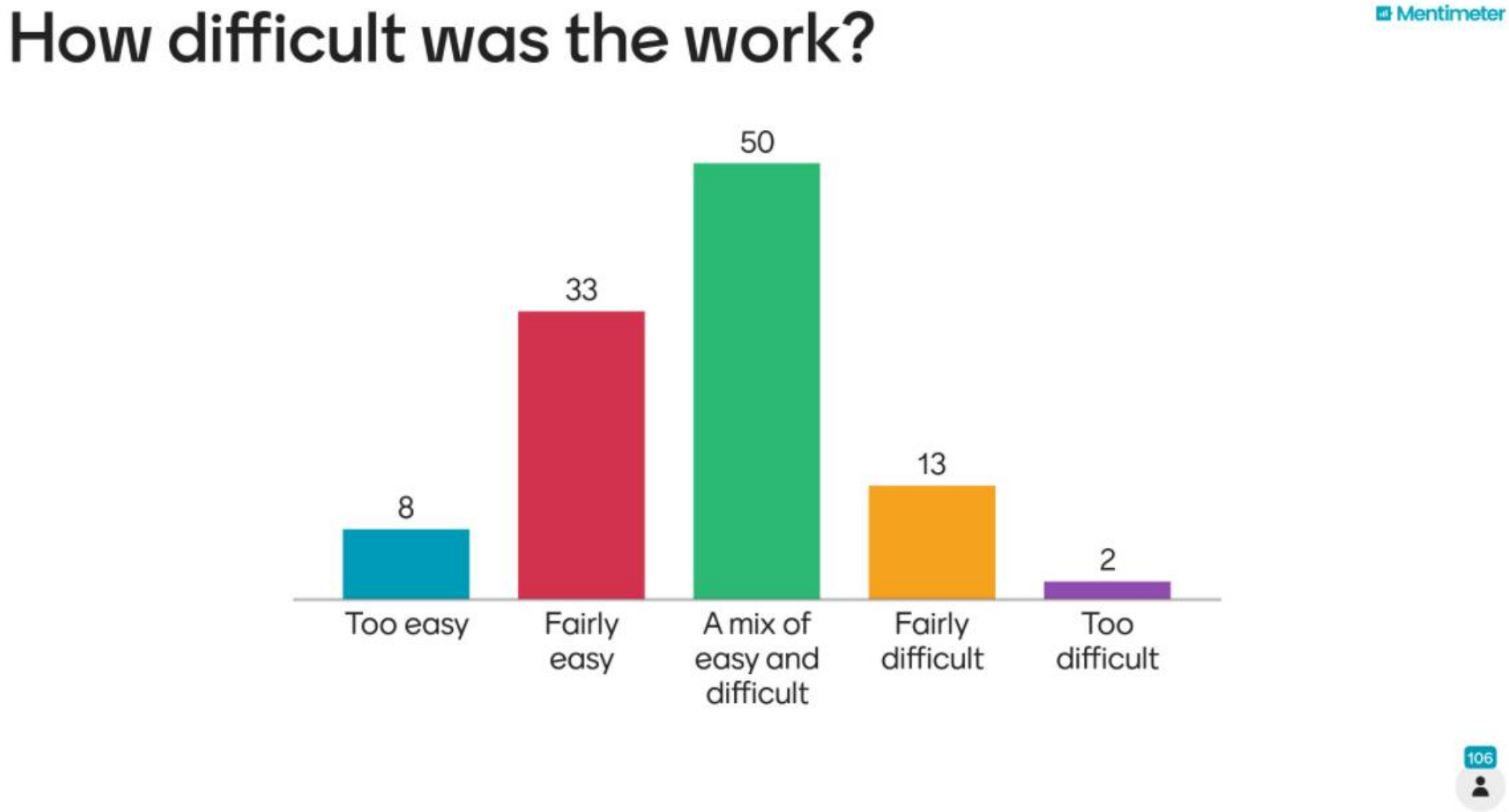
How much time did you spend on the Worksheet?

Mentimeter



104

Mentimeter Session 3 Result

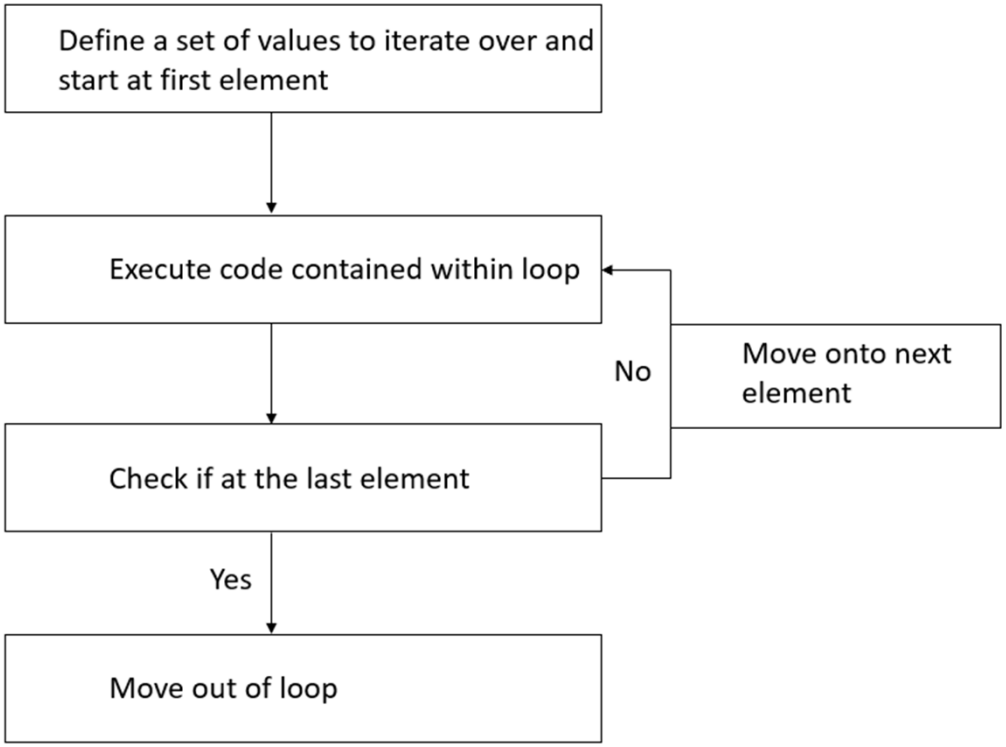


General Feedback

- Many of you with previous programming experience found this session easier than session 2 ï the concepts were familiar.
- If this is your first time coding then please ensure that you understand the ideas about control flow, functions, for loops, if statements, etc. These are fundamental concepts in programming.

General Feedback

- Carefully read the Worksheet & think about what you are doing.
- Many students found a big jump between green and optional exercises ï optional exercises required you to plan your code!
- Did you make a flow chart or write pseudo-code before writing your code?



Year 1 Computing course: aims and structure

Aim: to be able to use Python to analyse your lab data and simulate physical scenarios.

- Session 1: Manipulating arrays and plotting data
- Session 2: Fitting data and error bars
- Session 3: Control Flow - Logic statements, loops, and functions
- Session 4: Numerical modelling
- Project: Using all your skills!

Year 1 Computing course: aims and structure

Aim: to be able to use Python to analyse your lab data and simulate physical scenarios.

- Session 1: Manipulating arrays and plotting data
- Session 2: Fitting data and error bars
- Session 3: Control Flow - Logic statements, loops, and functions
- Session 4: Numerical modelling
- Project: Using all your skills!

Lecture 4 – Part 2

1. Review of Session 3
2. Planning your code & Debugging using Spyder
3. Session 4: Numerical Methods

Dr. Brian Appelbe b.appelbe07@imperial.ac.uk

How to write code to solve a Physics problem:

1. Carefully formulate the question
2. Work out the Maths / Physics (in your notebook!)
3. Organise your workings into sequential steps (e.g. build a flow diagram)
4. Translate your step-by-step workings into code

Recap Session 3 ï the Fibonacci Series

$$f(n) = f(n-1) + f(n-2)$$

Exercise: create a function that generates the first *n* fibonacci numbers.

1. Formulate the question

Create a function that calculates fibonacci numbers.

Input: *n*
Output: fibonacci series

E.g. fibonacci series up to *n* = *10:*

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

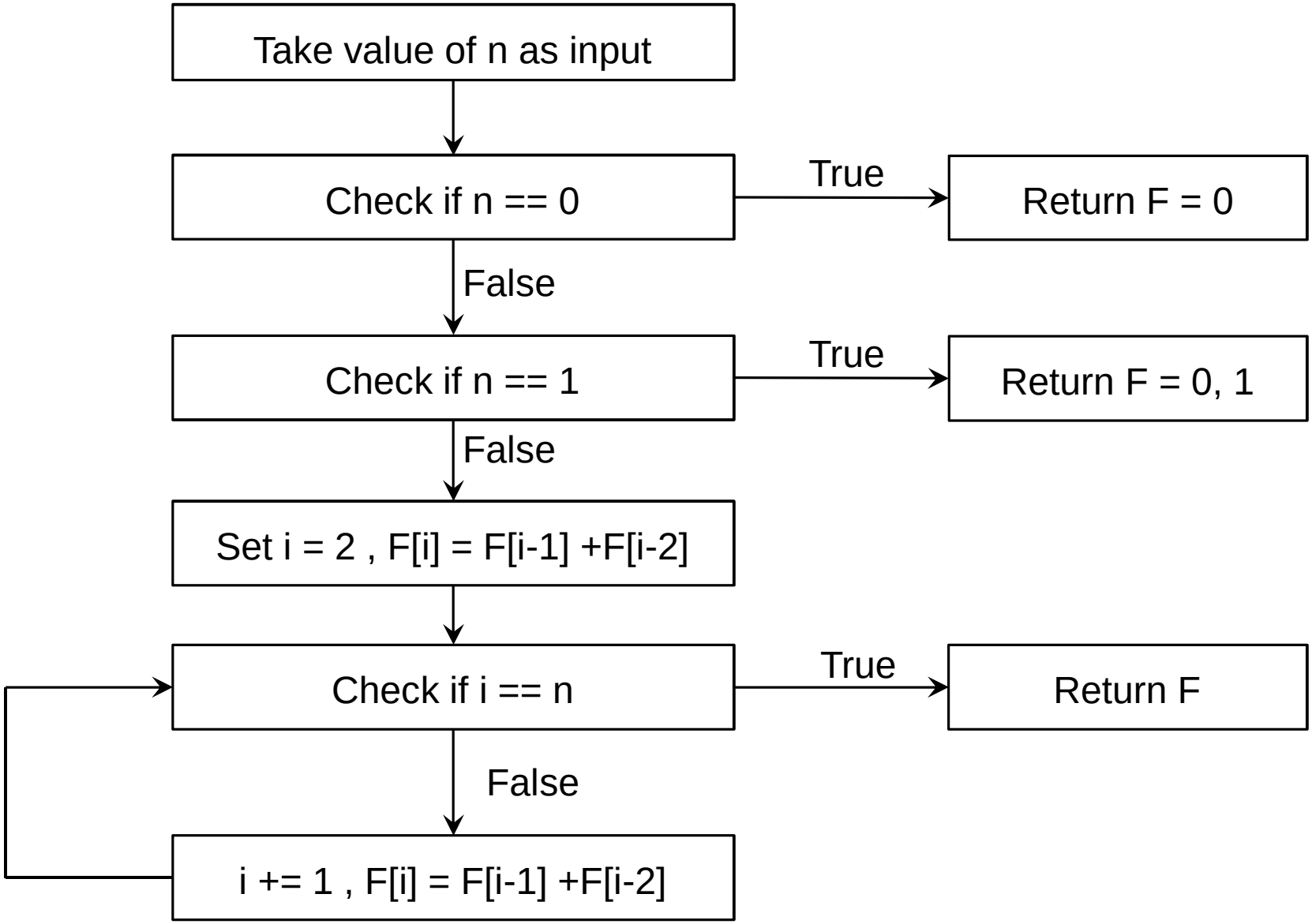
2. Work out the Maths/Physics of the problem

( ) ( ) ( )

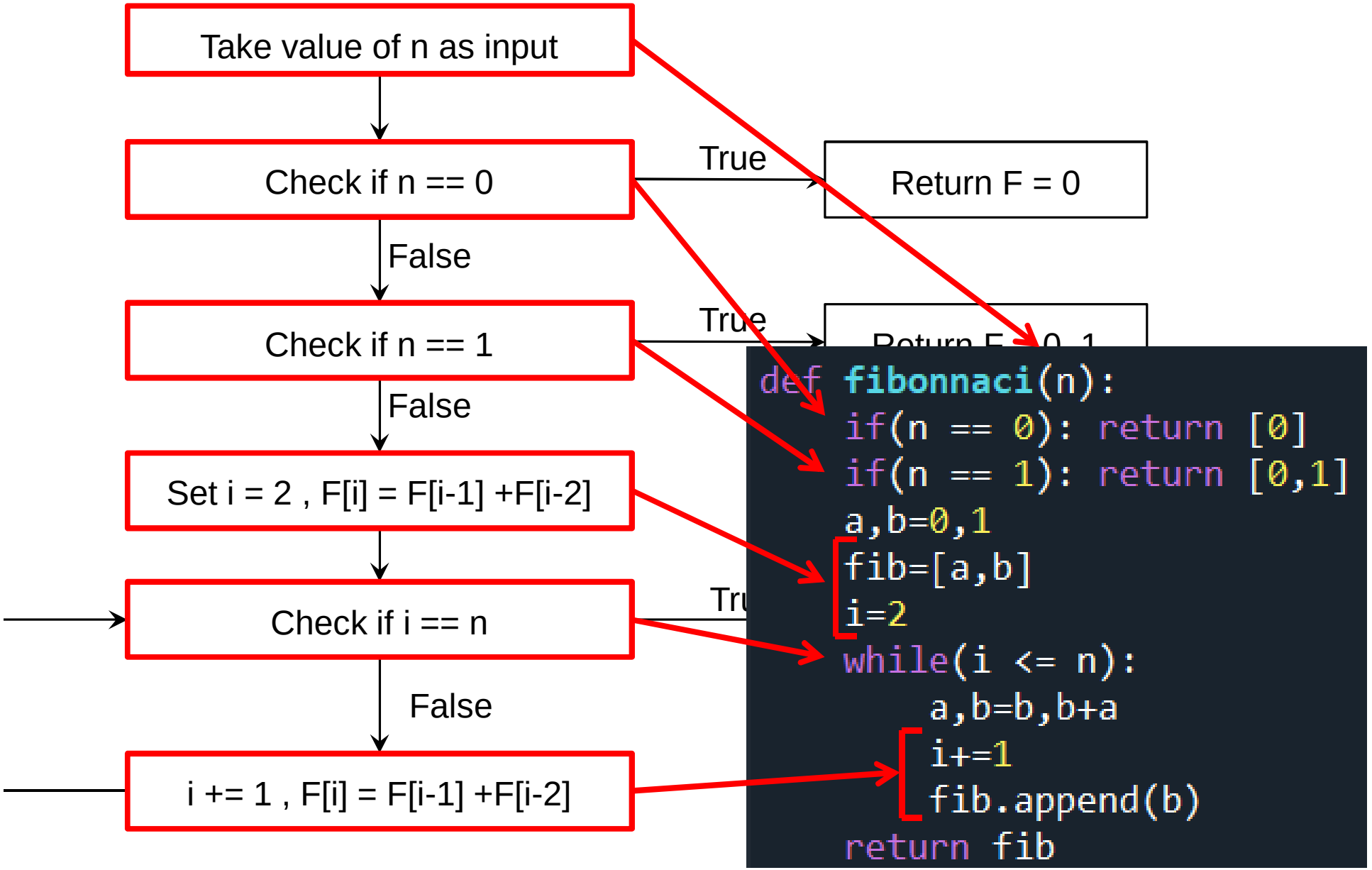
for integer *with*

( )
( )

3. Work out steps - Build a flow diagram



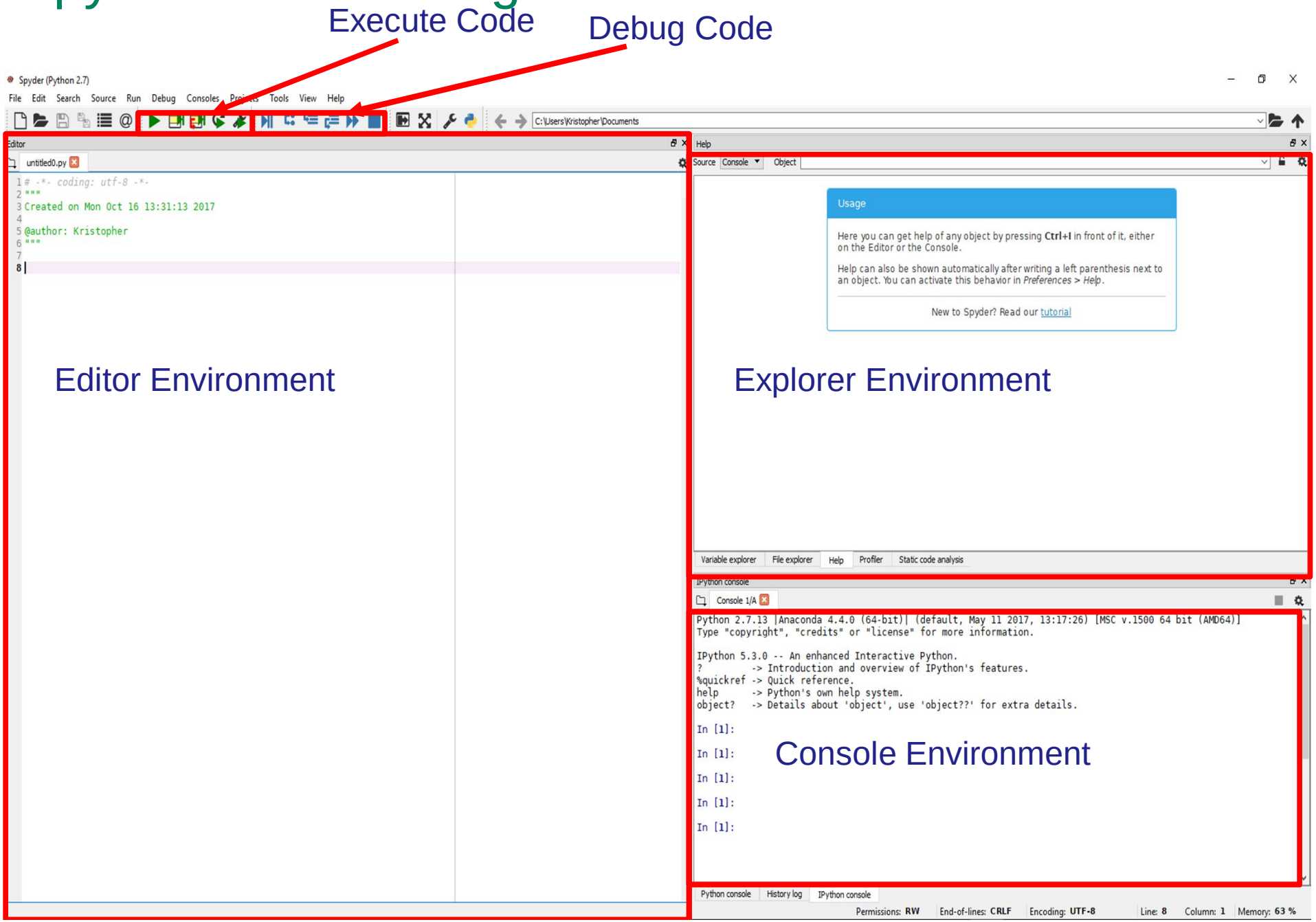
4. Translate your step-by-step workings into code



Spyder IDE ï Debug Mode

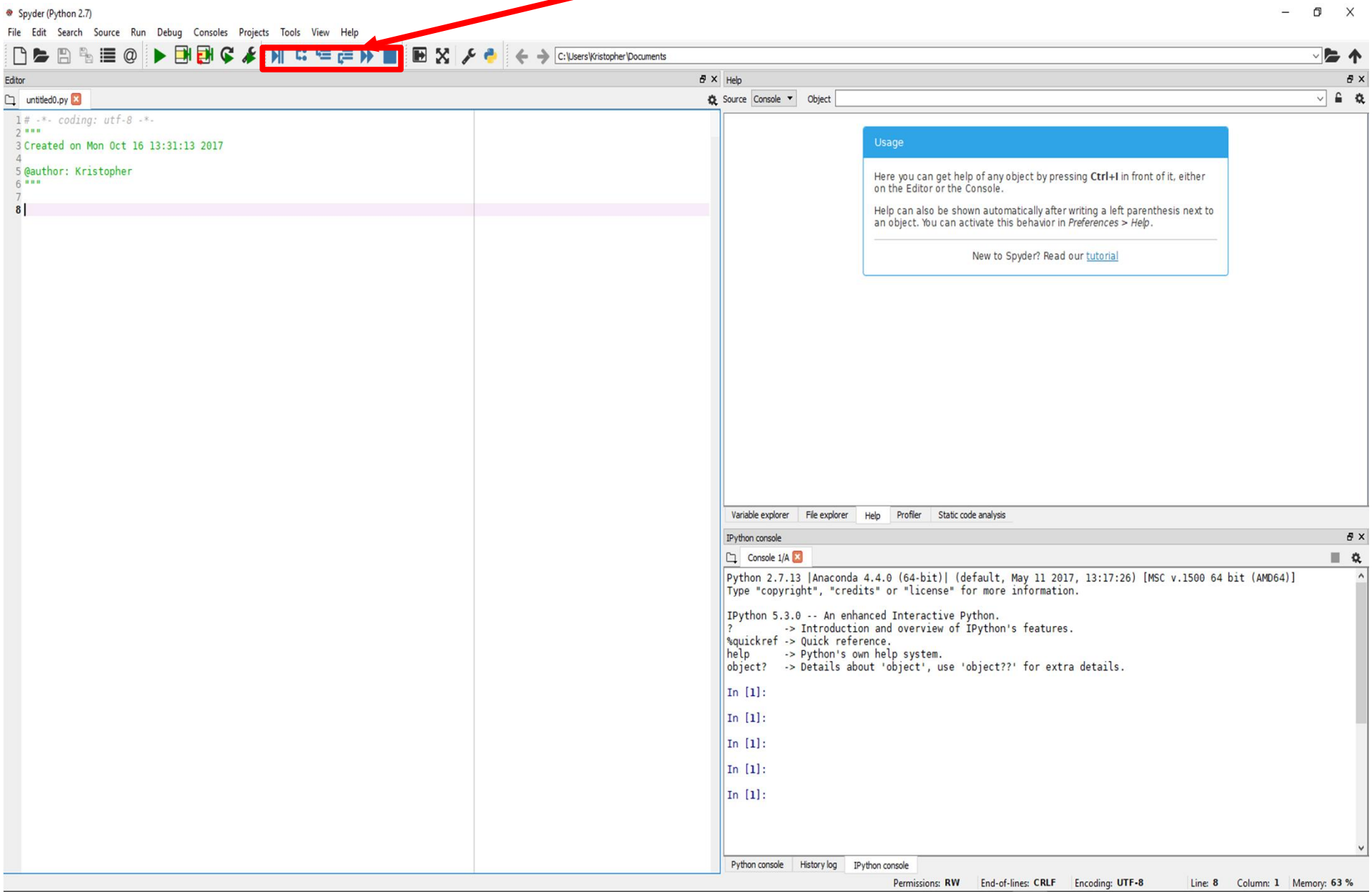
- Debugging is the process of finding & correcting errors (ñbugsò) in your code.
- There are many techniques and strategies for debugging your code.
- Debug mode in Spyder allows you to pause the execution of your code so that you can inspect intermediate results, check the control flow of the program, etc.

Spyder IDE ï Debug Mode

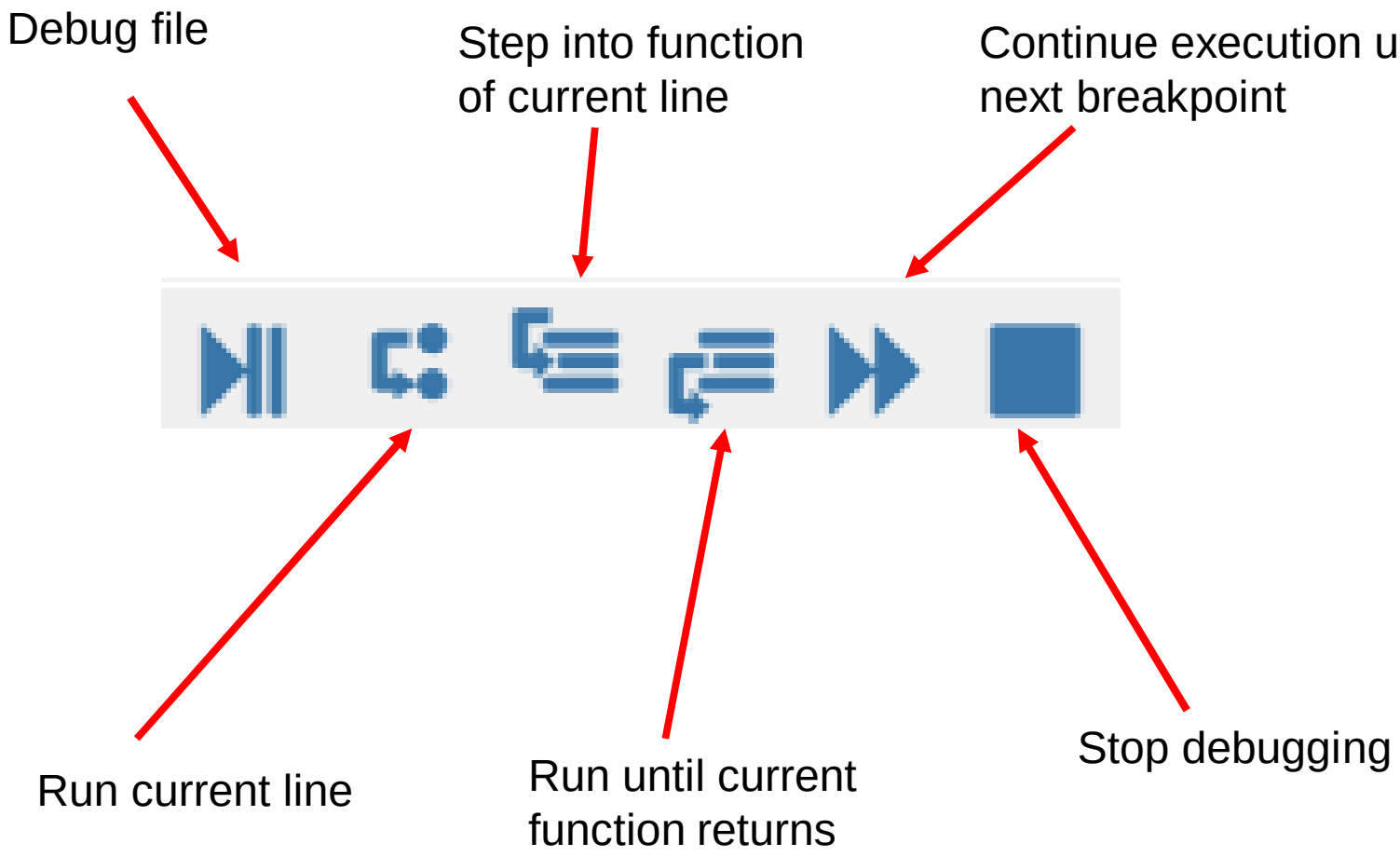


Spyder IDE ï Debug Mode

Debug Code



Spyder IDE ï Debug Mode



Lecture 4 – Part 3

1. Review of Session 3
2. Planning your code & Debugging using Spyder
3. Session 4: Numerical Methods

Dr. Brian Appelbe b.appelbe07@imperial.ac.uk

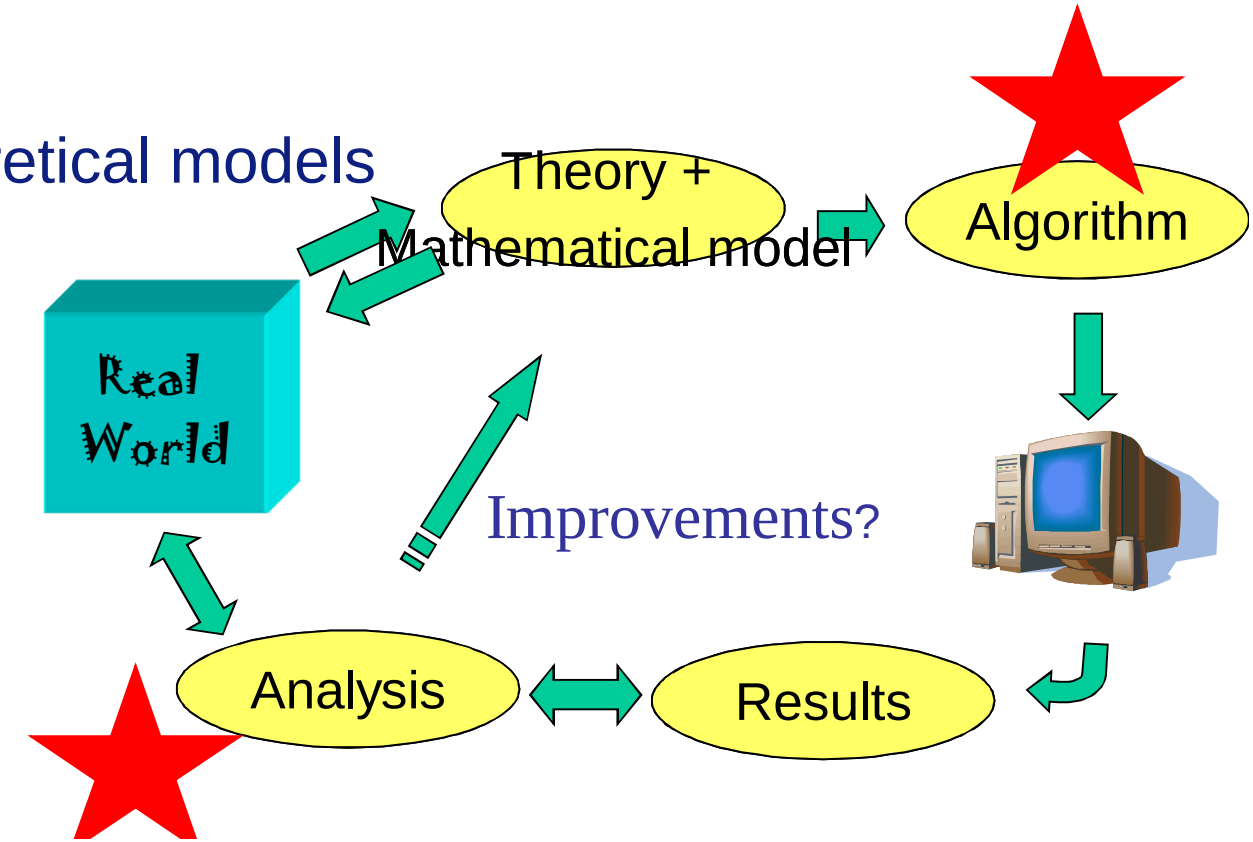
Why *scientific* computing?

Complex calculations

- Numerical computation of equations
- Multiphysics computation

Repeated calculations

- Data analysis
- Simulations of theoretical models



Numerical Methods

Numerical methods use iterative algorithms to find solutions to mathematical or physical models.

This weekôs session focuses on:

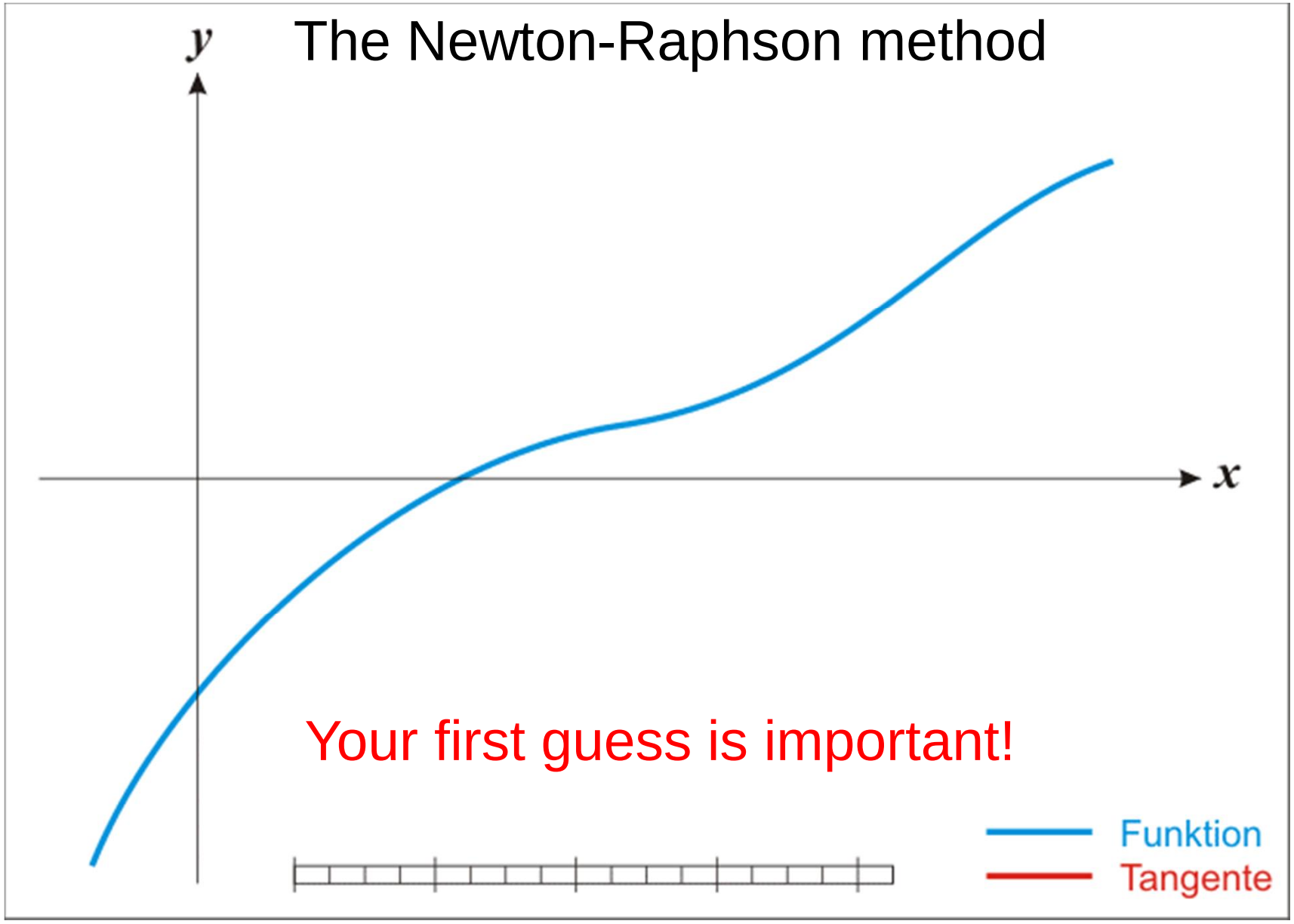
- Solving equations
- Optimisation (i.e. minimisation)
- Differentiation and Integration
- Ordinary Differential Equation solving (Advanced Worksheet)

Solving equations

1. Polynomials: $3x^3 + x^2$ ï $x = 10$

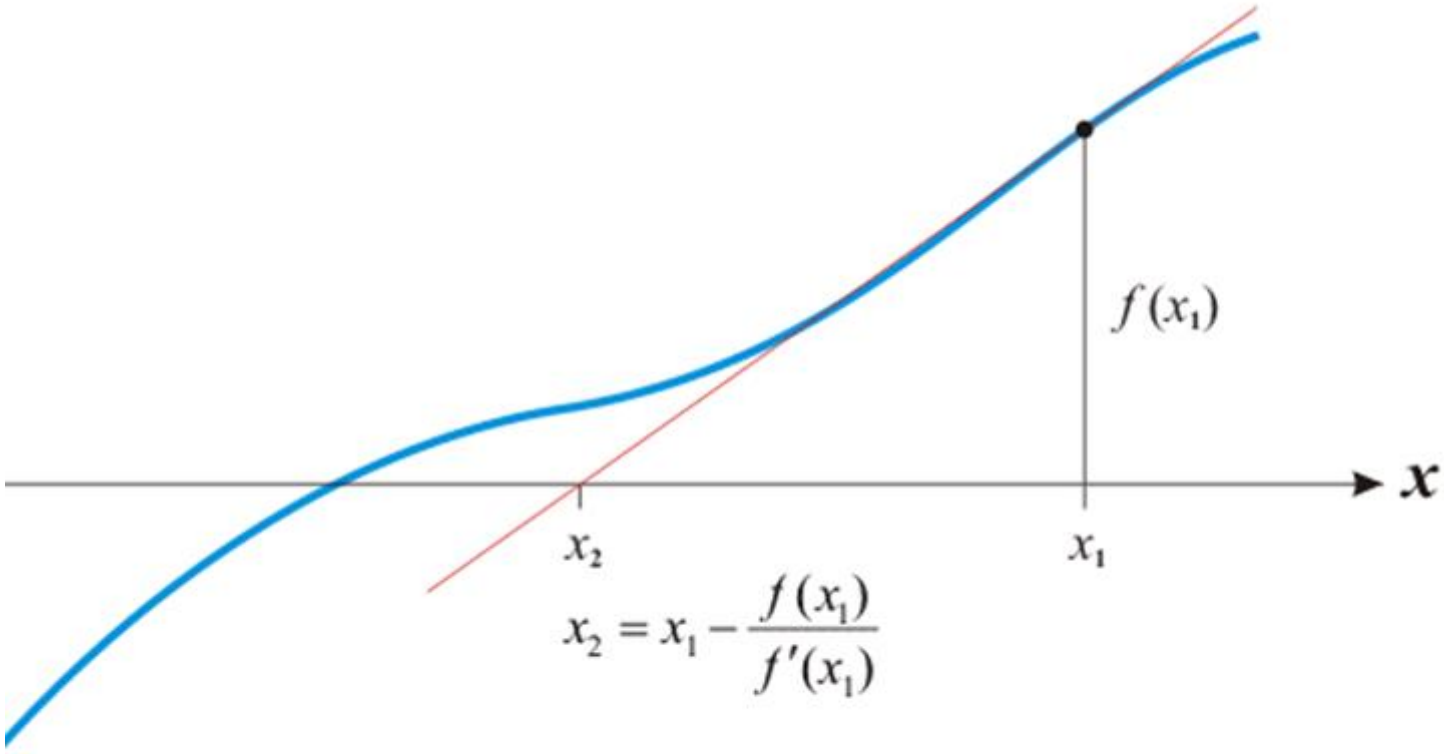
2. General equations: $\cos(x) = x$

Solving equations: example of iterative method



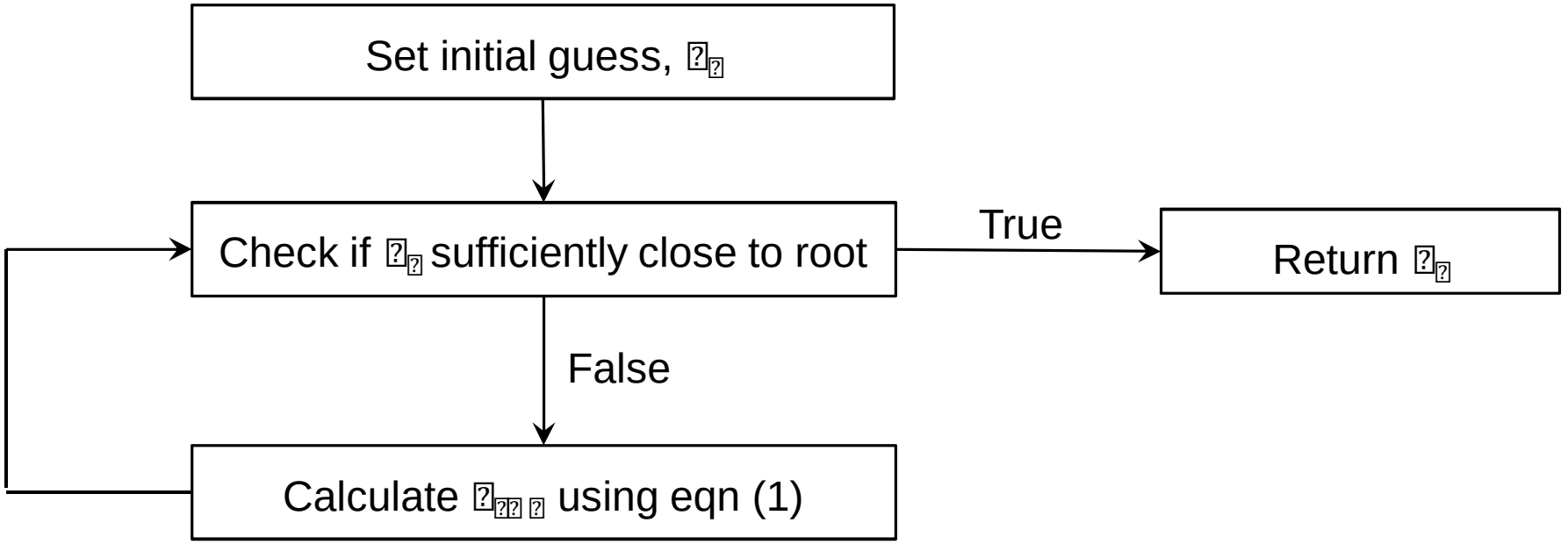
Solving equations: the Newton-Raphson method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad \text{eqn (1)}$$



Solving equations: the Newton-Raphson method

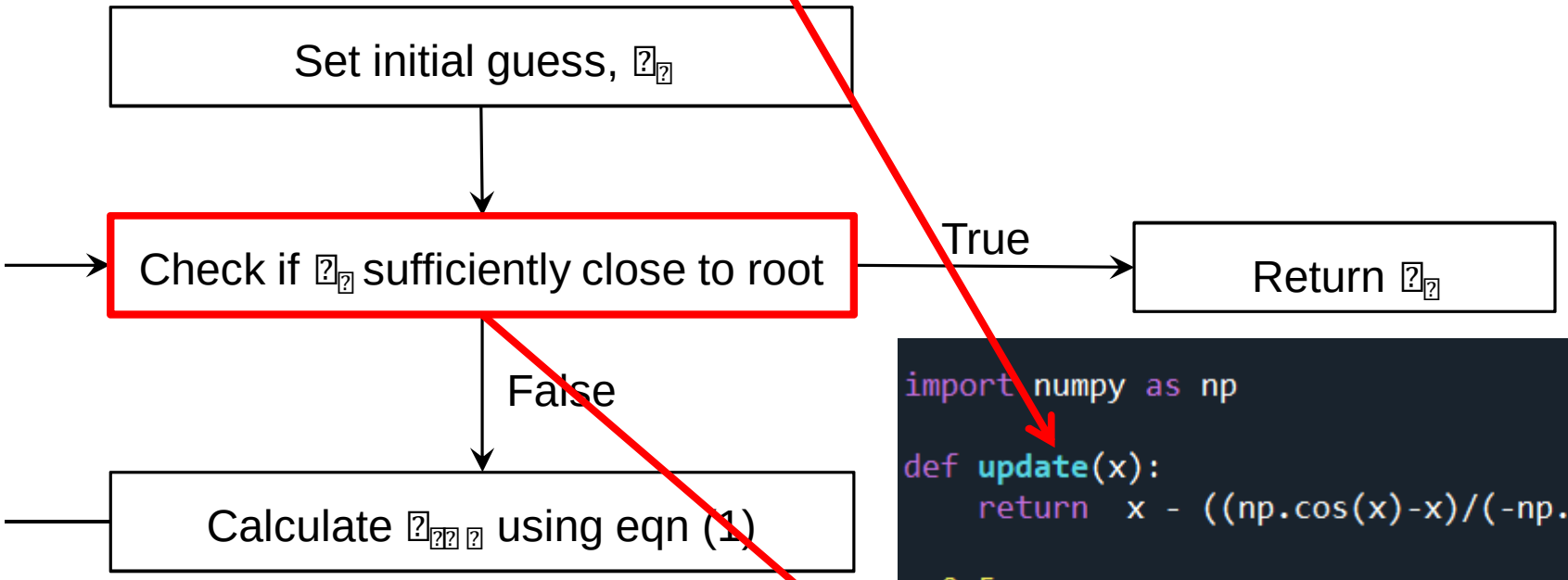
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad \text{eqn (1)}$$



Solving equations: the Newton-Raphson method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

eqn (1)




```python
import numpy as np

def update(x):
    return  x - ((np.cos(x)-x)/(-np.sin(x)-1))

x=0.5
dx=x
while abs(dx) > 0.000000001 :
    xnew = update(x)
    print("xnew = ", xnew)
    dx   = np.absolute(x-xnew)
    x    = xnew

print("The solution is x = ", x)
print("f(x)", np.cos(x)-x)
```

Solving equations

1. Polynomials: $3x^3 + x^2$ ï $x = 10$
   Use numpy.roots() function.

2. General equations: $\cos(x) = x$
   Use scipy.optimize.fsolve() function.

Both of these algorithms find the *roots* of given functions, so rewrite your function accordingly!

E.g. $\cos(x)$ ï $x = 0$, so find root of $\cos(x) - x$

Optimisation

- Optimisation usually involves *minimising* a parameter, e.g. what is the value of x such that f(x) is at a minimum?

- Analytically we do this by setting the derivative to zero.

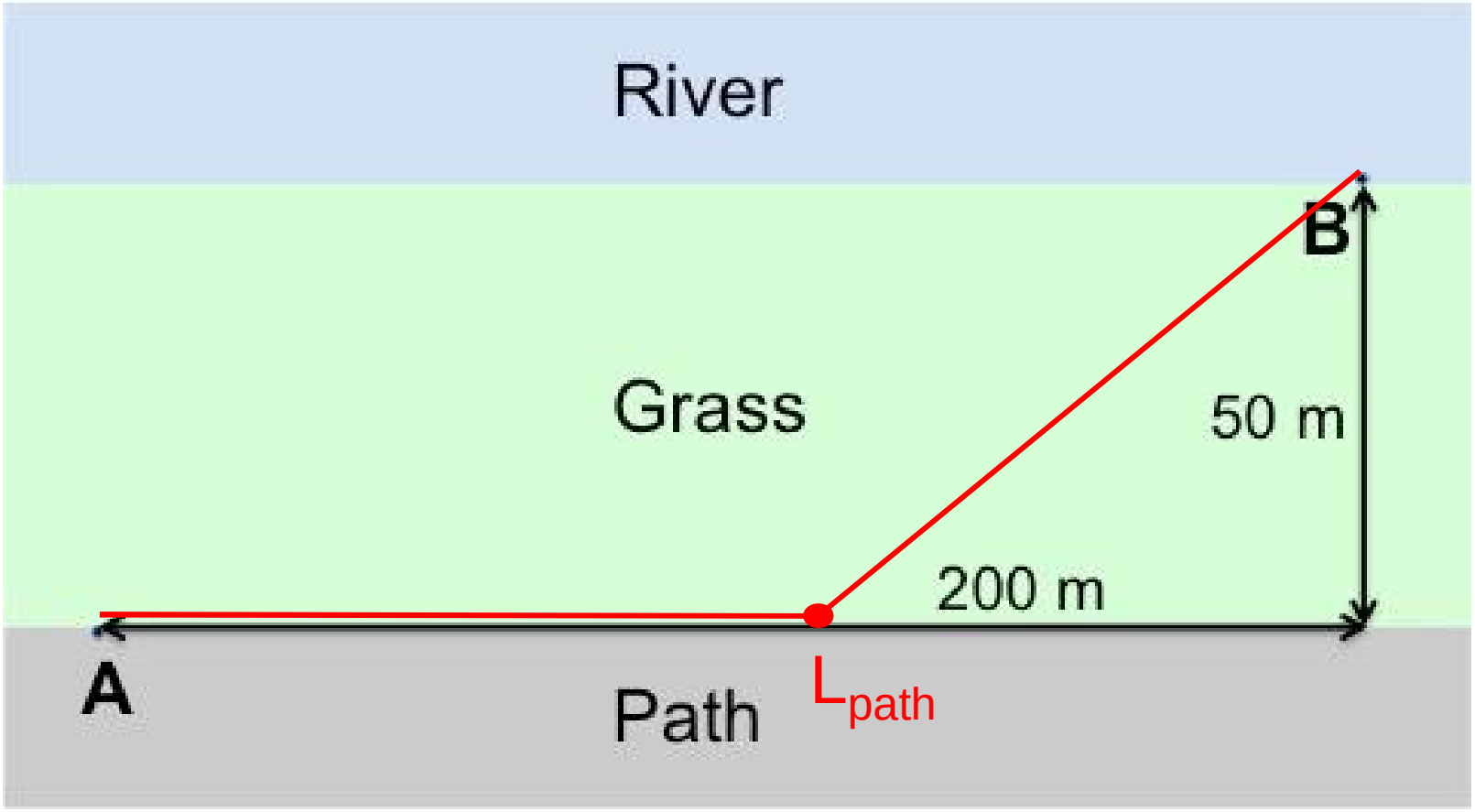
- Numerical methods can do this when derivatives are unknown.
  → scipy.optimize.fmin()

Example optimisation problem

A cyclist is biking along a path when she witnesses a child falling into the river. At this moment, the cyclist is at point *A* and the child at point *B*. The cyclist's maximum speed is 30 km/h on the path, but only 15 km/h on the grass separating the path from the river. At which point should the cyclist cross the grass to reach the child in the shortest time? How long does it take the cyclist to get to the child?

Example optimisation problem

What path should the cyclist take from A to B?



Minimise time for range of L_{path} between 0 ï 200m.

Differentiation, Integration and ODEôs

- Integrating and differentiating data is critical in calculating quantities relevant to your work.
- For differentiation you can use

$$f'(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}$$

Solving Ordinary Differential Equations (ODEs)

The rate of decay of a radioactive substance is described by a 1st-order ODE:

$$\frac{dN}{dt} = -\alpha N.$$

We can use scipy.integrate.odeint() to solve this, i.e. calculate *N* for a range of values *t*.

Solving Ordinary Differential Equations (ODEs)

```
import scipy.integrate as spi

alpha = 0.0050228

def dif_func(N,t):
    dN_dt = -alpha*N
    return dN_dt
```

Second-order ODEs

Newton's second law $$m\frac{d^2x}{dt^2} = F(x),$$

Write in terms of 1st-order ODEs:

$$\frac{dv}{dt} = F(x)/m, \qquad \frac{dx}{dt} = v.$$

Example: ball thrown up into the air:

$$\frac{dv}{dt} = -g, \qquad \frac{dx}{dt} = v.$$

The user-defined function:

Array with two variables [*x*,*v*]

```
def dif_func(variables,t):
    x = variables[0]
    v = variables[1]
    a = -g
    return [v,a]
```

Array with derivatives of two variables

Solving the ODEs

```
t = sp.linspace(0.,2.5,100)
variables0=[0.,10]

soln=spi.odeint(dif_func,variables0,t)
```

```
[[   0.          10.        ]
 [   0.24940057   9.75252525]
 [   0.49255178   9.50505051]
 [   0.72945363   9.25757576]
 [   0.96010611   9.01010101]
```

Summary: solving ODEs

1. Write down the Physics problem on paper:
   - ï 1st-order ODEs
   - ï Initial conditions

2. Write a function that calculates ODEs
   Input: variables to be differentiated & *t* (usually)
   Output: derivatives of input variables

3. Create array with initial conditions, an array with time steps, and call the ODE solver odeint().

Imperial College London

Computing - Lecture 5

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Dr. Aidan Crilly

a.crilly16@imperial.ac.uk

Lecture 5

1. Review of Session 4
2. Project: Calculating Hubble's constant

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Lecture 5 – Part 1

1. Review of Session 4
2. Project: Calculating Hubble's constant

Dr. Brian Appelbe

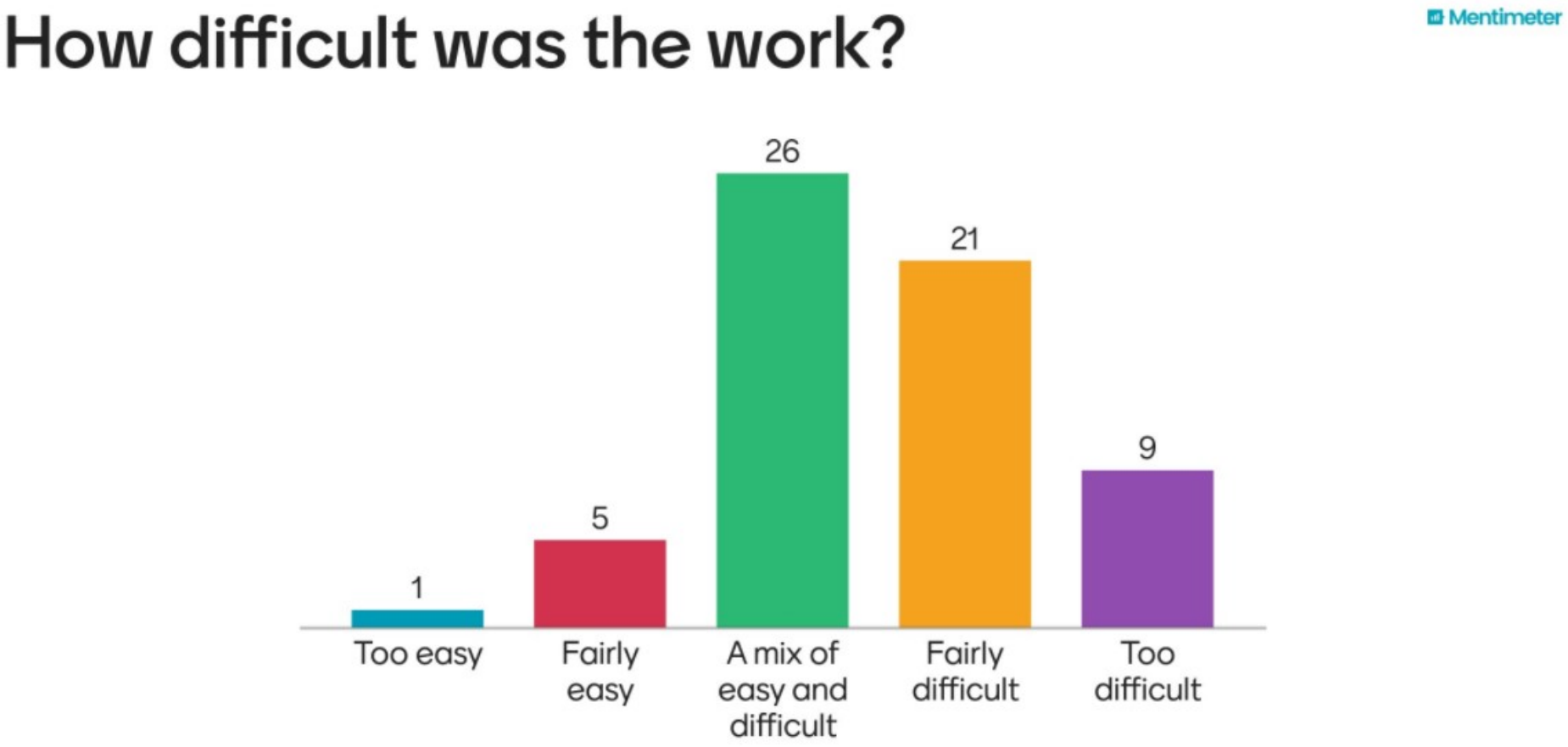
b.appelbe07@imperial.ac.uk

Session 4 -Numerical Methods

Topics covered:

- Solving equations - *roots*(), *fsolve*()
- Optimisation - *fmin*()
- Differentiation and Integration - *quad*(), *dblquad*()
- Ordinary Differential Equation solving - *odeint*()

Mentimeter Session 4 Result



General Feedback

- Many students found first two exercises on algorithms very challenging – Writing an algorithm requires planning and logical thinking.

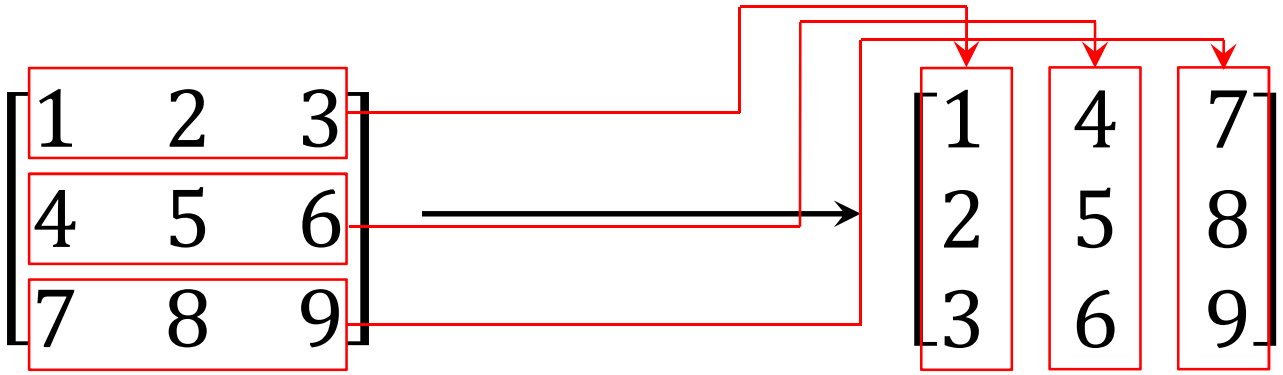
- We have covered a lot of material in the four Computing sessions – Doing the worksheets again in your own time is an excellent way to consolidate your knowledge.

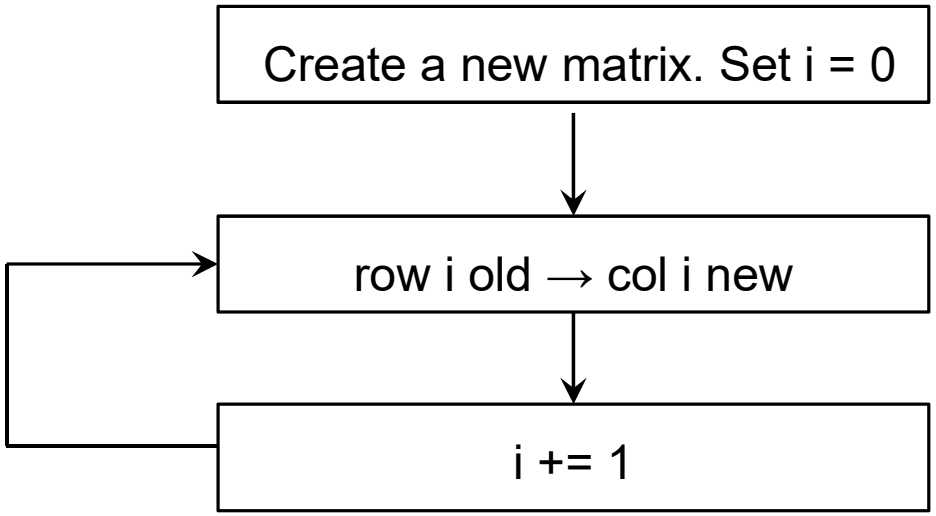
Writing Algorithms

Exercise: One common matrix operation is to calculate its transpose. This works by replacing its rows values with column values. Take the follow matrix:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

And calculate its transpose.





Writing Algorithms

Exercise: One common function of a Python list is the sort command, which takes an unordered sequence of numbers and returns an ordered list like:

$$list1 = [2, 4, 3, 6, 8, 7, 9, 5] -- > list2 = [2, 3, 4, 5, 6, 7, 8, 9]$$

Take list1 and write an algorithm that will return the ordered list2. This is a non-trivial task and there are multiple ways that the problem can be attempted

$$[2, 4, 3, 6, 8, 7, 9, 5]$$

$$[2, 3, 4, 6, 8, 7, 9, 5]$$

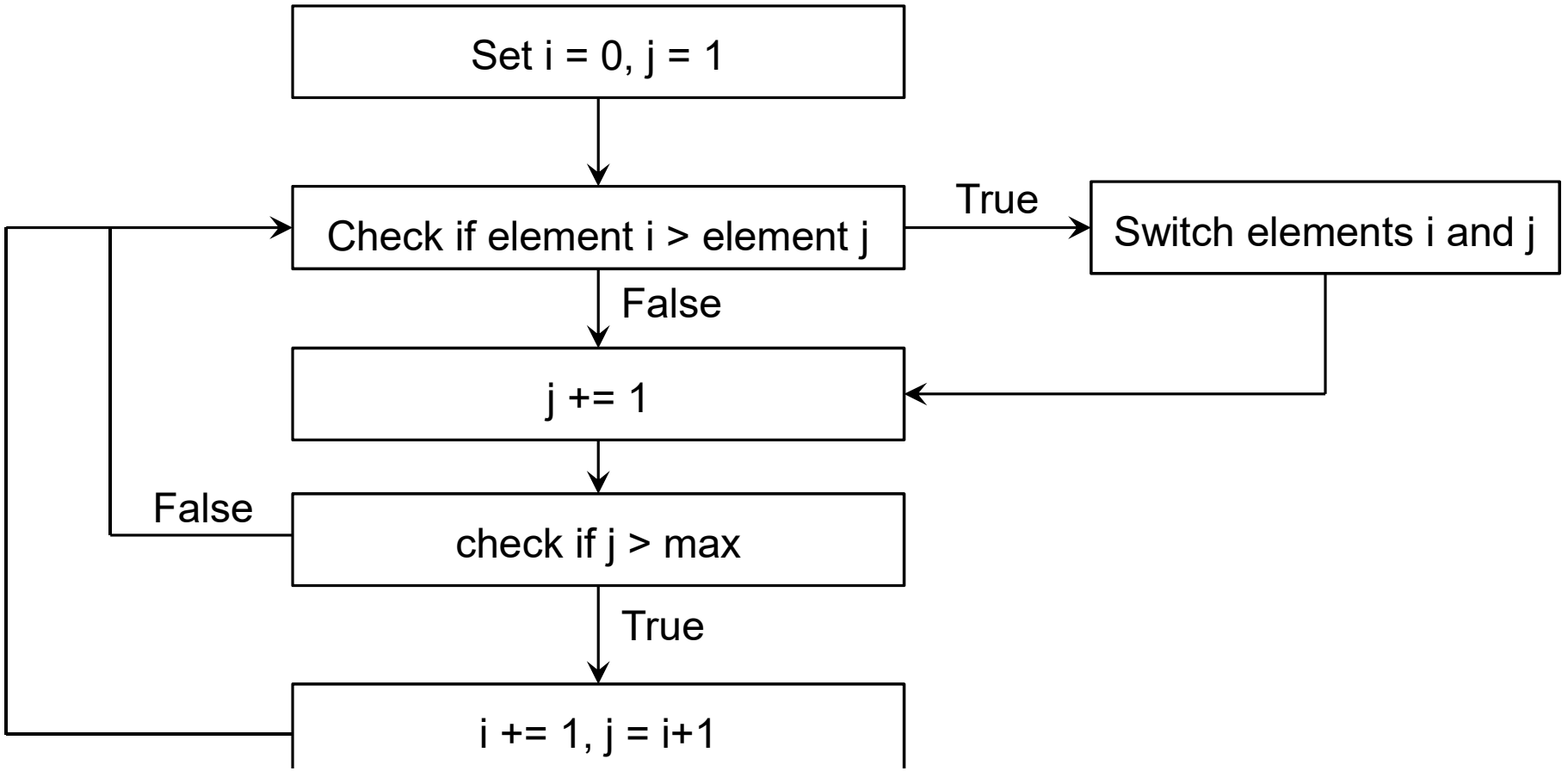
$$[2, 3, 4, 5, 8, 7, 9, 6]$$

Writing Algorithms

Exercise: One common function of a Python list is the sort command, which takes an unordered sequence of numbers and returns an ordered list like:

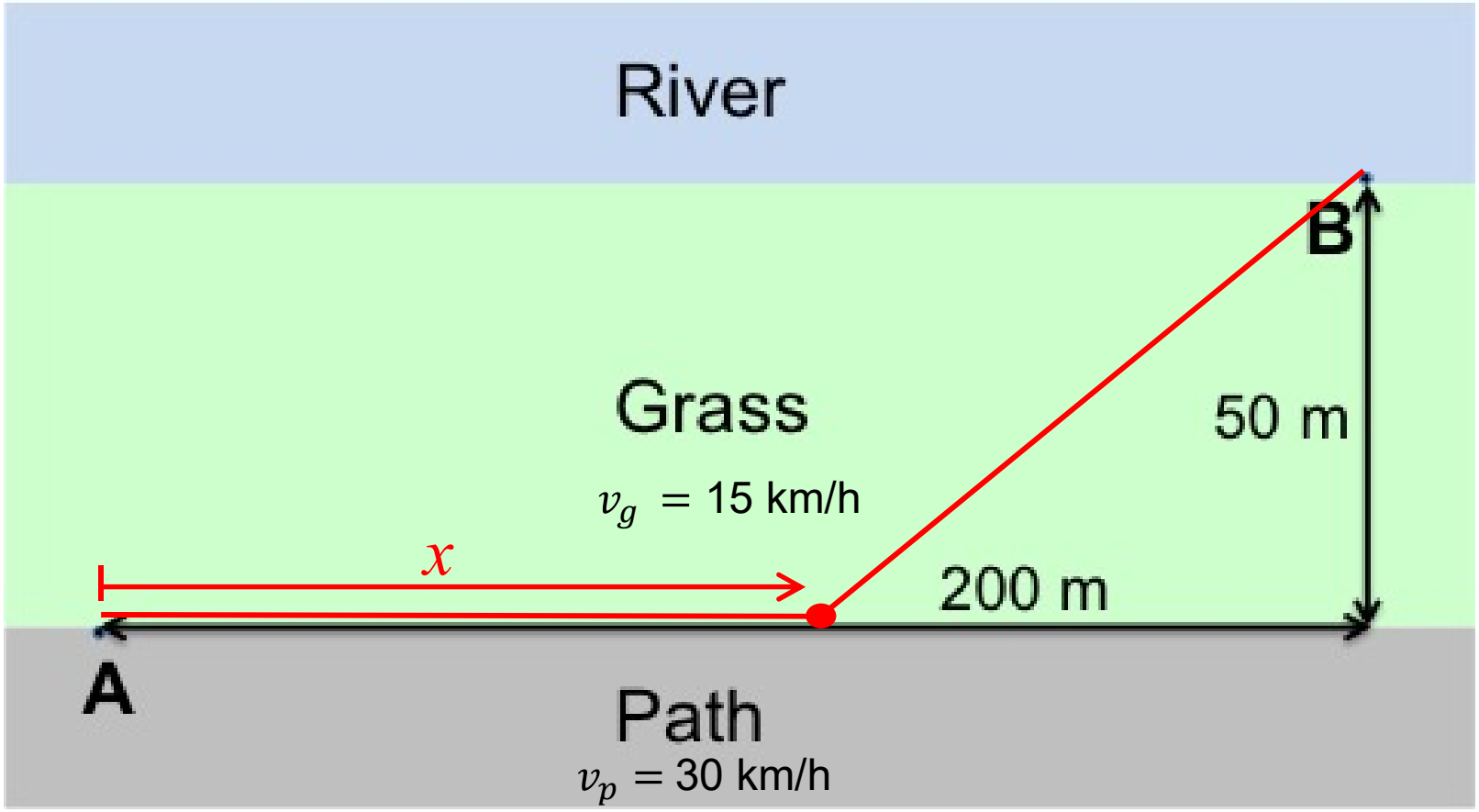
$$list1 = [2, 4, 3, 6, 8, 7, 9, 5] -- > list2 = [2, 3, 4, 5, 6, 7, 8, 9]$$

Take list1 and write an algorithm that will return the ordered list2. This is a non-trivial task and there are multiple ways that the problem can be attempted



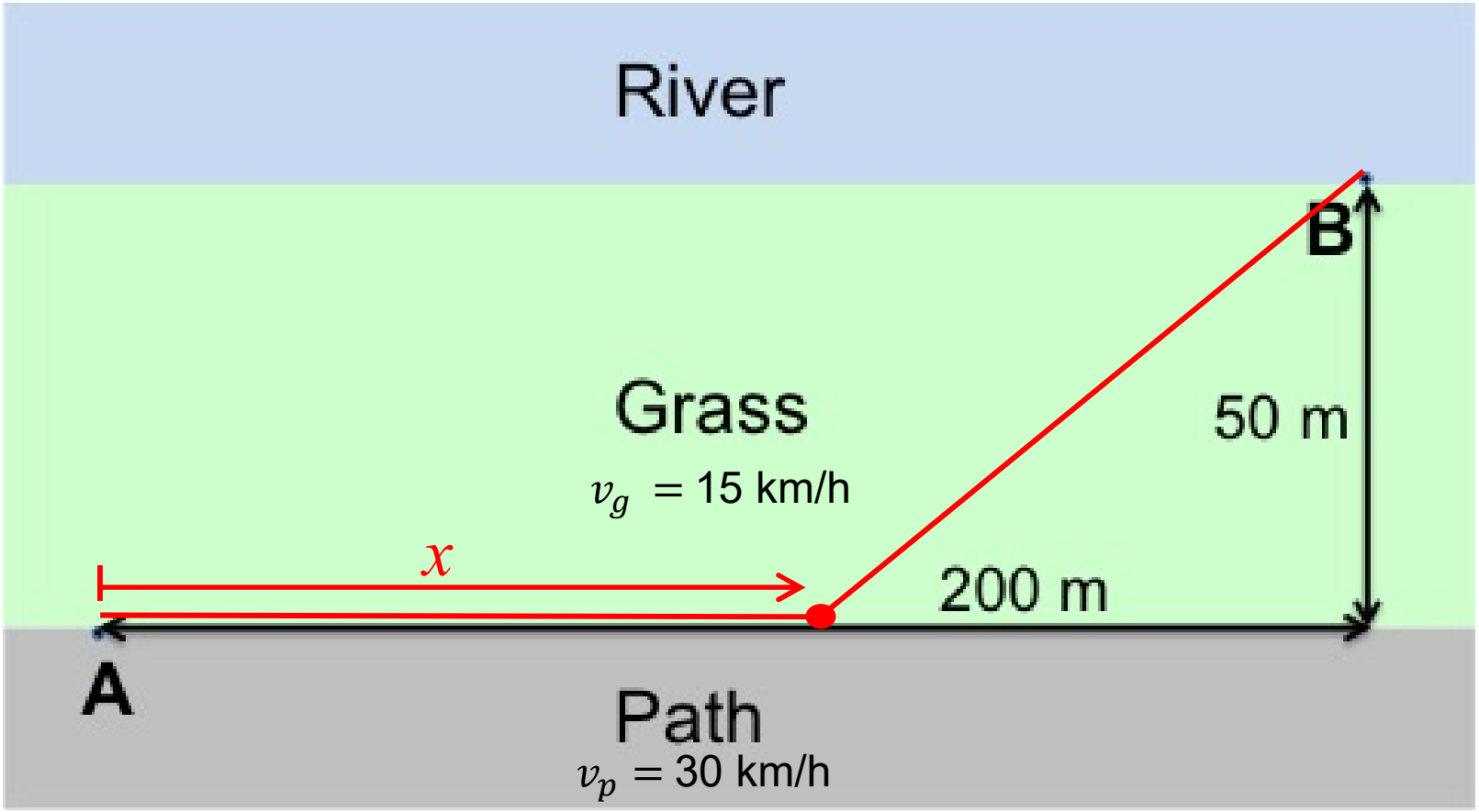
Optimisation problem

What path should the cyclist take from A to B?



Minimise time for range of x between 0 – 200m.

Optimisation problem



$$t = \frac{x}{v_p} + \frac{\sqrt{50^2 + (200 - x)^2}}{v_g}$$

Solving Ordinary Differential Equations (ODEs)

The rate of decay of a radioactive substance is described by a 1st-order ODE:

$$\frac{dN}{dt} = -\alpha N.$$

We can use scipy.integrate.odeint() to solve this, i.e. calculate *N* for a range of values *t*.

Summary: solving ODEs

1. Write down the Physics problem on paper:
   - 1^{st}-order ODEs
   - Initial conditions

2. Write a function that calculates ODEs
   Input: variables to be differentiated & *t* (usually)
   Output: derivatives of input variables

3. Create array with initial conditions, an array with time steps, and call the ODE solver odeint().

Year 1 Computing course: aims and structure

Aim: to be able to use Python to analyse your lab data and simulate physical scenarios.

- Session 1: Manipulating arrays and plotting data
- Session 2: Fitting data and error bars
- Session 3: Control Flow - Logic statements, loops, and functions
- Session 4: Numerical modelling
- Project: Using all your skills!

Year 1 Computing course: aims and structure

Aim: to be able to use Python to analyse your lab data and simulate physical scenarios.

- Session 1: Manipulating arrays and plotting data
- Session 2: Fitting data and error bars
- Session 3: Control Flow - Logic statements, loops, and functions
- Session 4: Numerical modelling
- Project: Using all your skills!

Lecture 5 – Part 2

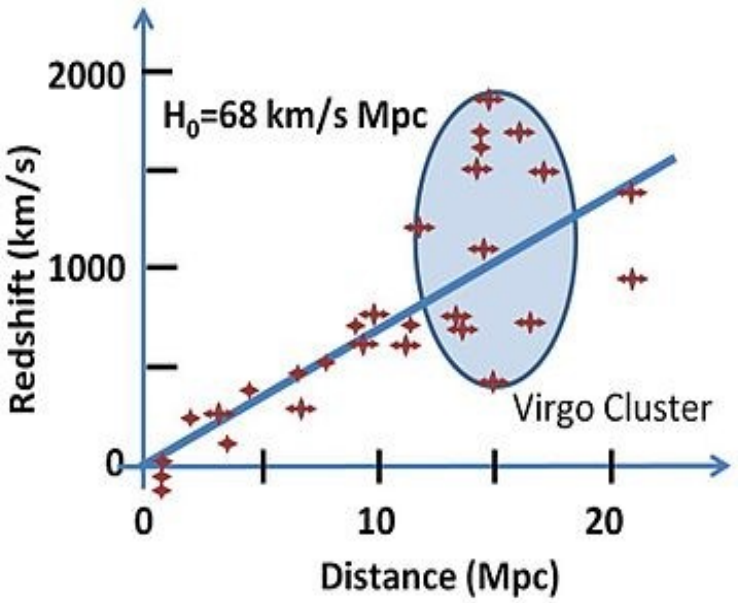
1. Review of Session 4
2. Project: Calculating Hubble's constant

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

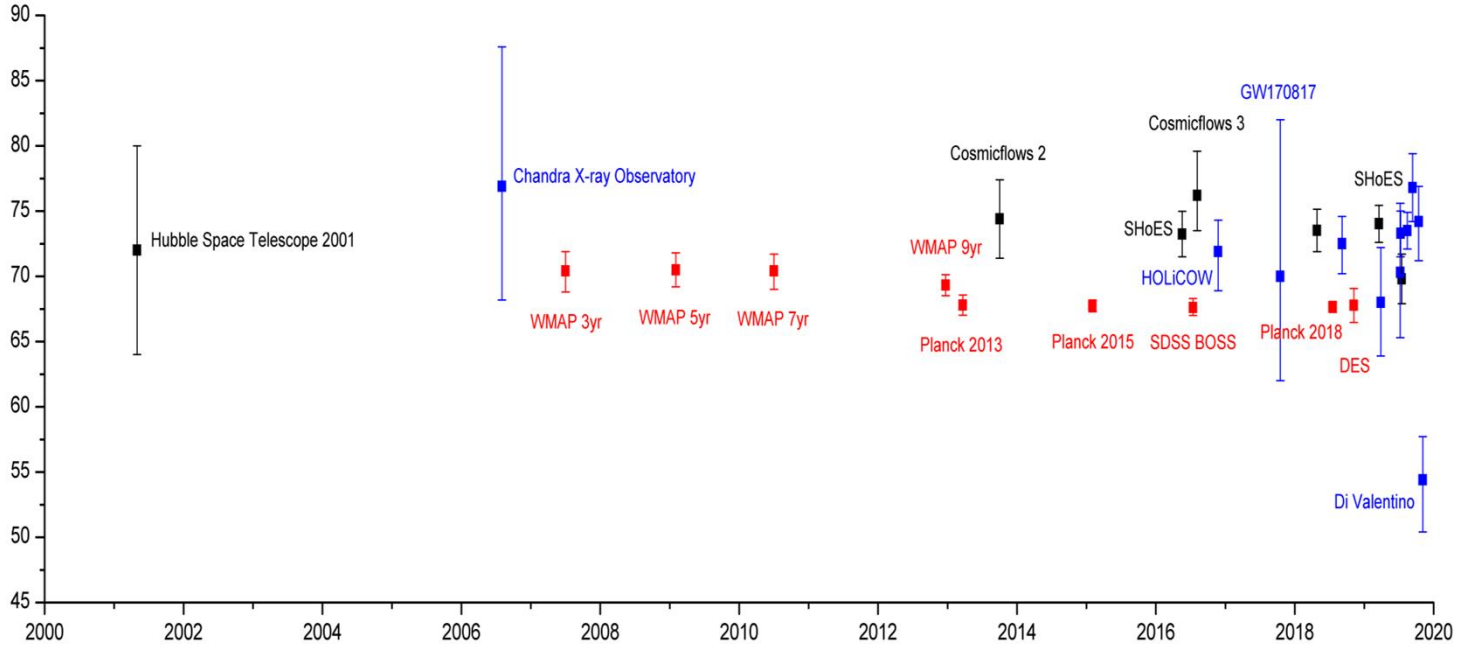
Hubble's Constant & the Expanding Universe

- Galaxies have a velocity directed away from Earth.
- This velocity is proportional to the distance of the galaxy from earth, $v = H_0 D$.
- H_0 is Hubble's constant, named after astronomer Edwin Hubble.
- The velocity is measured by the amount of "redshift" (i.e. doppler-shift) of light emitted by the galaxies.
- The distance is measured from the brightness of the galaxy.



Hubble's Constant Today

- In 1998, measurements of redshifts from Supernovae showed that the expansion of the Universe was accelerating, possibly due to dark energy.
- Measurements of Hubble's constant using CMB do not agree with value from redshifts.



Project: Calculating Hubble's constant from redshift data

- Calculate a value for H_0 given 30 observations of galaxy redshifts

Data Given:

- 1 file containing 30 spectral intensity measurements
- 1 file containing distance to galaxy for every observation and instrument response

Project: Calculating Hubble's constant from redshift data

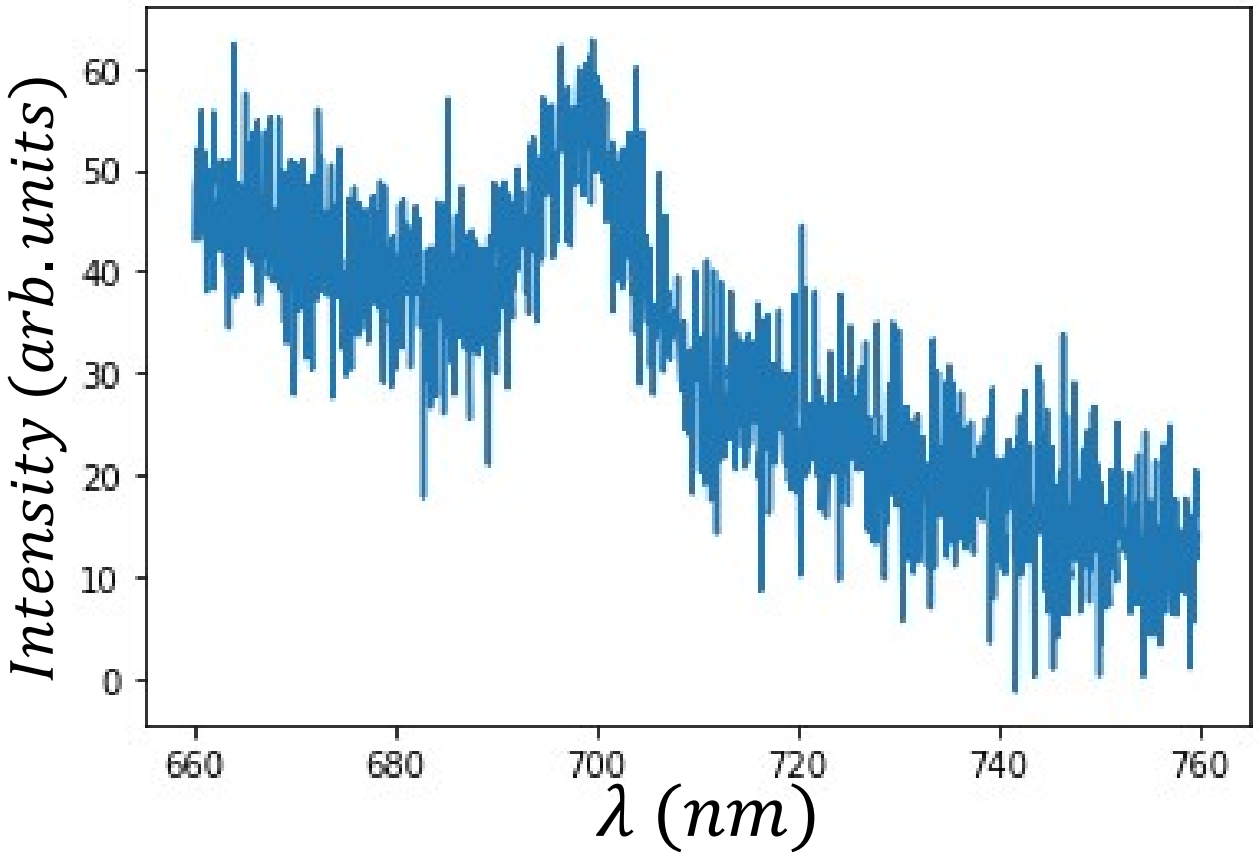
To Do:

- Fit the spectral data for every observation and find the shifted wavelength
- Calculate the redshift velocity from the shifted wavelength
- Make a plot of redshift velocity vs distance
- Carry out a linear fit to get H_0

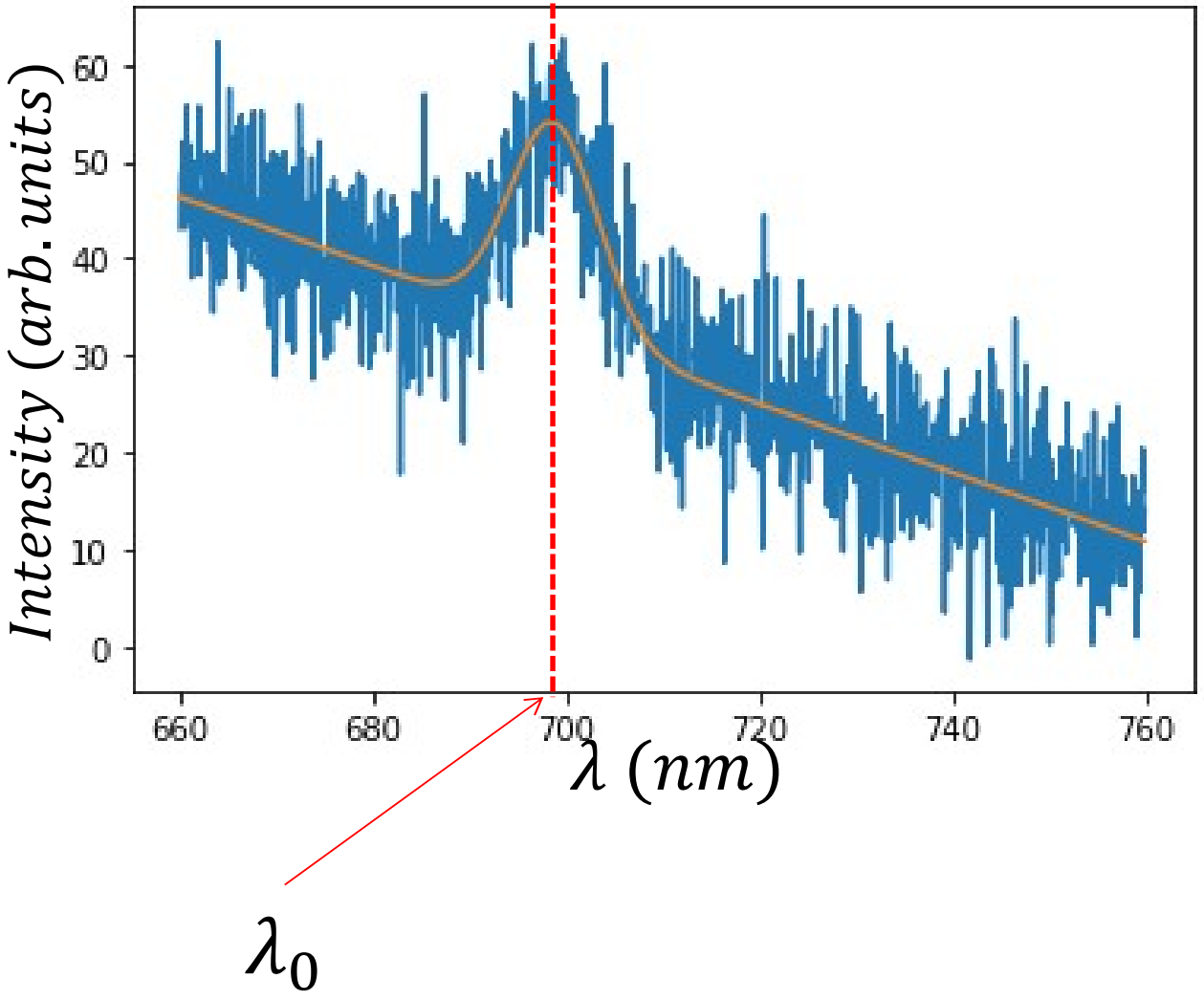
To Submit:

- Your source code (.py file)
- Plot of redshift velocity vs distance
- Value for H_0 with uncertainty due to fitting

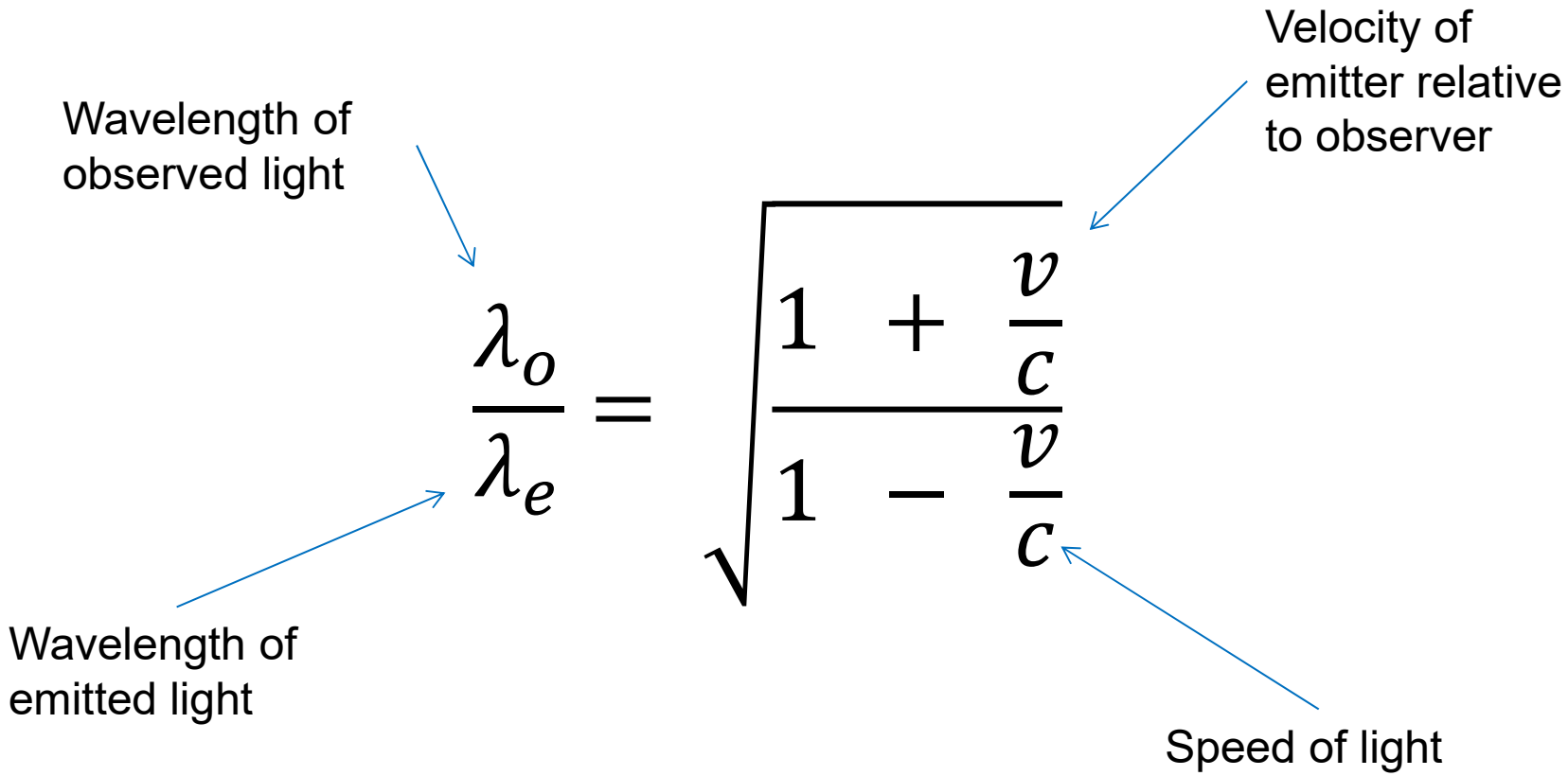
Spectral Intensity vs. Wavelength



Spectral Intensity vs. Wavelength - Finding λ_0



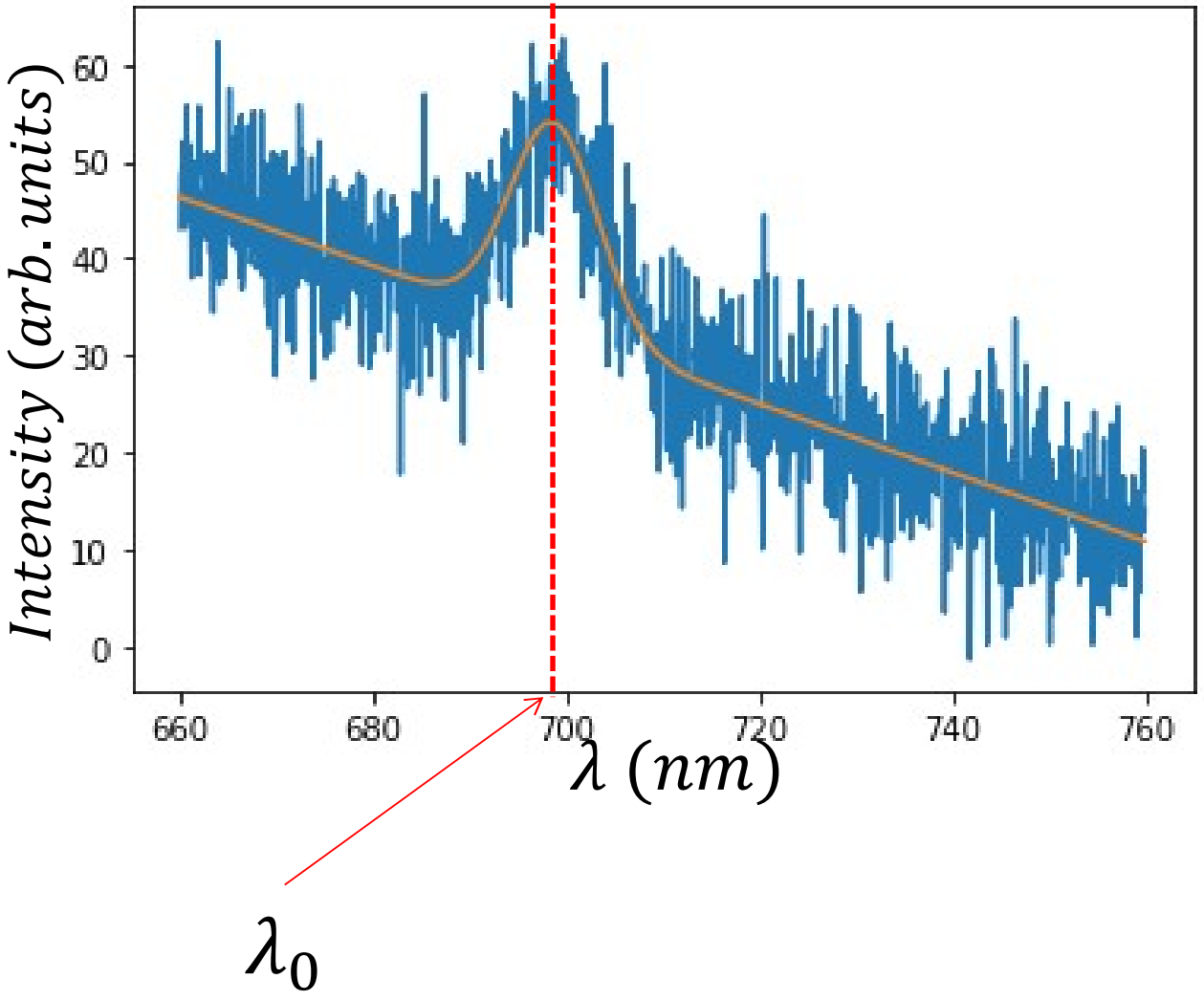
Redshift formula – relationship between velocity and observed wavelength



$$\lambda_e = 656.28\, nm$$

$$c = 2.9979 \times 10^8\, ms^{-1}$$

Spectral Intensity vs. Wavelength - Finding λ_0



Finding λ_0

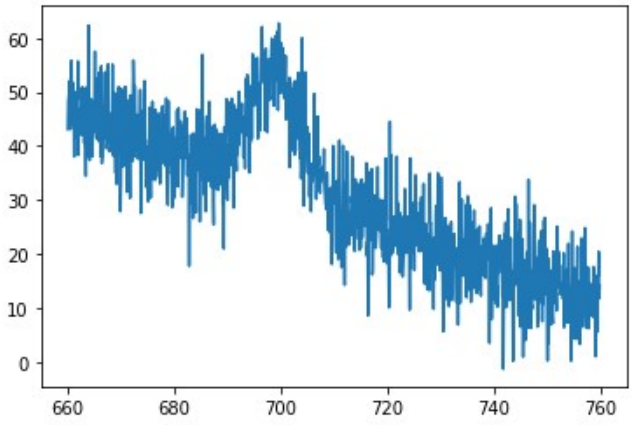
From Core Worksheet 2:

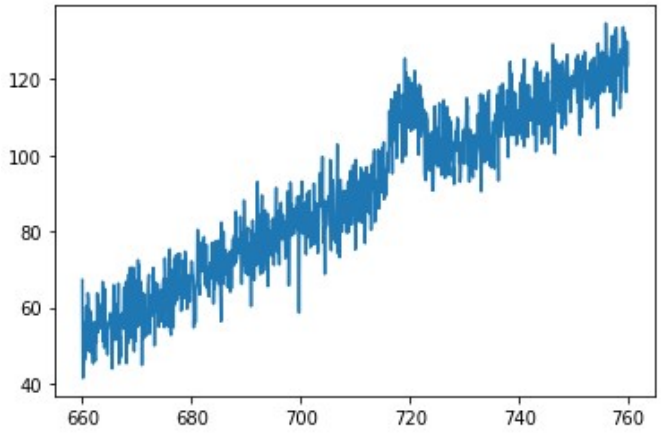
Exercise: use the curve_fit() function to fit a straight line for the background and a Gaussian function for the feature. Hence, find the amplitude and width of the Gaussian feature. Note that in this case it is not a good idea to fit the line first and afterwards the Gaussian from the residuals, as the Gaussian will influence the line fit. Instead, fit both background and the Gaussian feature at once.

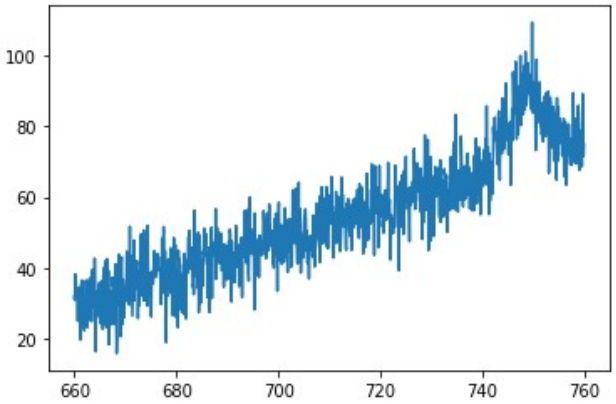
```
def fit_func(x,a,mu,sig,m,c):
    gaus = a*sp.exp(-(x-mu)**2/(2*sig**2))
    line = m*x+c
    return gaus + line

initial_guess=[10,120,10,-2,100]
po,po_cov=sp.optimize.curve_fit(fit_func,x,y,initial_guess,y_error)
```

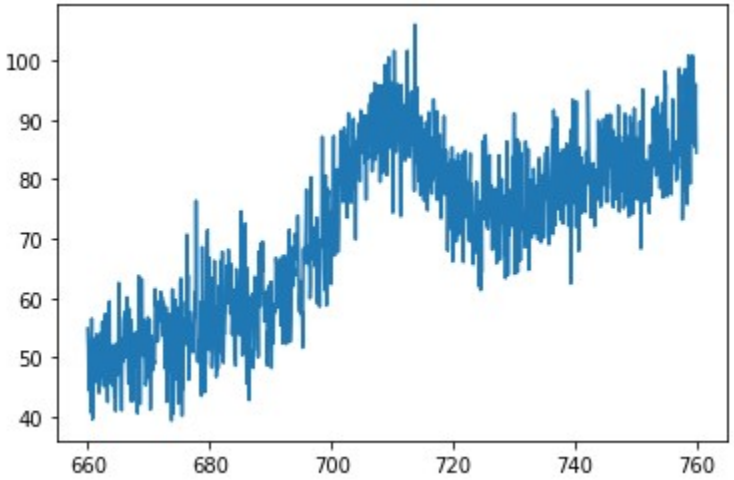
Can you automate the procedure for obtaining the initial guess?

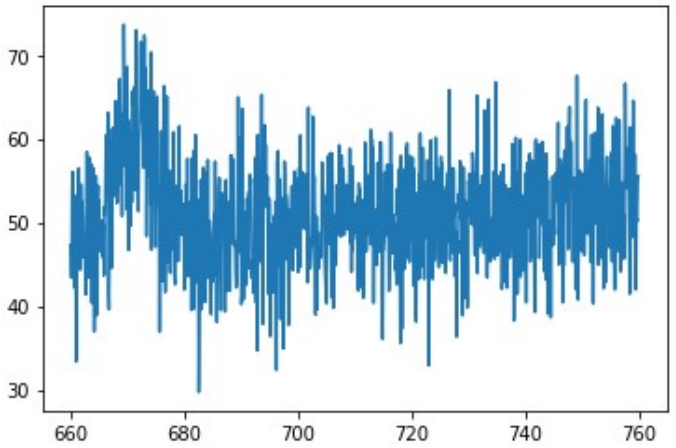
Finding λ_0



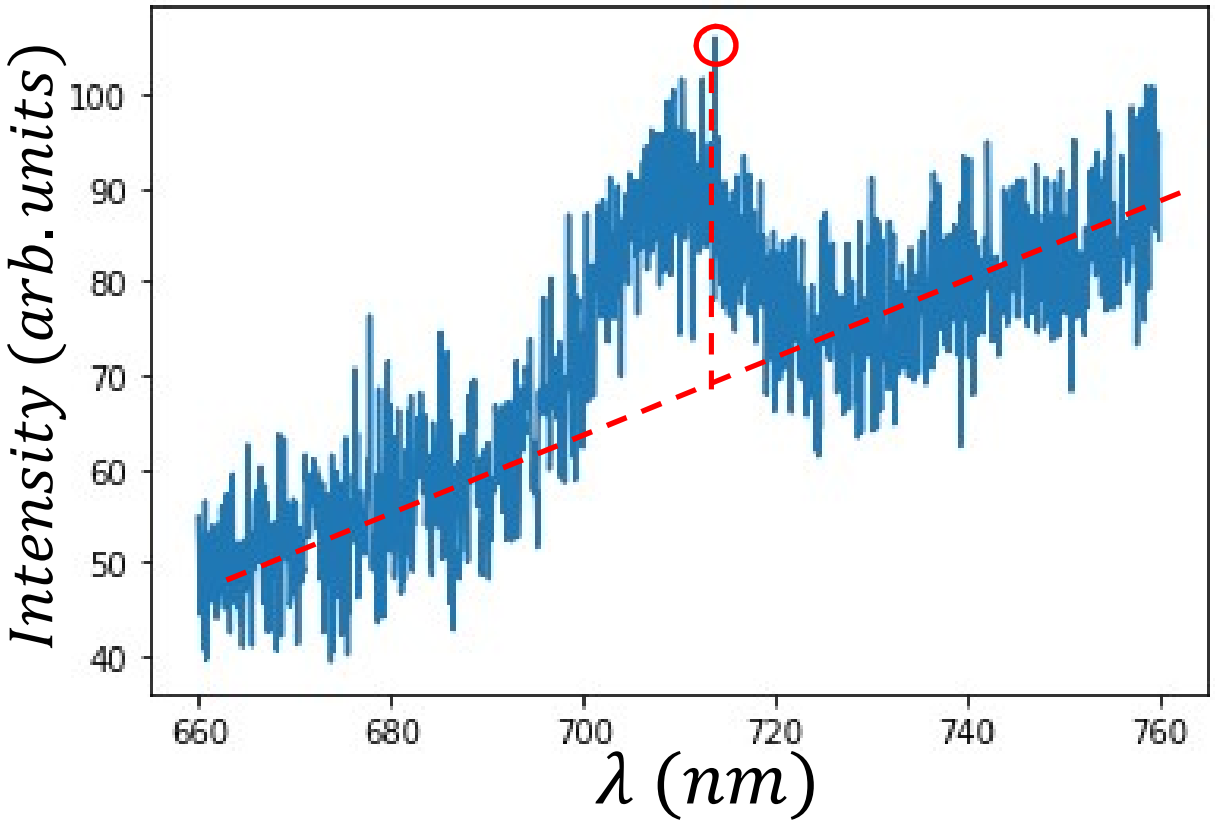






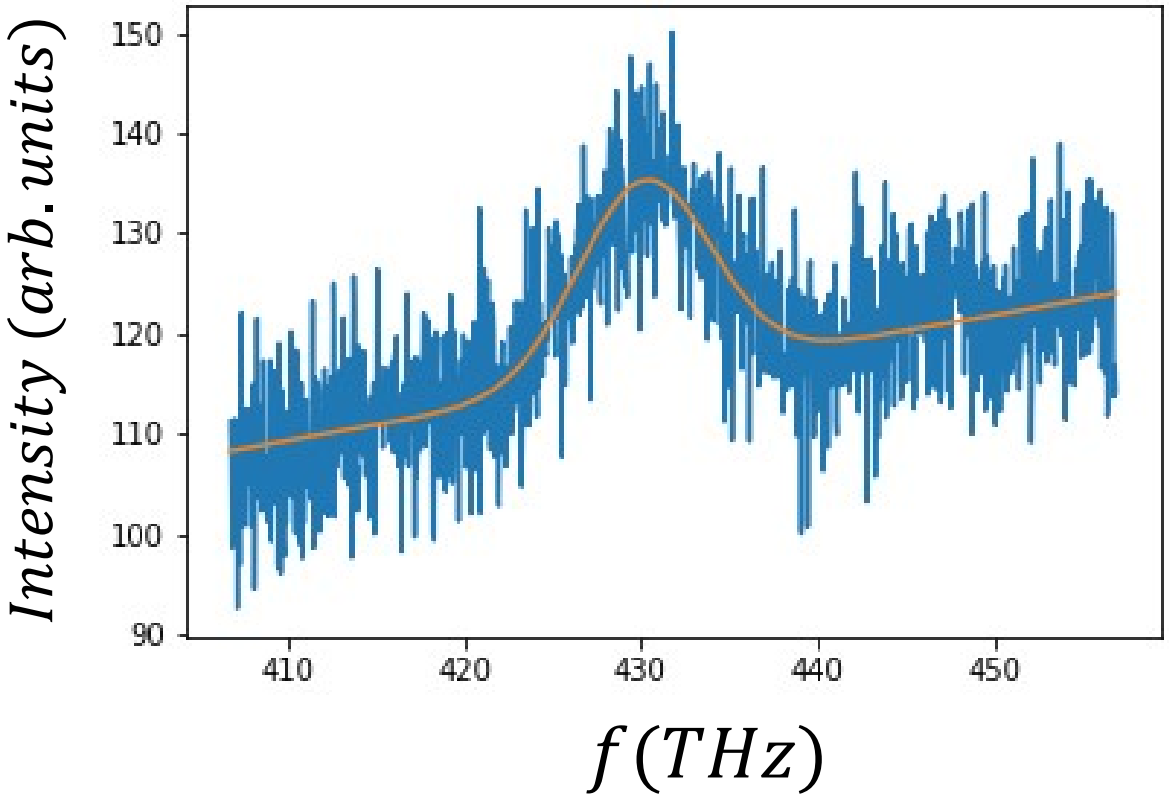


Finding λ_0



- Estimate slope & intercept of line using points at beginning and end of data
- Estimate location of gaussian peak & gaussian amplitude from difference between data and line estimate

Wavelength & frequency



- The spectral data is in a function of frequency

Finding λ_0 - Instrument Response & File Reading

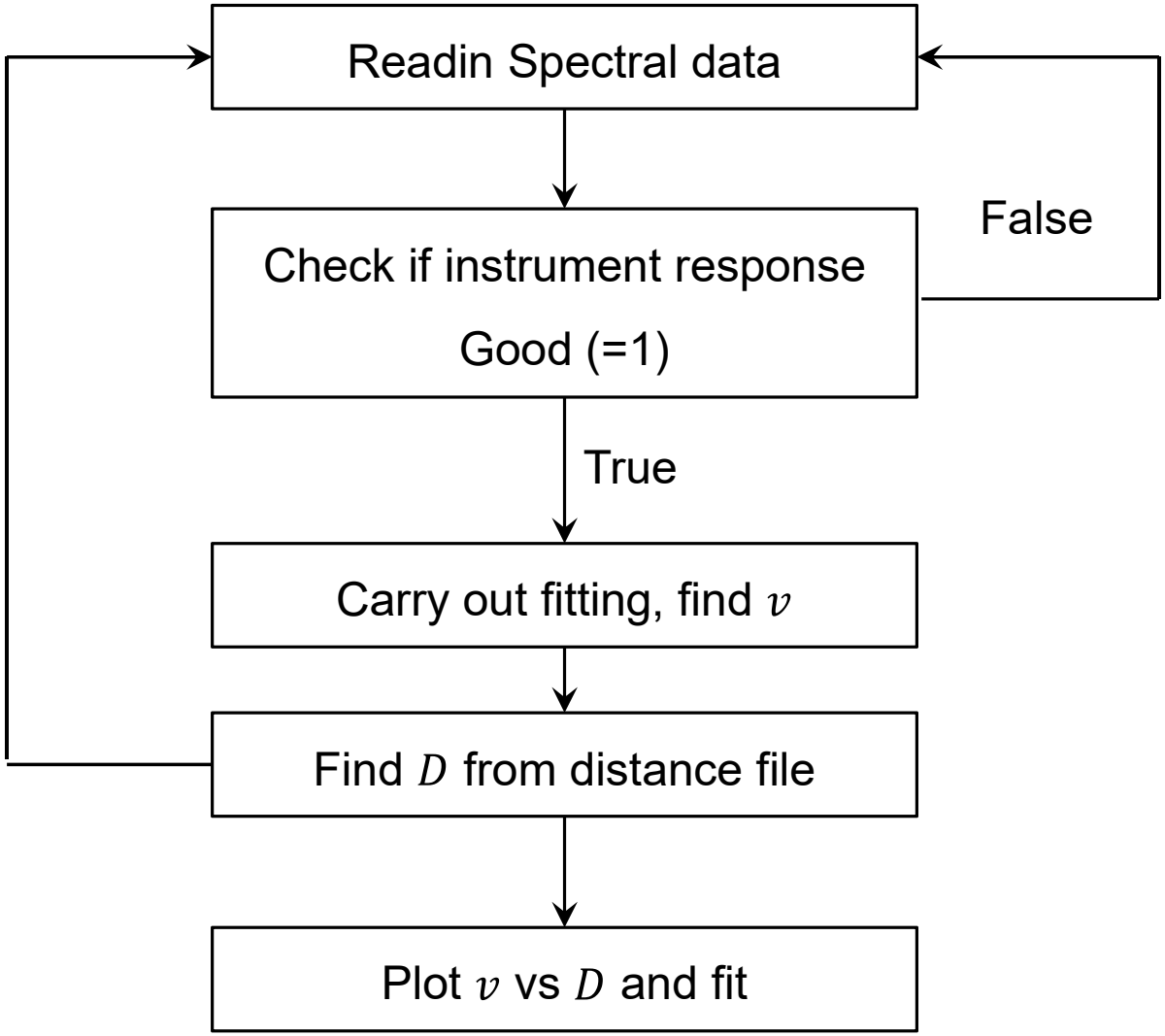
- Not all observations can be used
- If the instrument response is bad (=0) then the data should be excluded
- Bad instrument response can arise from: low intensity signal, drift in calibration wavelength due to thermal expansion, atmospheric interference, etc.
- Can you write code to automatically exclude bad instrument response data?
- Can you write code to match observation numbers from distance file with those in spectral intensities file?

Observation	Distance(Mpc)	Valid Instrument Response
23522	517.9323713	0
31991	363.2371155	1
31544	213.5461456	1
41090	267.8456873	1
19909	295.9881898	1
43937	198.2728487	1
39451	405.9085394	1
36537	250.8782078	1
27028	502.9875908	0
23095	483.3087199	1
16076	144.7971076	1
30915	408.9550979	1
30768	412.5117835	1
47565	155.91514	1
36999	387.4582306	1

# Date: 19	Author: E.Hubble							
# First column is observation number								
# First row is frequency (Hz)								
# All other rows are spectra								
0	4.07E+14	4.07E+14	4.07E+14	4.07E+14	4.07E+14	4.07E+14	4.07E+14	4.07E+
19909	1.23E+02	1.29E+02	1.28E+02	1.20E+02	1.36E+02	1.33E+02	1.29E+02	1.32E+
27028	9.89E+01	1.07E+02	1.09E+02	1.14E+02	1.07E+02	1.18E+02	1.06E+02	1.15E+
16652	1.35E+02	1.32E+02	1.36E+02	1.18E+02	1.33E+02	1.36E+02	1.23E+02	1.30E+
31458	1.16E+02	1.24E+02	1.24E+02	1.18E+02	1.23E+02	1.23E+02	1.23E+02	1.25E+
47548	1.10E+02	1.19E+02	1.07E+02	1.10E+02	1.16E+02	1.14E+02	1.14E+02	1.20E+
21100	1.34E+02	1.36E+02	1.20E+02	1.28E+02	1.32E+02	1.31E+02	1.21E+02	1.37E+
11768	1.31E+02	1.24E+02	1.32E+02	1.37E+02	1.27E+02	1.28E+02	1.22E+02	1.20E+
36969	1.11E+02	1.07E+02	9.87E+01	1.10E+02	1.12E+02	1.09E+02	1.09E+02	1.06E+
36656	1.14E+02	1.31E+02	1.22E+02	1.26E+02	1.28E+02	1.26E+02	1.17E+02	1.24E+
14915	1.42E+02	1.35E+02	1.45E+02	1.37E+02	1.41E+02	1.45E+02	1.37E+02	1.34E+
16766	1.22E+02	1.25E+02	1.21E+02	1.09E+02	1.08E+02	1.25E+02	1.28E+02	1.35E+
18197	1.47E+02	1.44E+02	1.41E+02	1.41E+02	1.48E+02	1.47E+02	1.40E+02	1.47E+
30915	1.29E+02	1.41E+02	1.33E+02	1.38E+02	1.25E+02	1.36E+02	1.31E+02	1.31E+
41090	1.48E+02	1.56E+02	1.51E+02	1.52E+02	1.44E+02	1.35E+02	1.42E+02	1.38E+
48054	1.40E+02	1.44E+02	1.39E+02	1.35E+02	1.45E+02	1.48E+02	1.41E+02	1.50E+

Plan your work!

- Make a flow diagram, write pseudocode, etc.
- Break the problem down into smaller pieces



Comments & Plagiarism:

- The Project will be checked for plagiarism.
- Comments should be used to explain how your code works.
- It is ok to use pieces of code from other sources but you must use comments to state the source of the code.

b.appelbe07@imperial.ac.uk a.crilly16@imperial.ac.uk

Notes:

- Project Outline and data files are on Blackboard

- Deadline: 5pm on Friday 26th November
- Details on how to submit (via blackboard) will be provided later this week
- Submission is mandatory (10% PP module)

- Drop-in sessions: Thur 11th, Tues 16th & 23rd Nov, 12-1pm in Blackett Computing Suite.

- Final Lecture: Monday 29th November

b.appelbe07@imperial.ac.uk a.crilly16@imperial.ac.uk

Imperial College London

Computing - Lecture 6

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Dr. Aidan Crilly

a.crilly16@imperial.ac.uk

Lecture 6

1. Project Summary & Results
2. Example Solution

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

Lecture 6 – Part 1

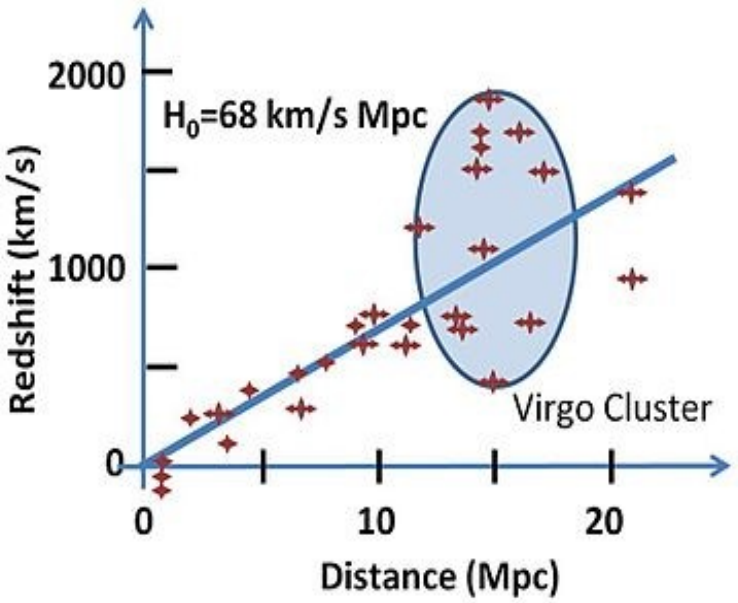
1. Project Summary & Results
2. Example Solution

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

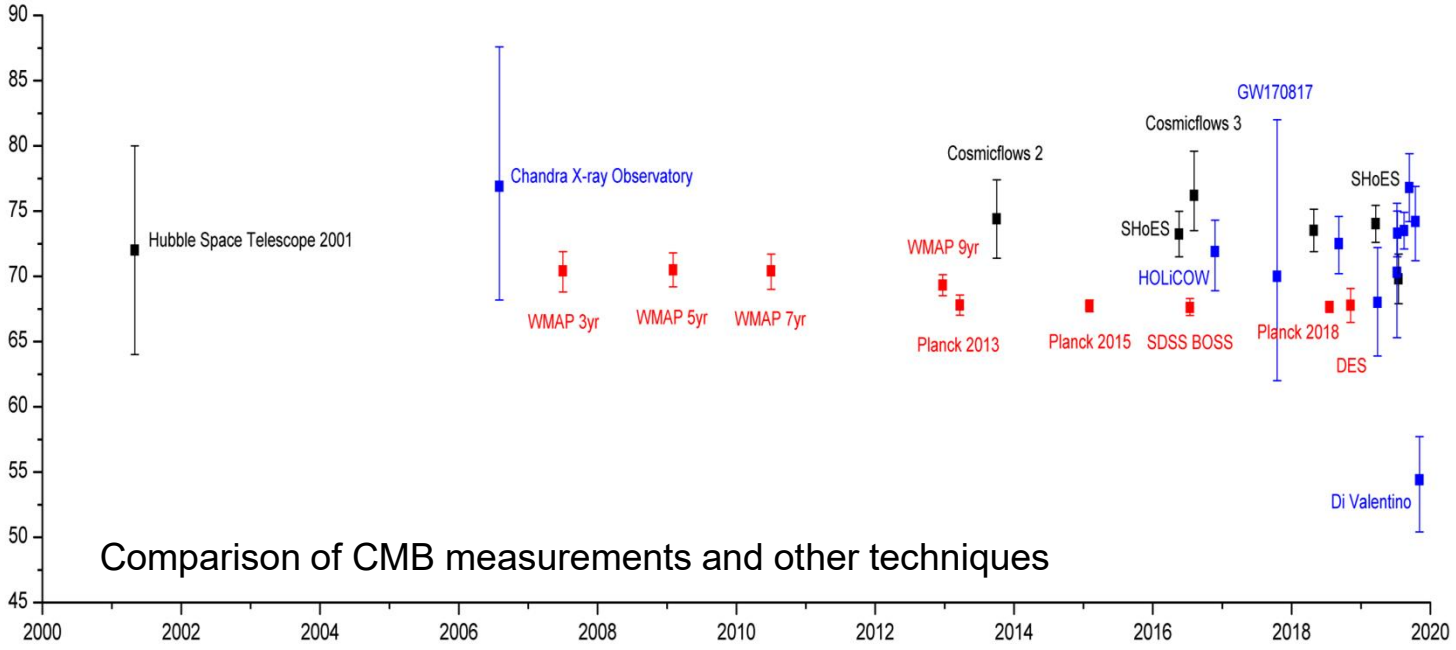
Hubble's Constant & the Expanding Universe

- Galaxies have a velocity directed away from Earth.
- This velocity is proportional to the distance of the galaxy from earth, $v = H_0 D$.
- H_0 is Hubble's constant, named after astronomer Edwin Hubble.
- The velocity is measured by the amount of "redshift" (i.e. doppler-shift) of light emitted by the galaxies.
- The distance is measured from the brightness of the galaxy.



Hubble's Constant Today

- In 1998, measurements of redshifts from Supernovae showed that the expansion of the Universe was accelerating, possibly due to dark energy.
- Measurements of Hubble's constant using CMB do not agree with value from redshifts.



Comparison of CMB measurements and other techniques

Project: Calculating Hubble's constant from redshift data

- Calculate a value for H_0 given 30 observations of galaxy redshifts and distances
- Ensure only measurements with a good instrument response are included

Observation	Distance(Mpc)	Valid Instrument Response
23522	517.9323713	0
31991	363.2371155	1
31544	213.5461456	1
41090	267.8456873	1
19909	295.9881898	1
43937	198.2728487	1
39451	405.9085394	1
36537	250.8782078	1
27028	502.9875908	0
23095	483.3087199	1
16076	144.7971076	1
30915	408.9550979	1
30768	412.5117835	1
47565	155.91514	1
36999	387.4582306	1

# Date: 19	Author: E.Hubble							
# First column is observation number								
# First row is frequency (Hz)								
# All other rows are spectra								
0	4.07E+14	4.07E+14	4.07E+14	4.07E+14	4.07E+14	4.07E+14	4.07E+14	4.07E+
19909	1.23E+02	1.29E+02	1.28E+02	1.20E+02	1.36E+02	1.33E+02	1.29E+02	1.32E+
27028	9.89E+01	1.07E+02	1.09E+02	1.14E+02	1.07E+02	1.18E+02	1.06E+02	1.15E+
16652	1.35E+02	1.32E+02	1.36E+02	1.18E+02	1.33E+02	1.36E+02	1.23E+02	1.30E+
31458	1.16E+02	1.24E+02	1.24E+02	1.18E+02	1.23E+02	1.23E+02	1.23E+02	1.25E+
47548	1.10E+02	1.19E+02	1.07E+02	1.10E+02	1.16E+02	1.14E+02	1.14E+02	1.20E+
21100	1.34E+02	1.36E+02	1.20E+02	1.28E+02	1.32E+02	1.31E+02	1.21E+02	1.37E+
11768	1.31E+02	1.24E+02	1.32E+02	1.37E+02	1.27E+02	1.28E+02	1.22E+02	1.20E+
36969	1.11E+02	1.07E+02	9.87E+01	1.10E+02	1.12E+02	1.09E+02	1.09E+02	1.06E+
36656	1.14E+02	1.31E+02	1.22E+02	1.26E+02	1.28E+02	1.26E+02	1.17E+02	1.24E+
14915	1.42E+02	1.35E+02	1.45E+02	1.37E+02	1.41E+02	1.45E+02	1.37E+02	1.34E+
16766	1.22E+02	1.25E+02	1.21E+02	1.09E+02	1.08E+02	1.25E+02	1.28E+02	1.35E+
18197	1.47E+02	1.44E+02	1.41E+02	1.41E+02	1.48E+02	1.47E+02	1.40E+02	1.47E+
30915	1.29E+02	1.41E+02	1.33E+02	1.38E+02	1.25E+02	1.36E+02	1.31E+02	1.31E+
41090	1.48E+02	1.56E+02	1.51E+02	1.52E+02	1.44E+02	1.35E+02	1.42E+02	1.38E+
48054	1.40E+02	1.44E+02	1.39E+02	1.35E+02	1.45E+02	1.48E+02	1.41E+02	1.50E+

Project Assessment

- Pass/Fail (10% PP Module)
- Fail only if submission shows evidence of plagiarism or insufficient effort
- If you fail then you will get a chance to resubmit
- You do not need to get correct answer to pass
- Group Feedback – this lecture!
- Example solution on Blackboard
- Any questions? Email: bappelbe@ic.ac.uk

Project: Calculating Hubble's constant from redshift data

- Calculate a value for H_0 given 30 observations of galaxy redshifts

  Key Computational Challenges:

  1. Checking if instrument response was good
  2. Fitting spectral data using Gaussian + line function
  3. Matching observation numbers to get distance for each spectral data file
  4. Efficiently doing 1-3 for all spectral datasets
  5. Fitting velocity vs. distance data to get value for H_0

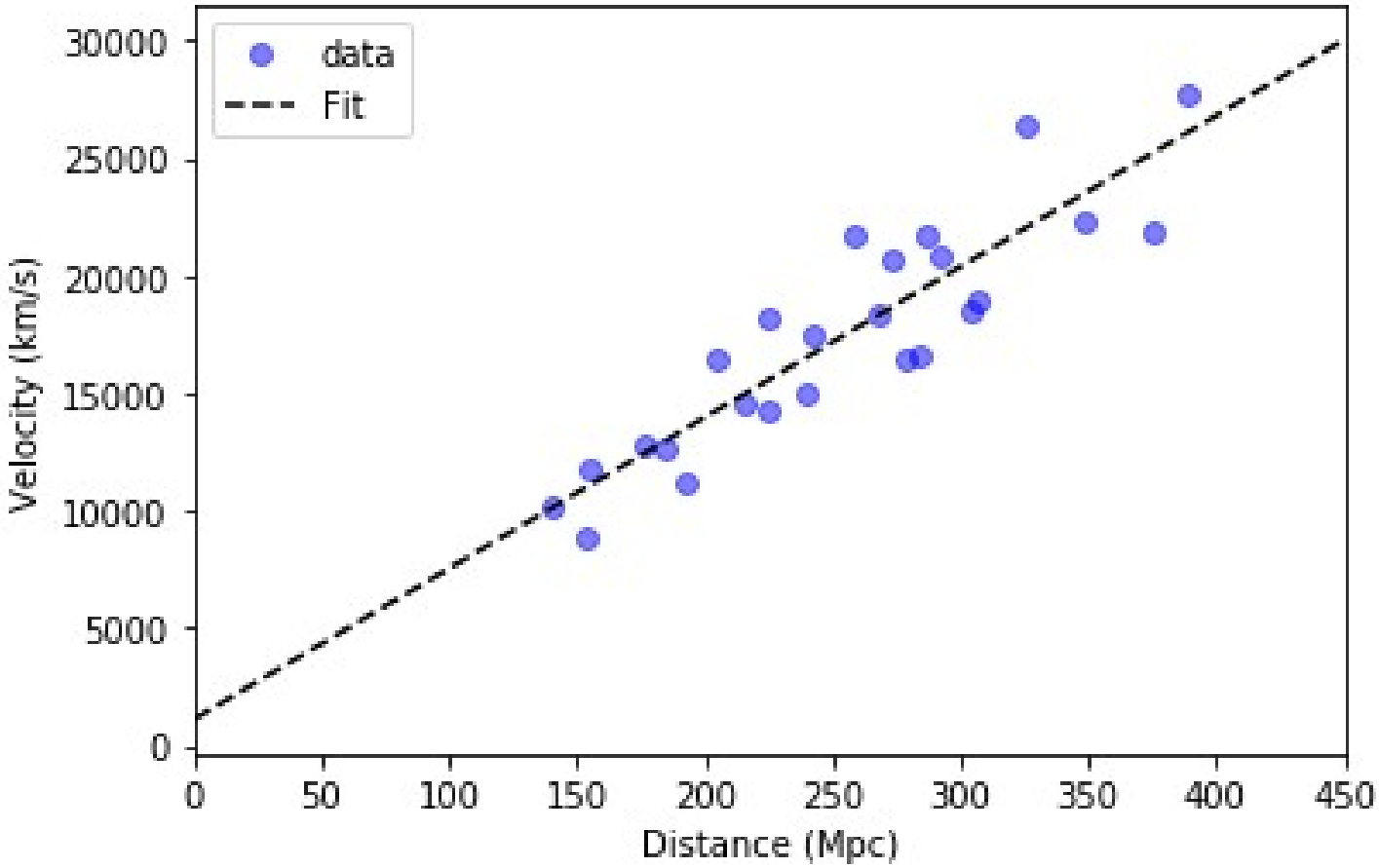
Example marking scheme (total = 10)

1. Checking if instrument response was good **(1)**
2. Fitting spectral data using Gaussian + line function **(1)**
3. Matching observation numbers to get distance for each spectral data file **(1)**
4. Efficiently doing 1-3 for all spectral datasets **(1)**
5. Fitting velocity vs. distance data to get value for H_0 **(0.5)**

- Code runs when executed without returning errors **(2)**
- Code that is well organised and commented **(2)**
- Plot of Velocities vs Distances **(1)**
- Value for H_0 **(0.5)**

This is only a guide – we will not assess projects using this marking scheme!

Results

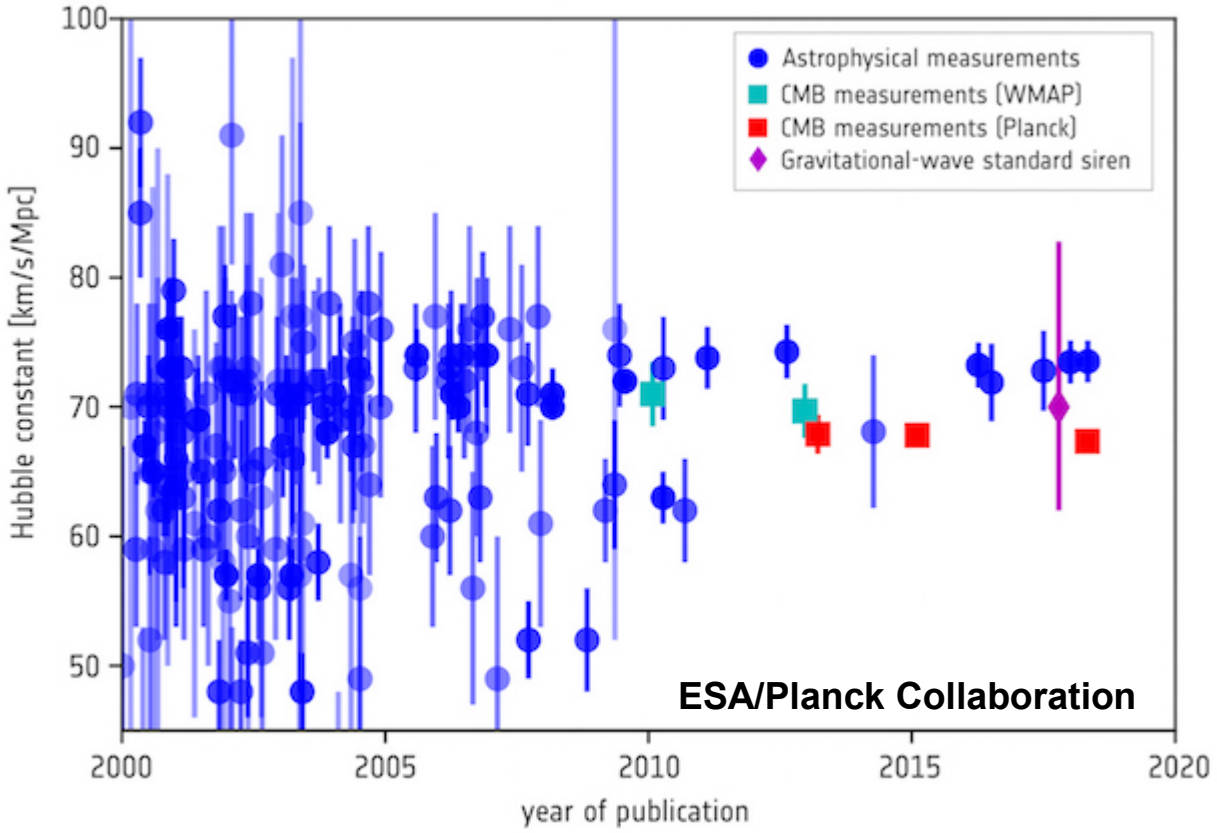


Hubble's Constant: 64.25 +/- 6.56 km/s/Mpc

25 Good Instrument Responses

5 Bad Instrument Responses

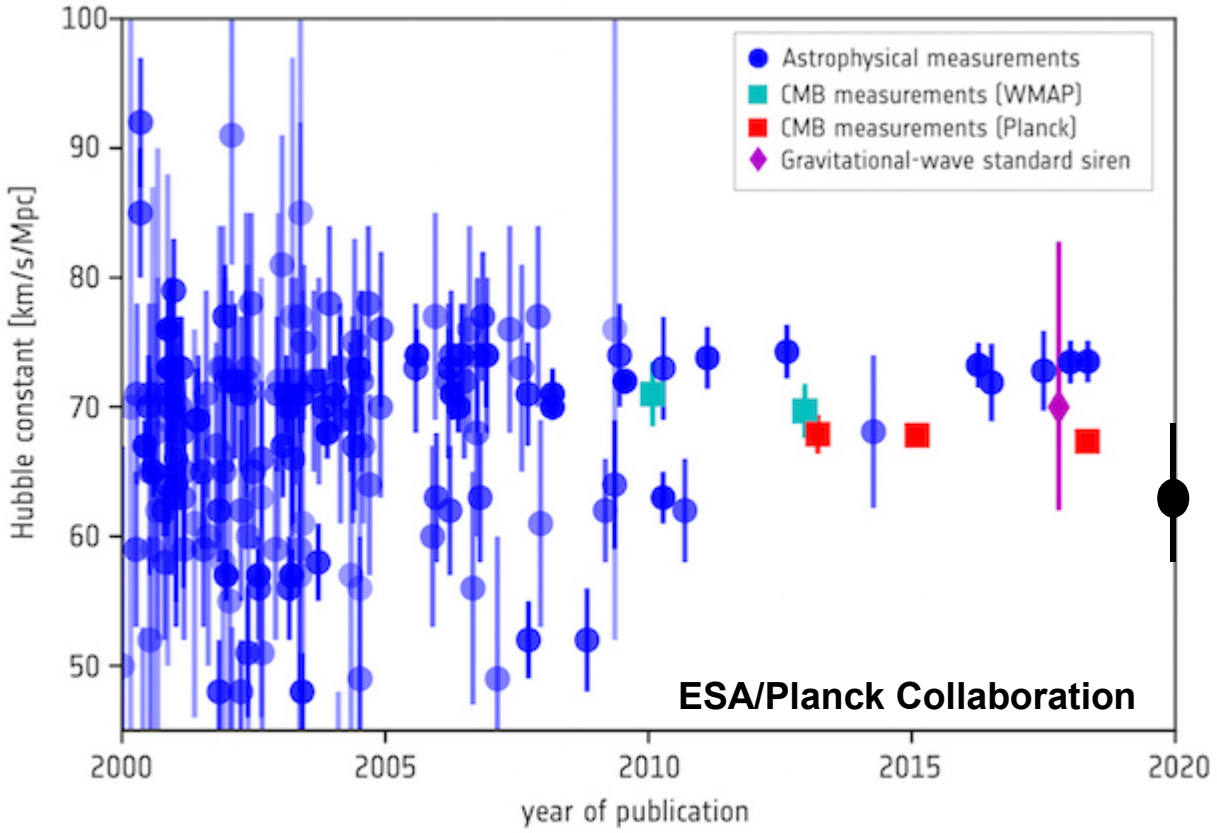
Summary of recent measurements of H_0



Astrophysical Observations = 73.5 km/s/Mpc

CMB = 67.4 km/s/Mpc

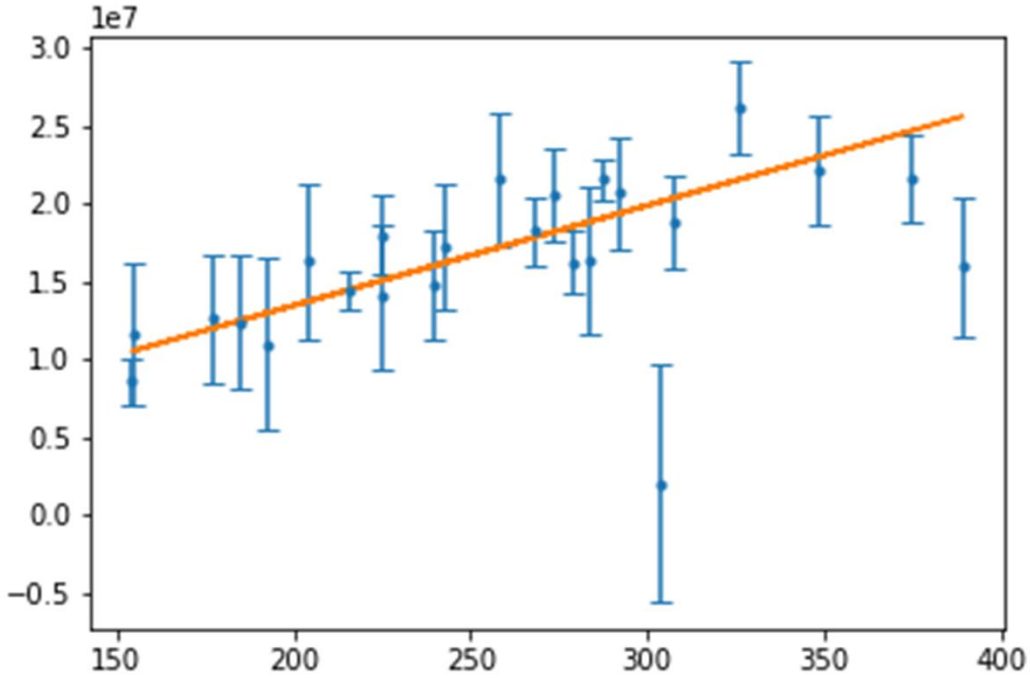
Summary of recent measurements of H_0



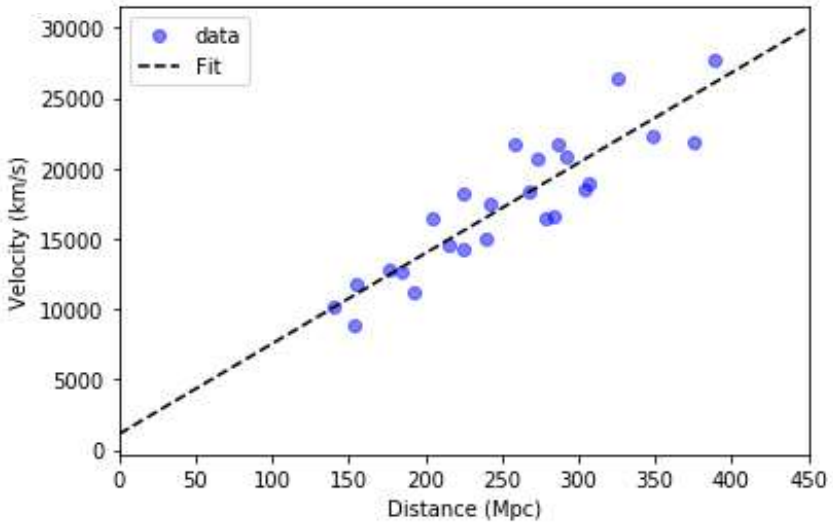
Astrophysical Observations = 73.5 km/s/Mpc

CMB = 67.4 km/s/Mpc

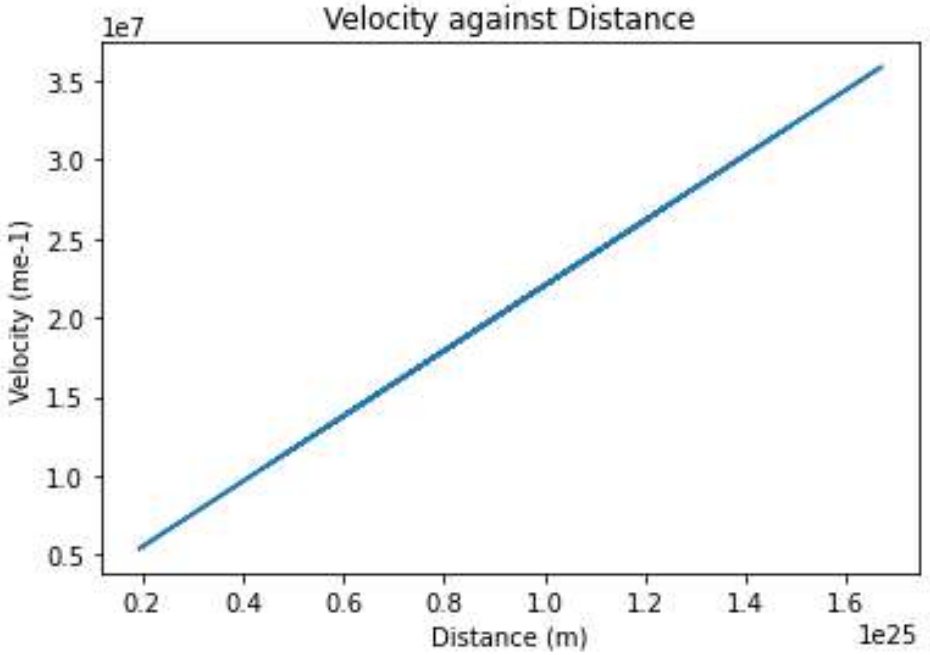
Plot of velocity vs distance



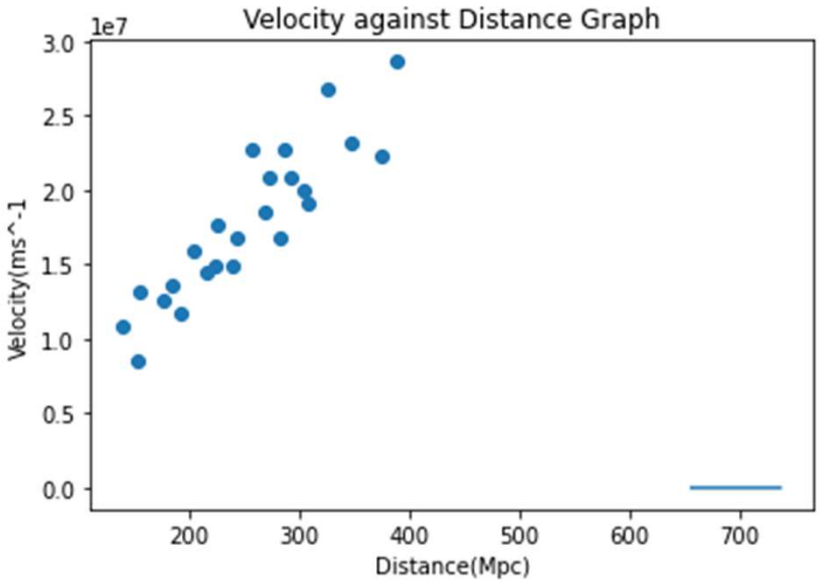
No axes labels

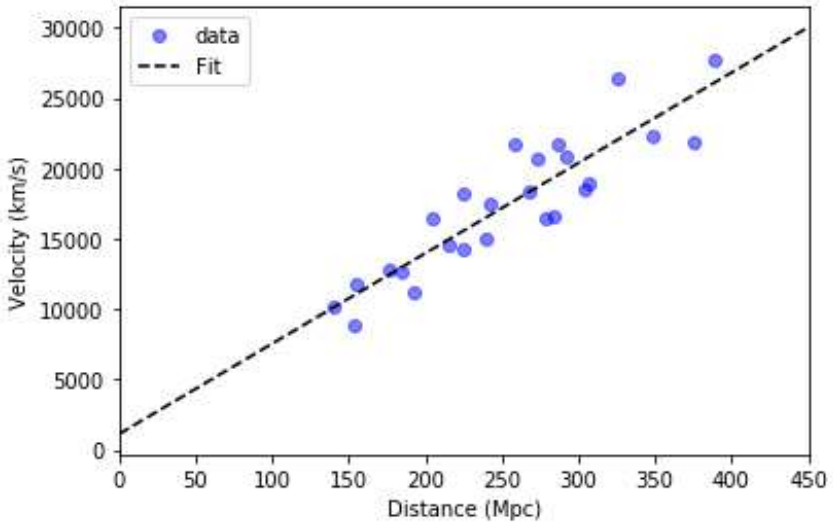


Plot of velocity vs distance

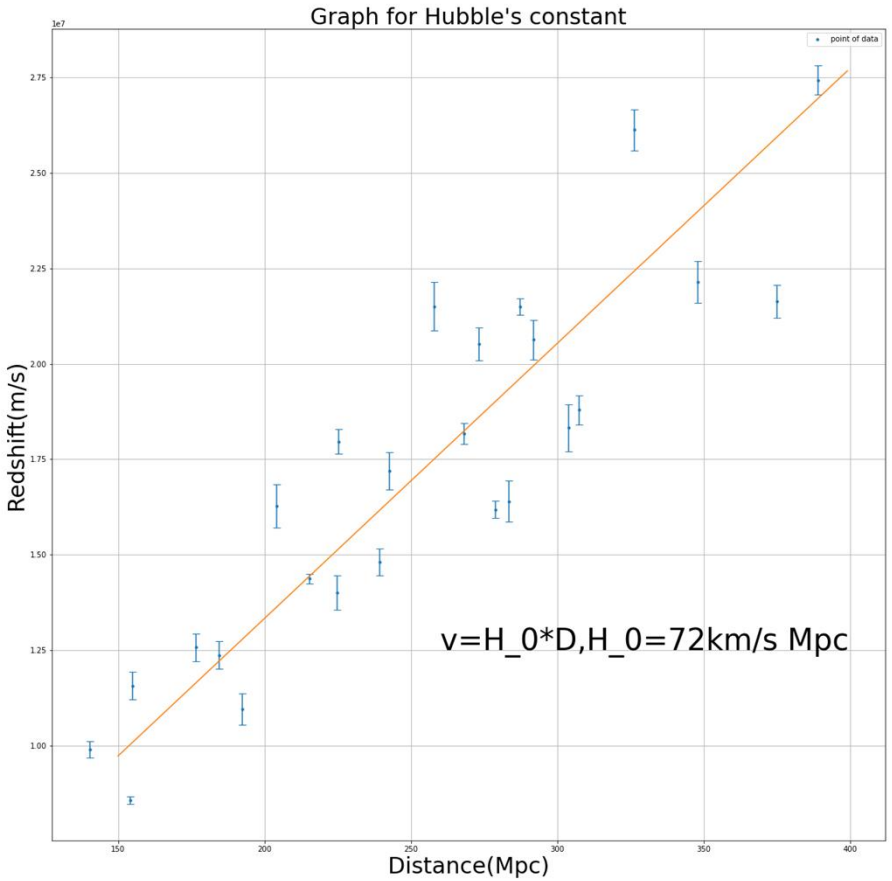


Data points and/or fit line not plotted

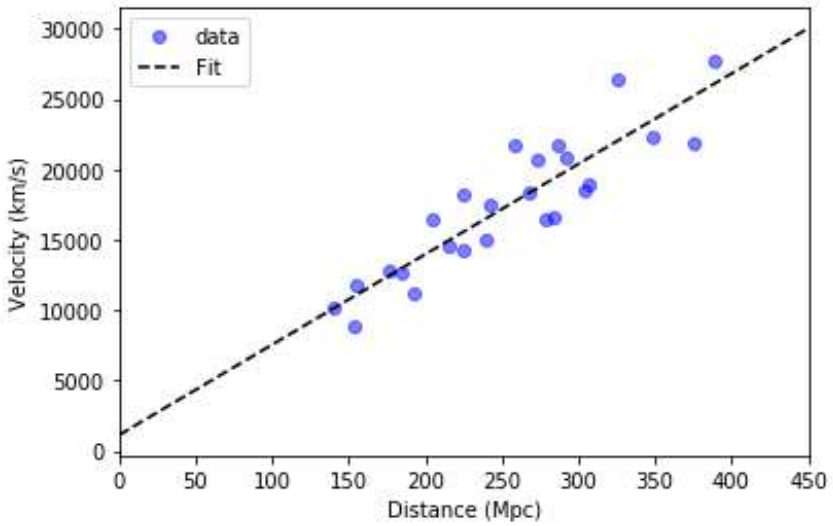




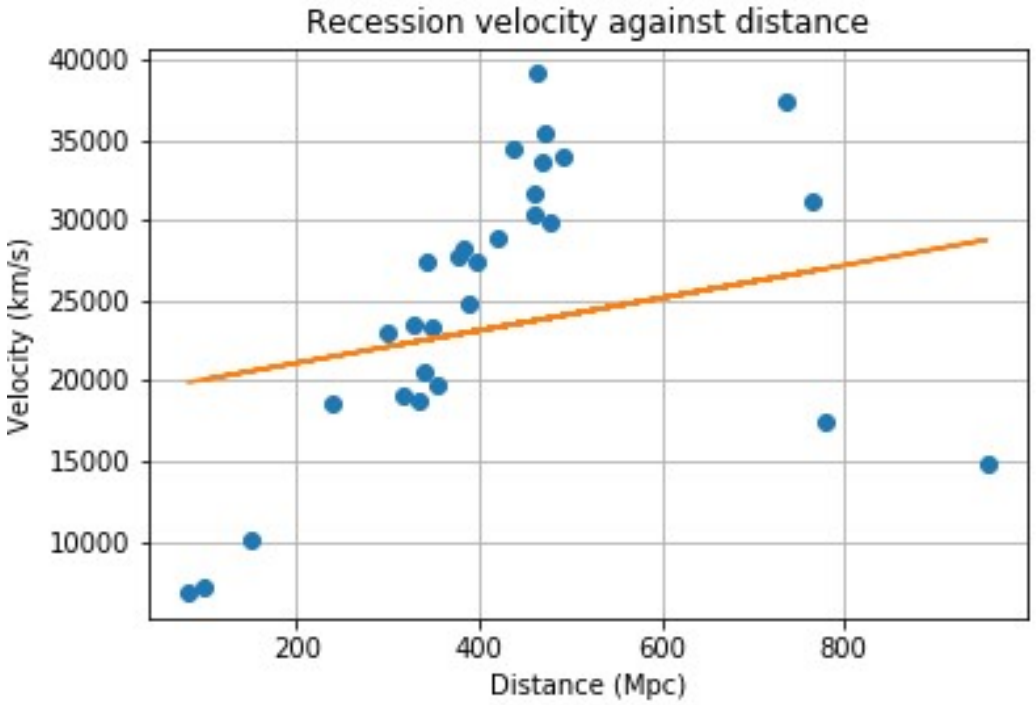
Plot of velocity vs distance



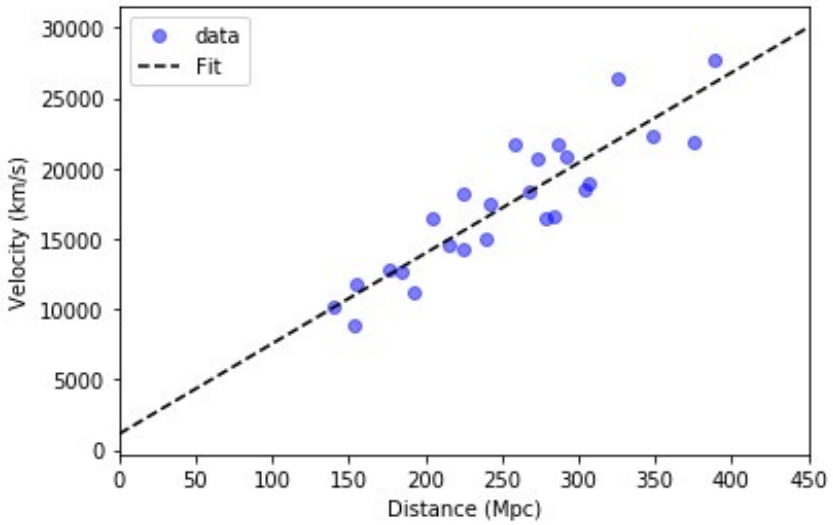
Data points & labels too small



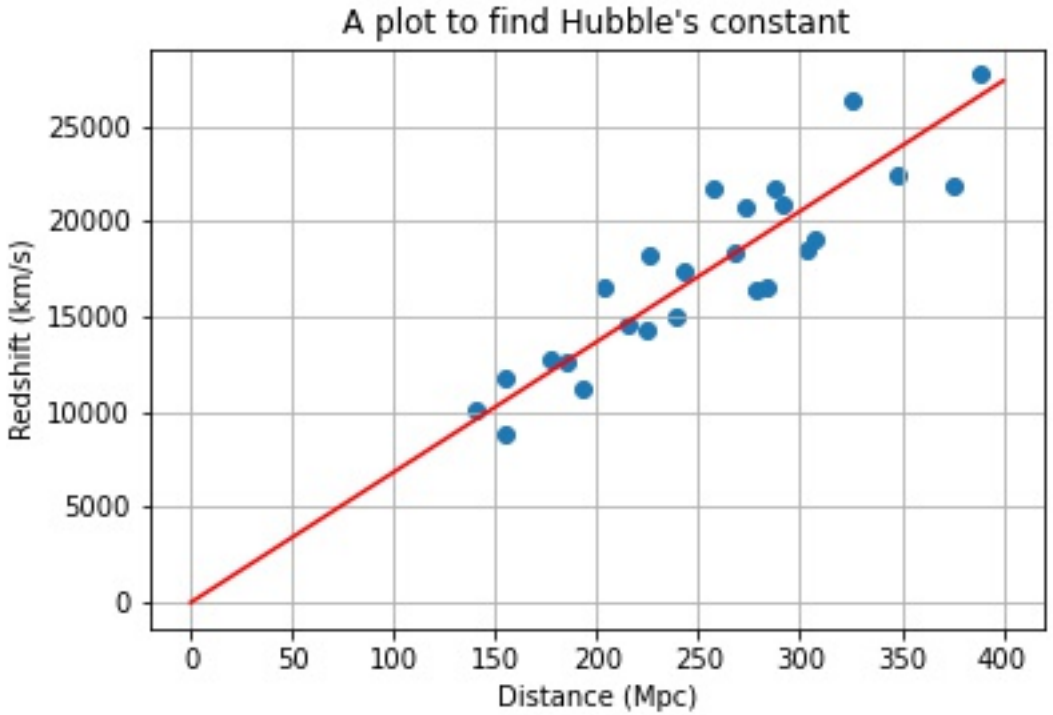
Plot of velocity vs distance



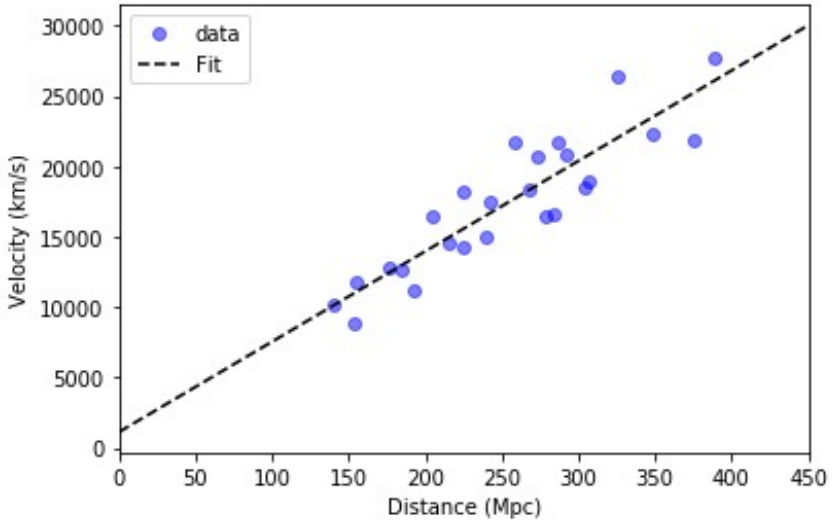
Fitted to both good & bad instrument responses



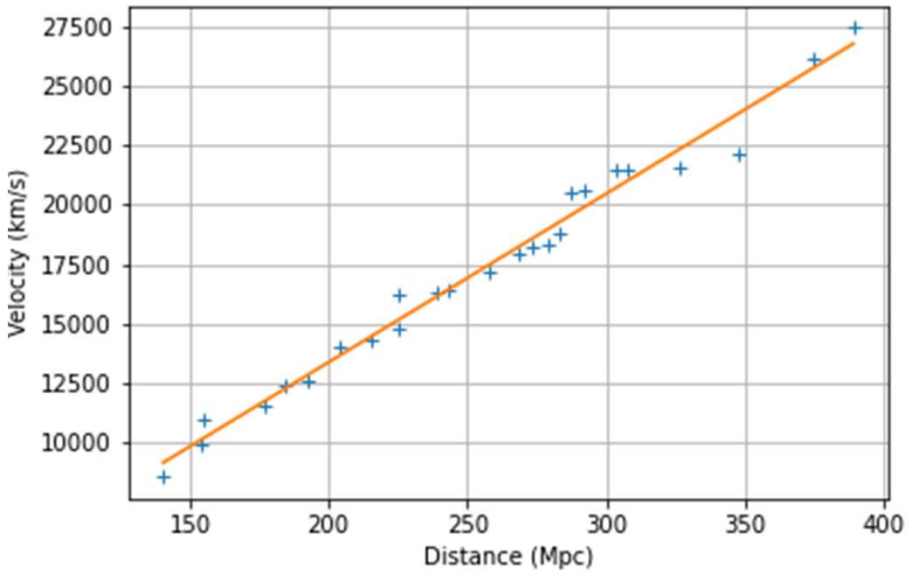
Plot of velocity vs distance



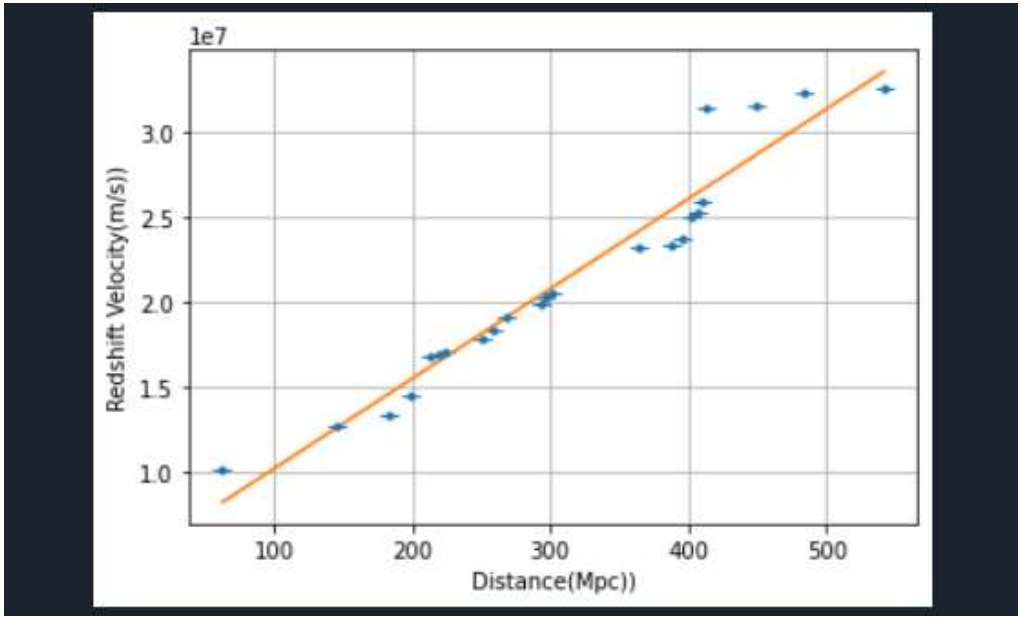
Fitted line forced to intersect origin

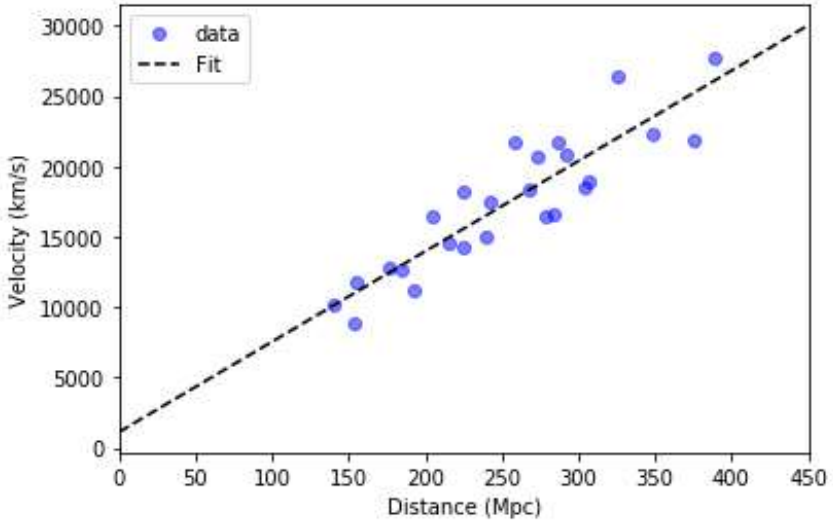


Plot of velocity vs distance

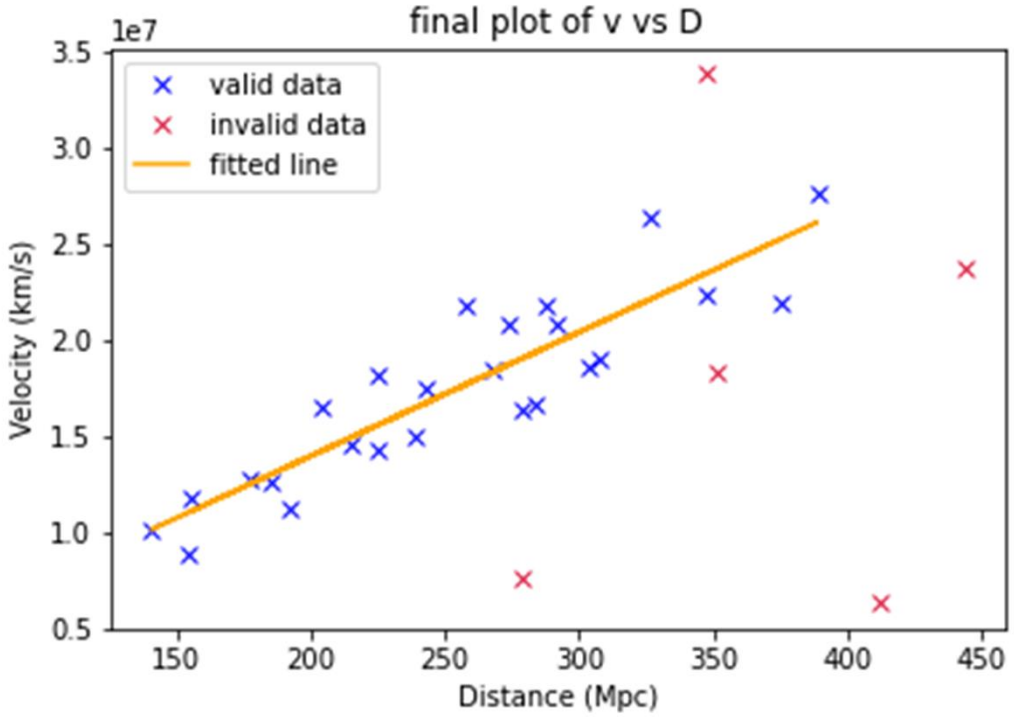


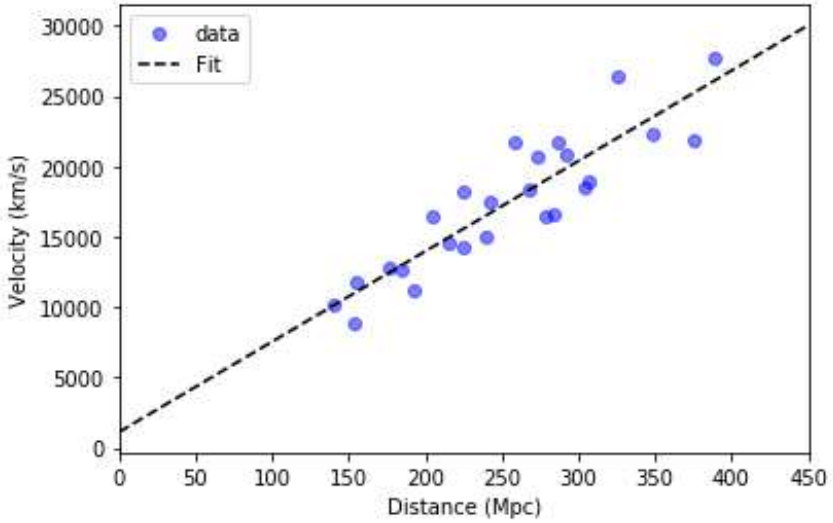
Incorrect matching of velocities & distances





Plot of velocity vs distance





Lecture 6 – Part 2

1. Project Summary & Results
2. Example Solution

Dr. Brian Appelbe

b.appelbe07@imperial.ac.uk

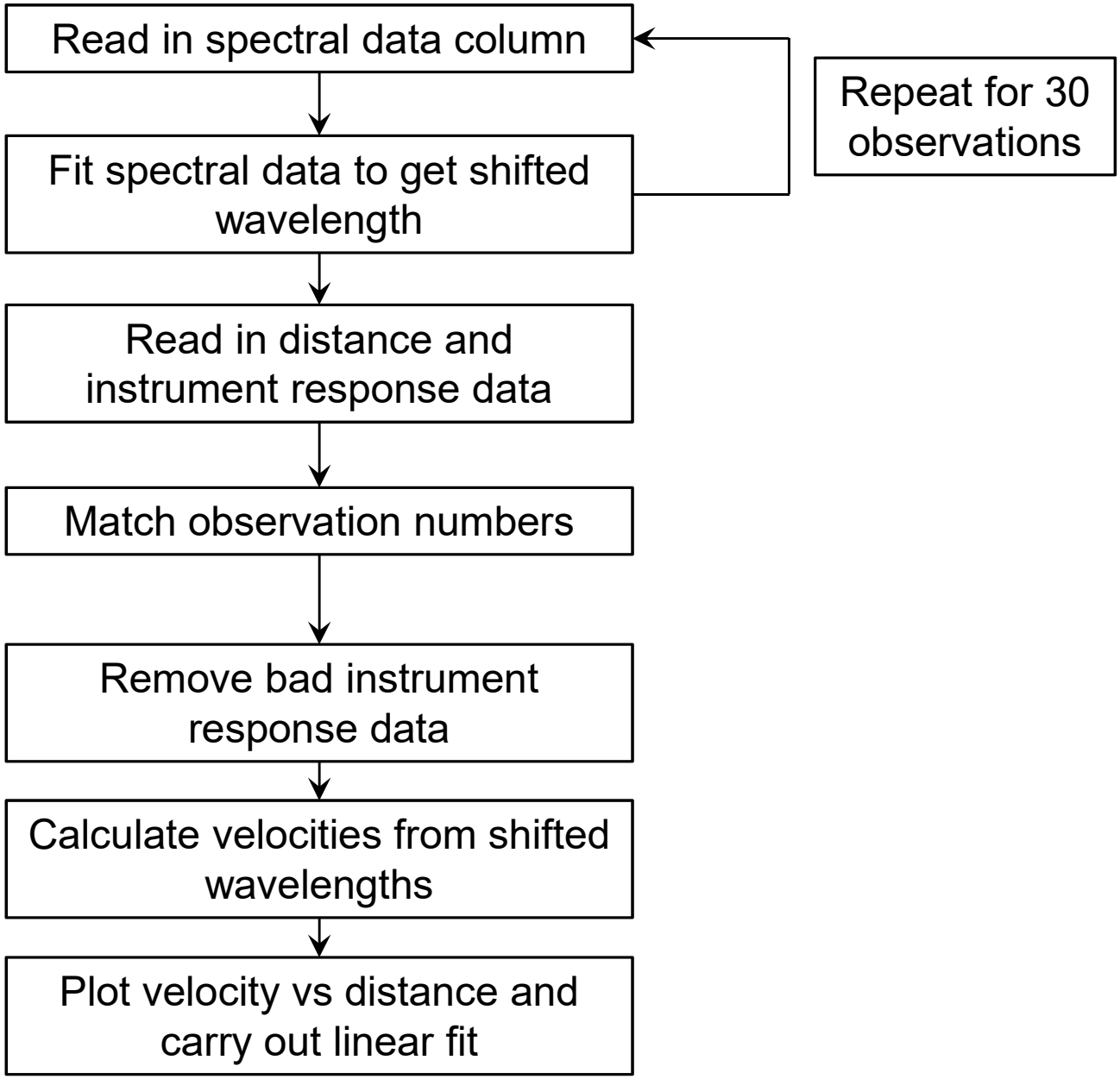
Example Project Solution

- Spectral data is measured as a function of wavelength
- Spectral data is given in columns for each observation
- Observation numbers given in header in spectral data file
- *Please note use of comments (#) in this example solution!*

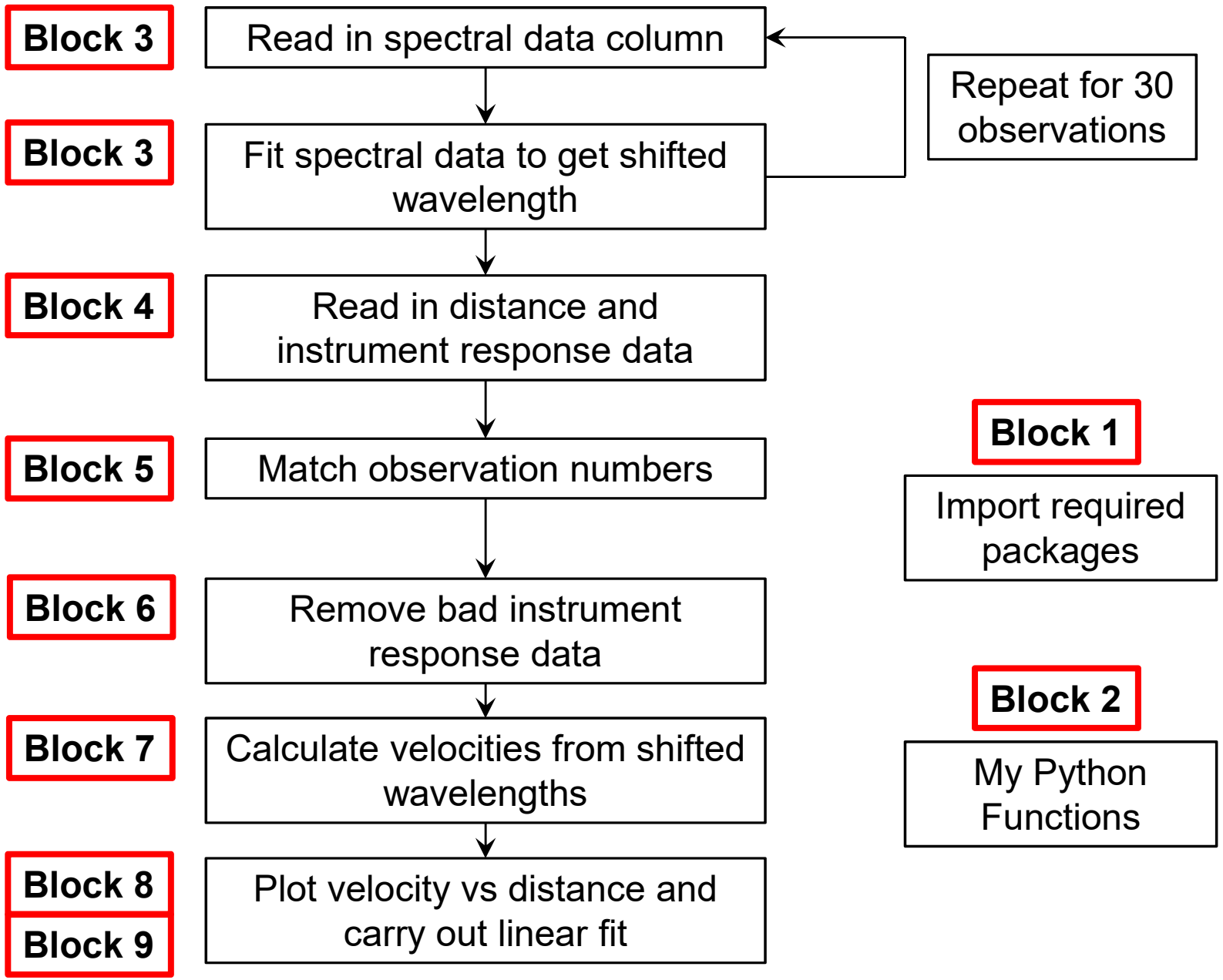
Observation	Distance(Mpc)	Valid Instrument Response
23522	517.9323713	0
31991	363.2371155	1
31544	213.5461456	1
41090	267.8456873	1
19909	295.9881898	1
43937	198.2728487	1
39451	405.9085394	1
36537	250.8782078	1
27028	502.9875908	0
23095	483.3087199	1
16076	144.7971076	1
30915	408.9550979	1
30768	412.5117835	1
47565	155.91514	1

Date: 19/07/01	Author: E.Hubble		
Wavelength (m)	Observation: 19909	Observation: 27028	Observation: 16652
6.60E-07	1.24E+02	1.48E+02	1.33E+02
6.60E-07	1.28E+02	1.45E+02	1.39E+02
6.60E-07	1.34E+02	1.48E+02	1.28E+02
6.60E-07	1.30E+02	1.48E+02	1.38E+02
6.60E-07	1.42E+02	1.45E+02	1.37E+02
6.61E-07	1.24E+02	1.53E+02	1.37E+02
6.61E-07	1.24E+02	1.43E+02	1.27E+02
6.61E-07	1.39E+02	1.48E+02	1.37E+02
6.61E-07	1.28E+02	1.49E+02	1.32E+02
6.61E-07	1.17E+02	1.46E+02	1.27E+02
6.61E-07	1.26E+02	1.45E+02	1.27E+02
6.61E-07	1.26E+02	1.36E+02	1.26E+02
6.61E-07	1.18E+02	1.34E+02	1.32E+02
6.61E-07	1.27E+02	1.45E+02	1.36E+02
6.61E-07	1.35E+02	1.42E+02	1.37E+02

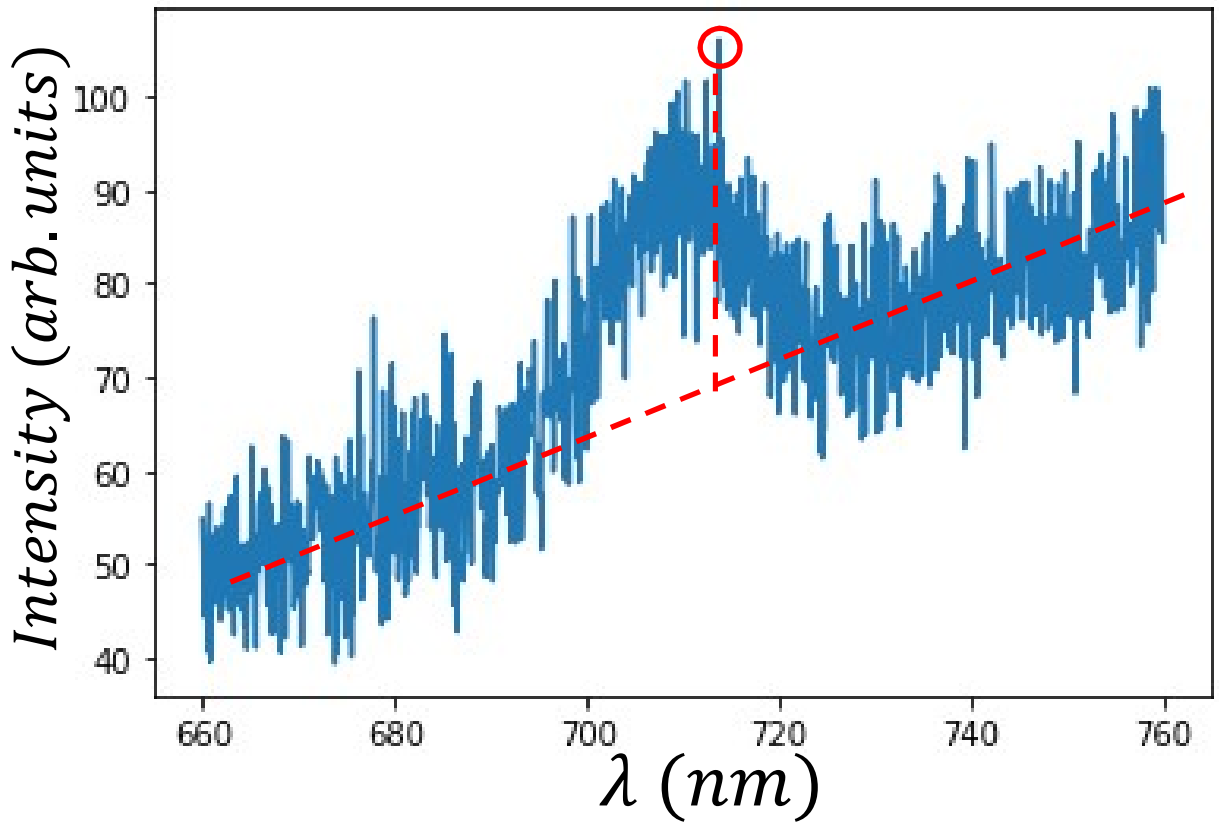
Project Flow Diagram



Project Flow Diagram

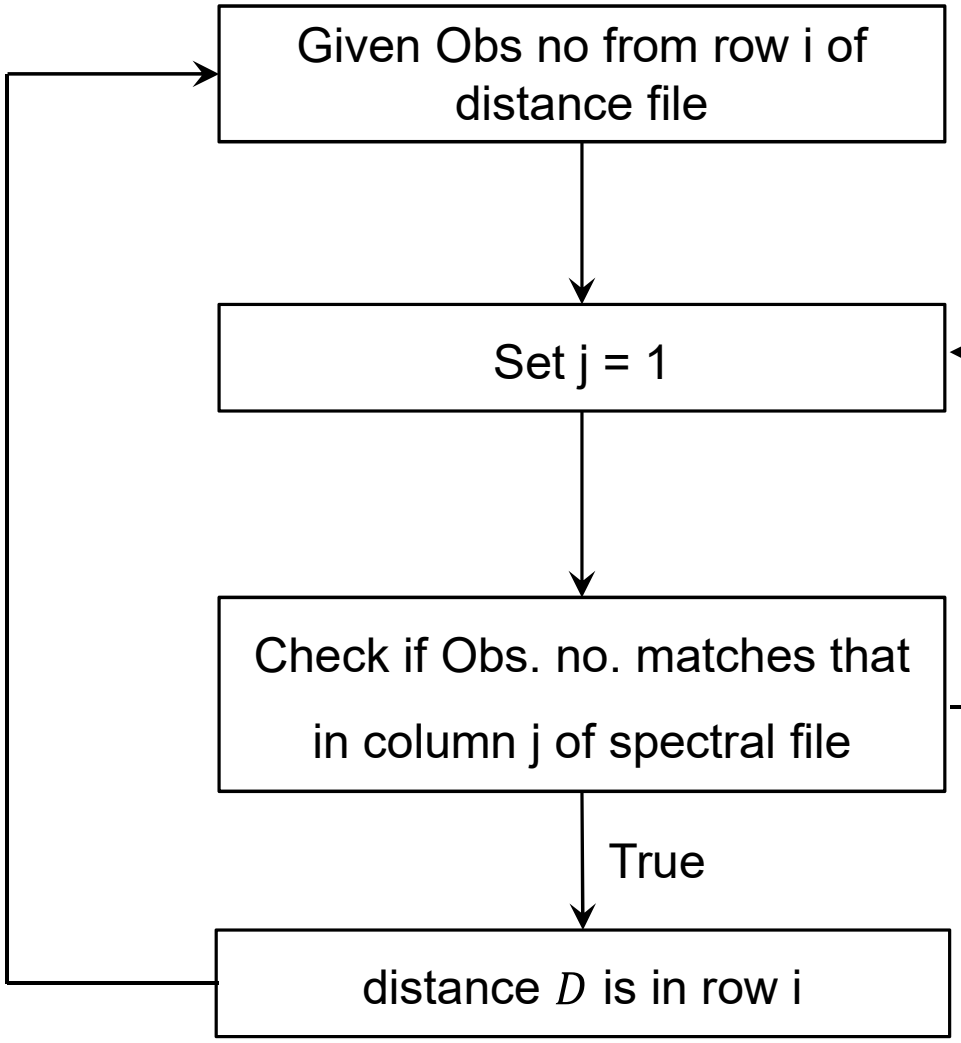


Fitting spectral data using Gaussian + line



- Estimate slope & intercept of line using points at beginning and end of data
- Estimate location of gaussian peak & gaussian amplitude from difference between data and line estimate

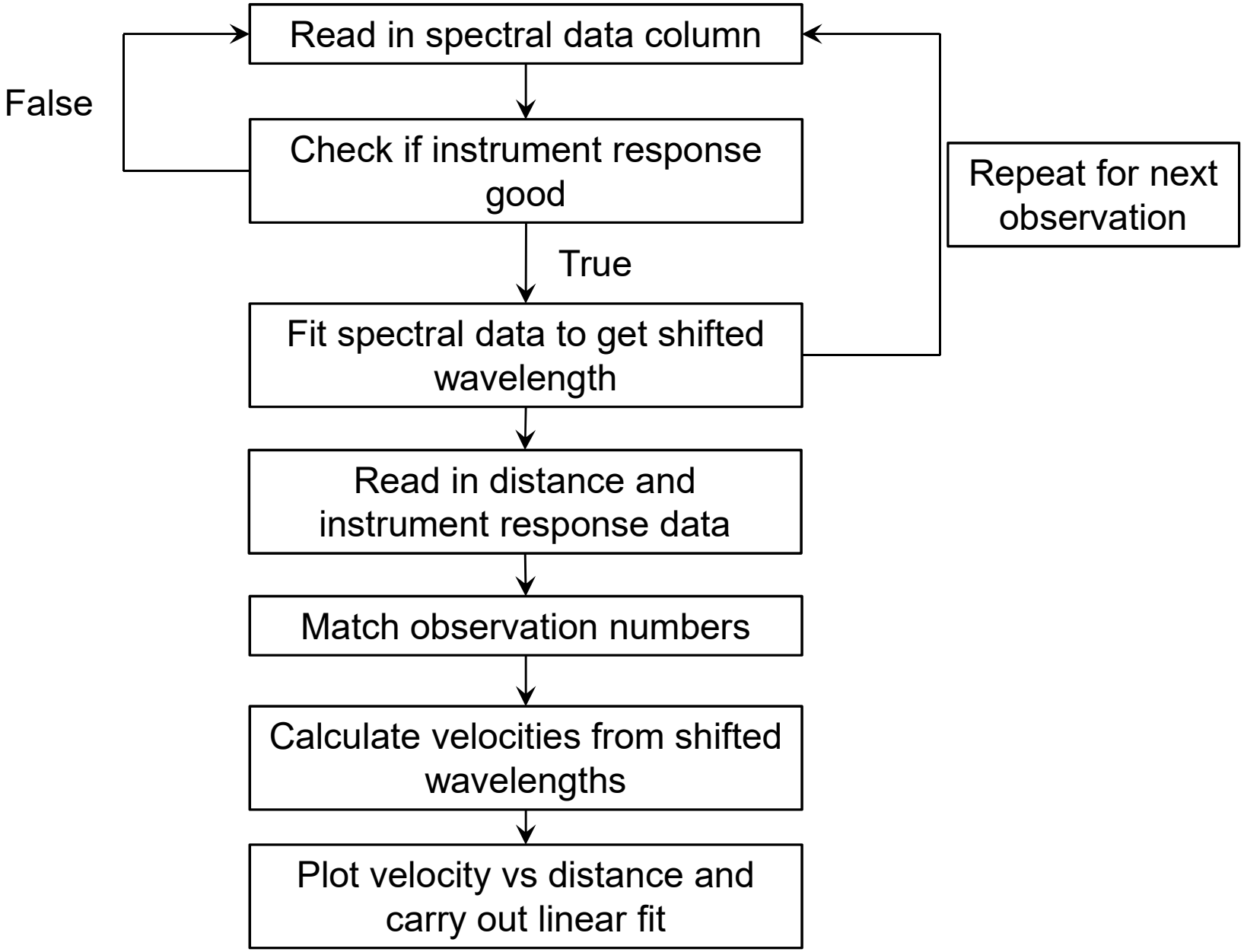
Matching observation numbers



Observation	Distance(Mpc)	Vali
23522	517.9323713	
31991	363.2371155	
31544	213.5461456	
41090	267.8456873	
19909	295.9881898	
43937	198.2728487	
39451	405.9085394	
36537	250.8782078	
27028	502.9875908	
23095	483.3087199	
16076	144.7971076	
30915	408.9550979	
30768	412.5117835	
47565	155.91514	

We can use a next FOR loop

Project Flow Diagram



A Common Program Error

```
for i in range(0,30):
    def fit_func(x,a,mu,sig,m,c):
        gaus=a*np.exp(-(x-mu)**2/(2*sig**2))
        line=m*x+c
        return gaus+line

    fit[i],cov=op.curve_fit(fit_func,mat[1:,0],mat[1:,i+1],p0=p0[i])
```

- The function is redefined every time the FOR loop iterates

Further Computing in Year 1

- Using Python in Labs & seminars
- Many Computing options for Summer term projects
- Python Physics Problems: pyproblems.github.io/book
- Python Helpdesk: Thursdays 12:00-13:00 (Dr. Yasmin Andrew)

The End

- Those of you with little previous Computing experience have made tremendous progress over the last 7 weeks – Well Done!
- The best way to keep improving your skills is practice!
- Computing becomes more enjoyable & rewarding as your skills improve.