Imperial College
London

Yonsei University Graduate Class

Energy Materials: Design, Discovery and Data

Python for Science and Engineering

Lucy Whalley

PhD student

Imperial College London /

Centre for Doctoral Training in Photovoltaics

✉ l.whalley16@ic.ac.uk

About Me

2007 – 2011:

MSci degree (physics) from University of Birmingham

2011 – 2012:

PGCE (mathematics) from Birmingham City University

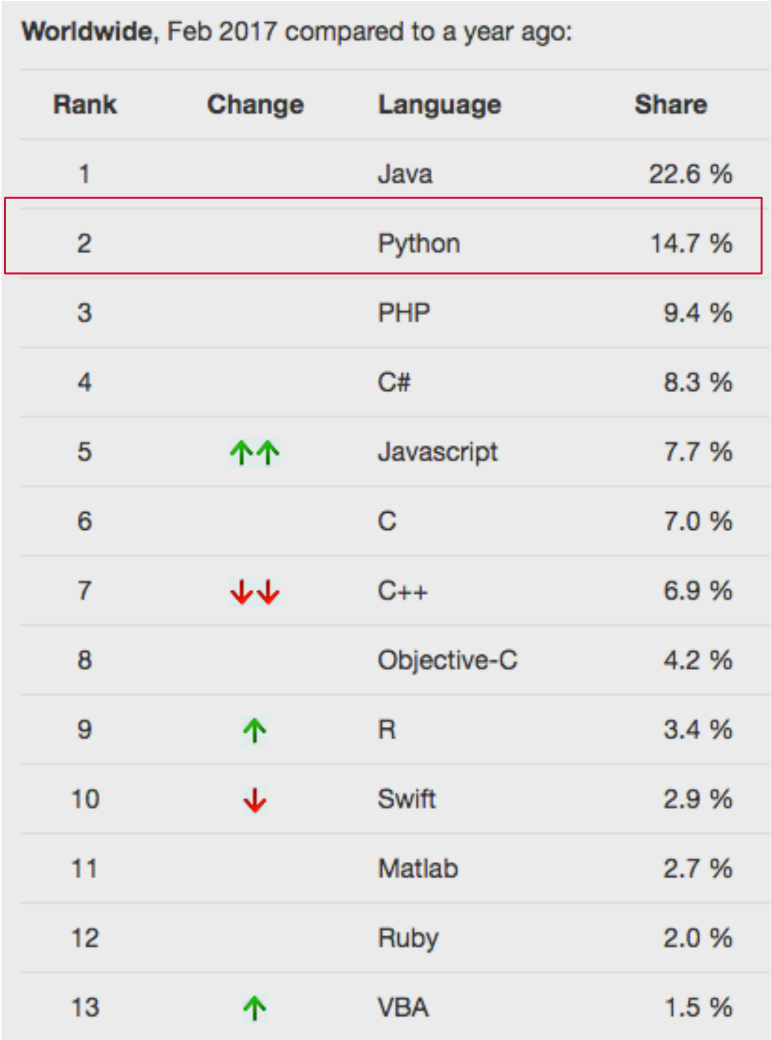
2015 – current:

PhD student at Imperial College London and CDT in Photovoltaics

Class Question

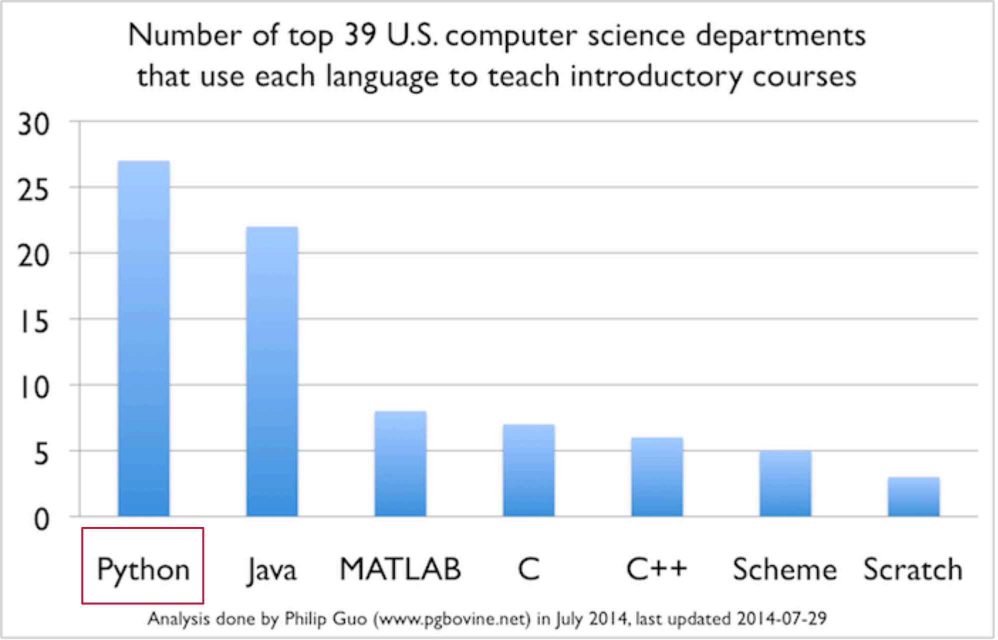
Why programming / Python?

Why Python?



Worldwide, Feb 2017 compared to a year ago:

Rank	Change	Language	Share
1		Java	22.6 %
2		Python	14.7 %
3		PHP	9.4 %
4		C#	8.3 %
5	↑↑	Javascript	7.7 %
6		C	7.0 %
7	↓↓	C++	6.9 %
8		Objective-C	4.2 %
9	↑	R	3.4 %
10	↓	Swift	2.9 %
11		Matlab	2.7 %
12		Ruby	2.0 %
13	↑	VBA	1.5 %

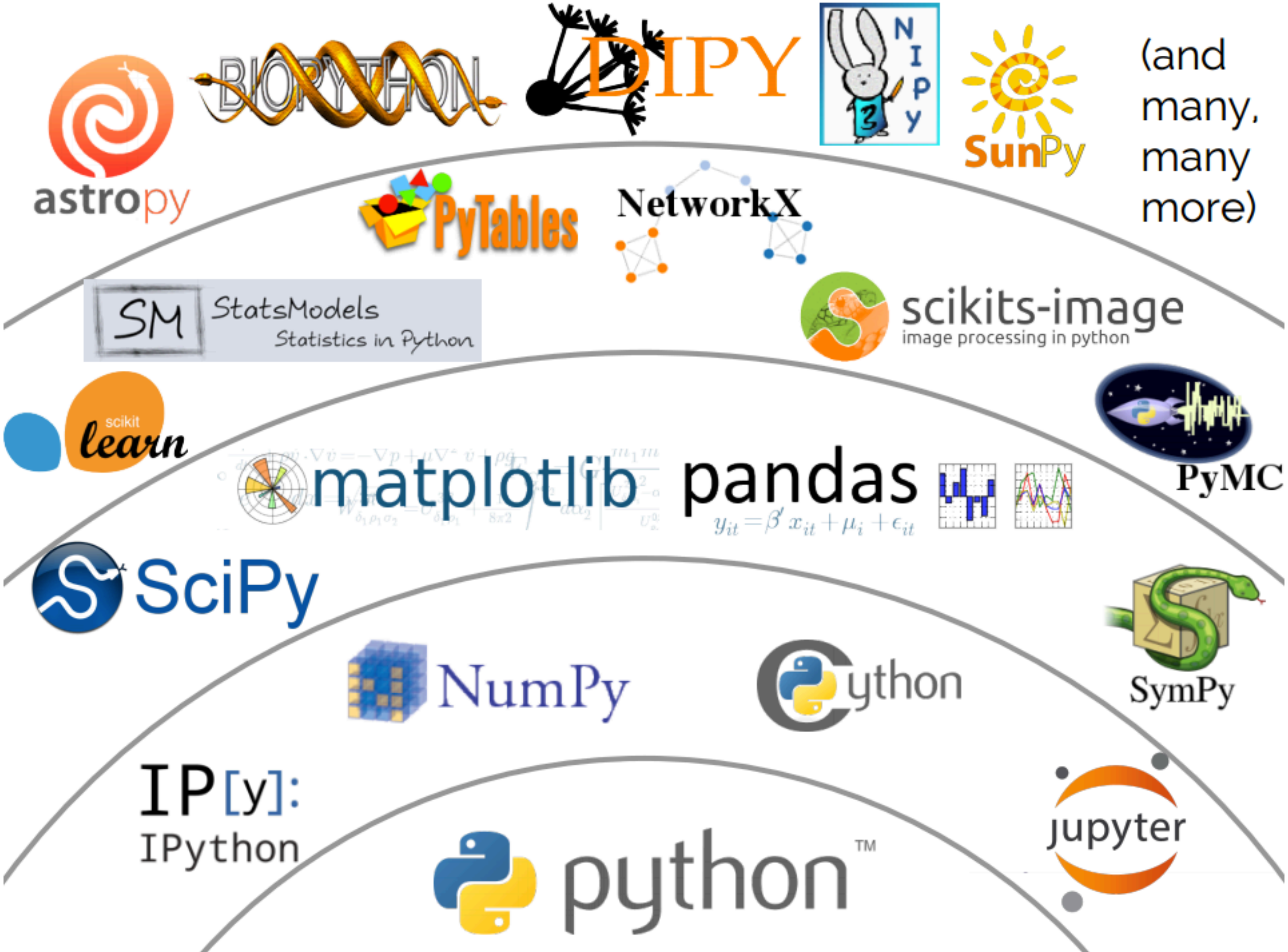


Number of top 39 U.S. computer science departments that use each language to teach introductory courses

Analysis done by Philip Guo (www.pgbovine.net) in July 2014, last updated 2014-07-29

Why Python?

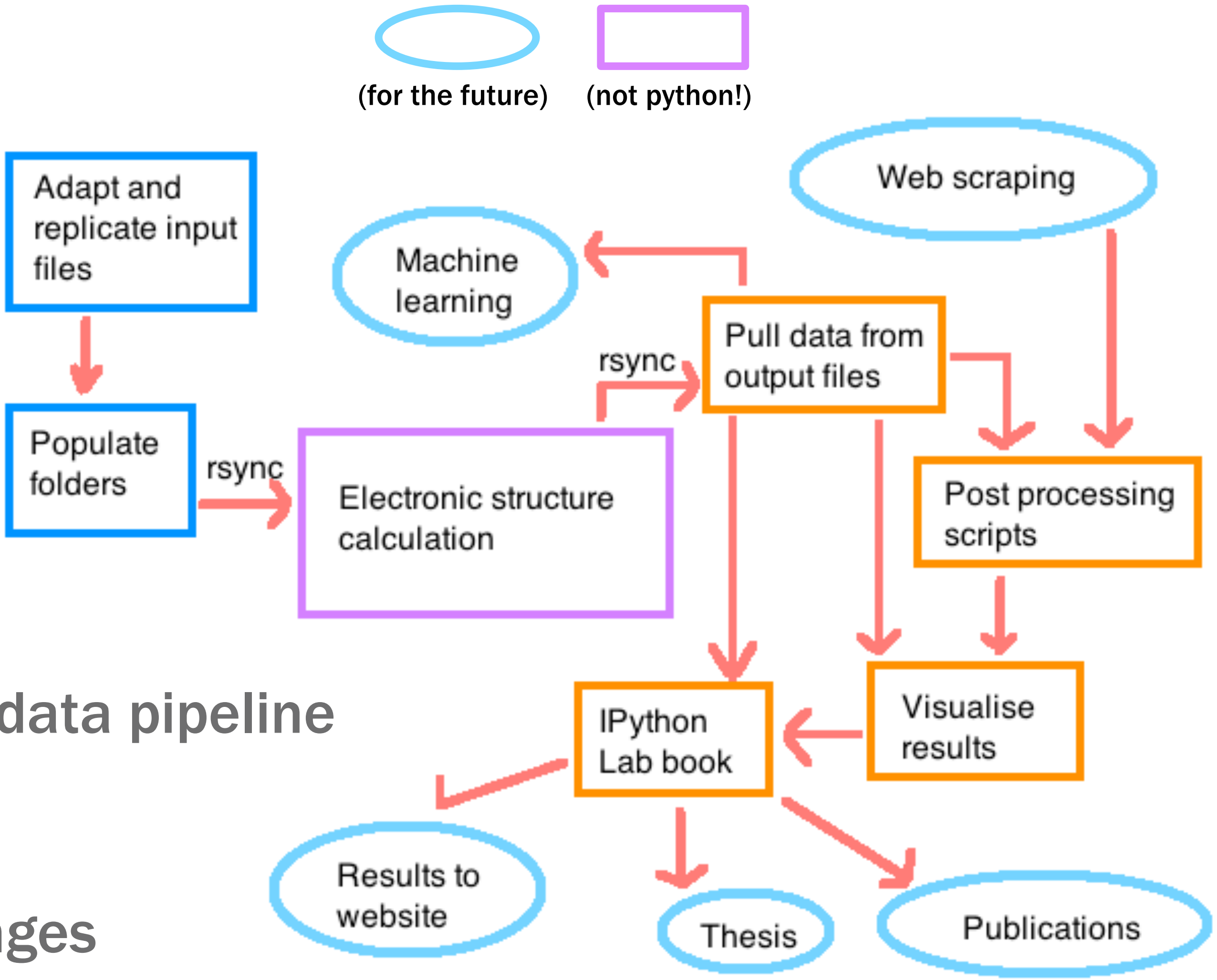
open source



(and
many,
many
more)

Imperial College London

What do I use Python for?



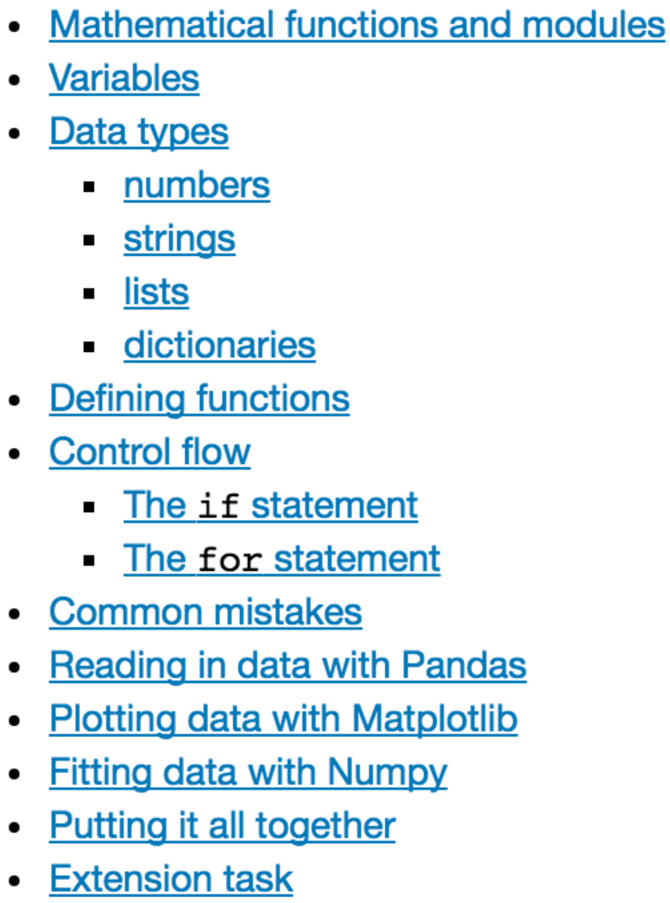
- Clear, transparent data pipeline
- Reduces errors
- Re-usable code
- Easy to make changes

Imperial College
London

Workshop Outline

Part one

Part two

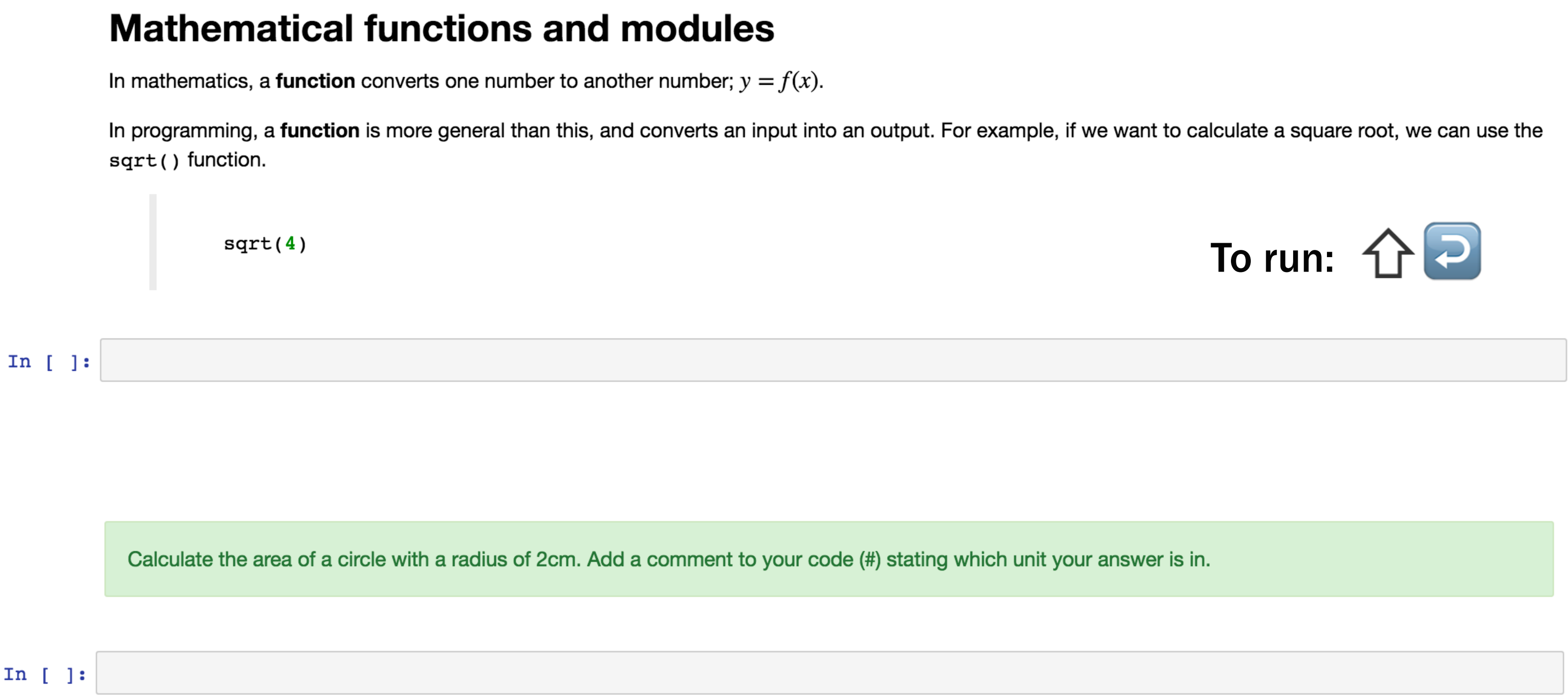


- Mathematical functions and modules
- Variables
- Data types
  - numbers
  - strings
  - lists
  - dictionaries
- Defining functions
- Control flow
  - The `if` statement
  - The `for` statement
- Common mistakes
- Reading in data with Pandas
- Plotting data with Matplotlib
- Fitting data with Numpy
- Putting it all together
- Extension task

Imperial College London

Module 4

https://github.com/WMD-group/yonsei17



Mathematical functions and modules

In mathematics, a **function** converts one number to another number; $y = f(x)$.

In programming, a **function** is more general than this, and converts an input into an output. For example, if we want to calculate a square root, we can use the `sqrt()` function.

```
sqrt(4)
```

To run: ⇧ ↩

```
In [ ]:
```

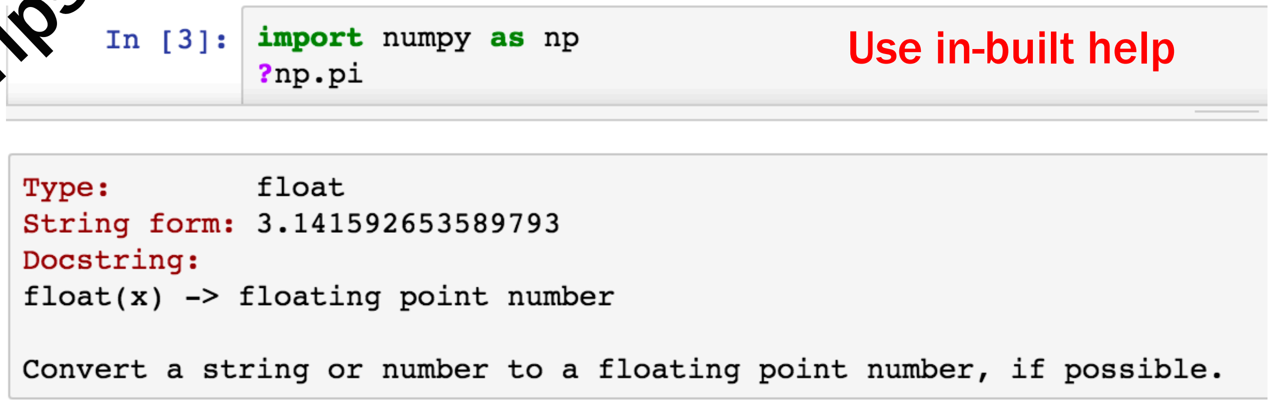
Calculate the area of a circle with a radius of 2cm. Add a comment to your code (#) stating which unit your answer is in.

```
In [ ]:
```

Module 4

Top Tips!!

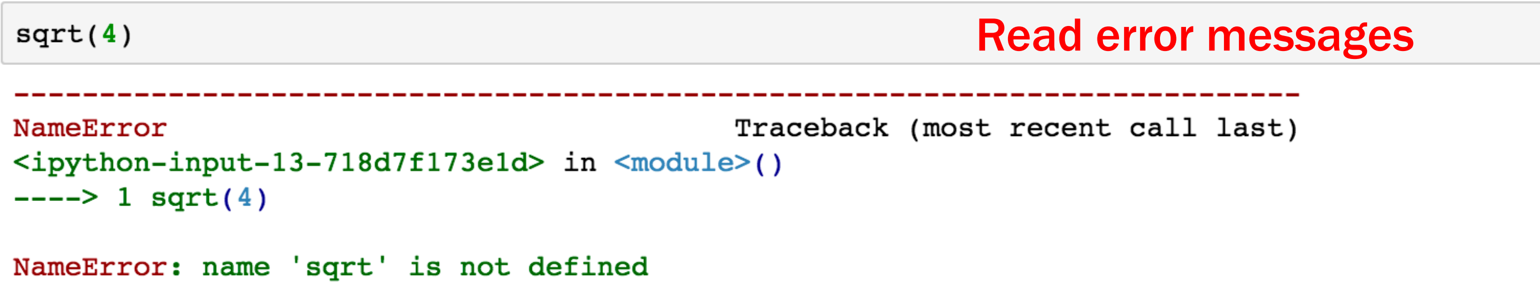
Use in-built help



```
In [3]: import numpy as np
        ?np.pi
```

```
Type:        float
String form: 3.141592653589793
Docstring:
float(x) -> floating point number

Convert a string or number to a floating point number, if possible.
```

Read error messages



```
sqrt(4)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-13-718d7f173e1d> in <module>()
----> 1 sqrt(4)

NameError: name 'sqrt' is not defined
```

It's Time For A Break

Workshop Outline

Part one

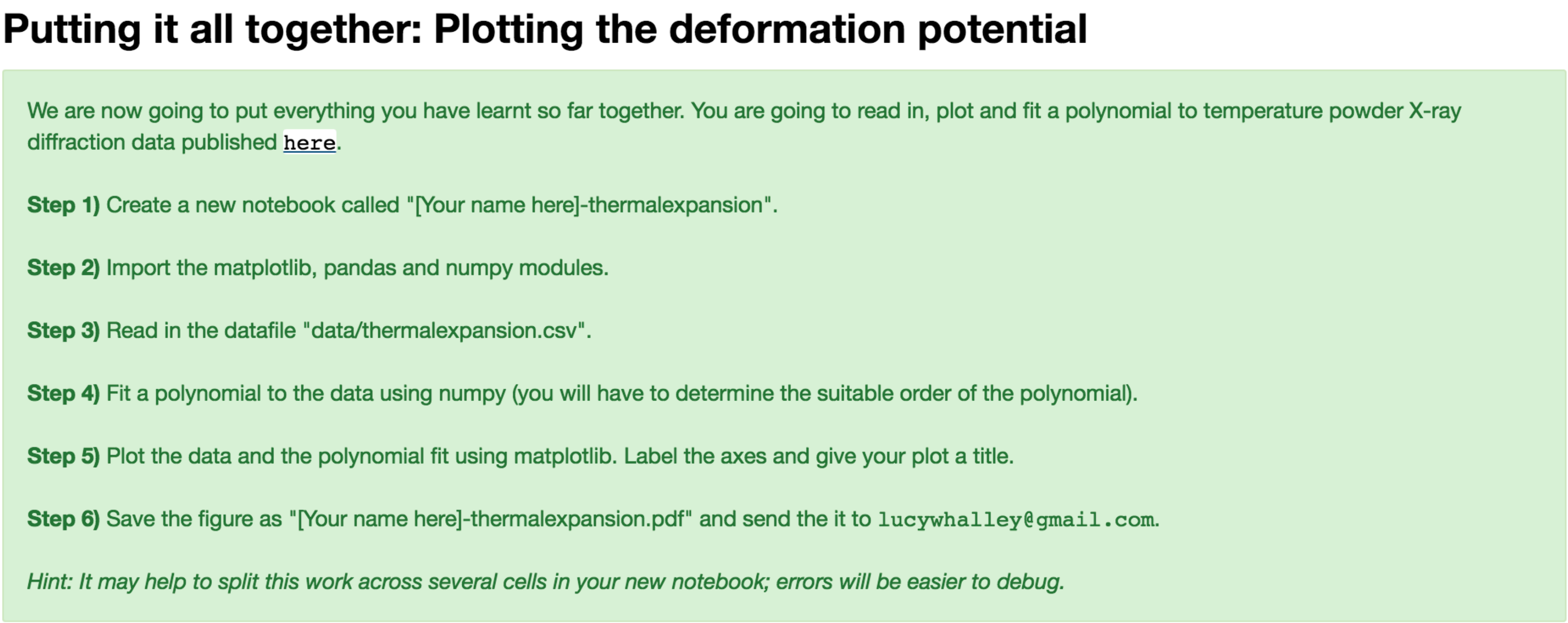
- Mathematical functions and modules
- Variables
- Data types
  - numbers
  - strings
  - lists
  - dictionaries
- Defining functions
- Control flow
  - The `if` statement
  - The `for` statement
- Common mistakes

Part two

- Reading in data with Pandas
- Plotting data with Matplotlib
- Fitting data with Numpy
- Putting it all together

- Extension task

Module 4



Putting it all together: Plotting the deformation potential

We are now going to put everything you have learnt so far together. You are going to read in, plot and fit a polynomial to temperature powder X-ray diffraction data published `here`.

Step 1) Create a new notebook called "[Your name here]-thermalexpansion".

Step 2) Import the matplotlib, pandas and numpy modules.

Step 3) Read in the datafile "data/thermalexpansion.csv".

Step 4) Fit a polynomial to the data using numpy (you will have to determine the suitable order of the polynomial).

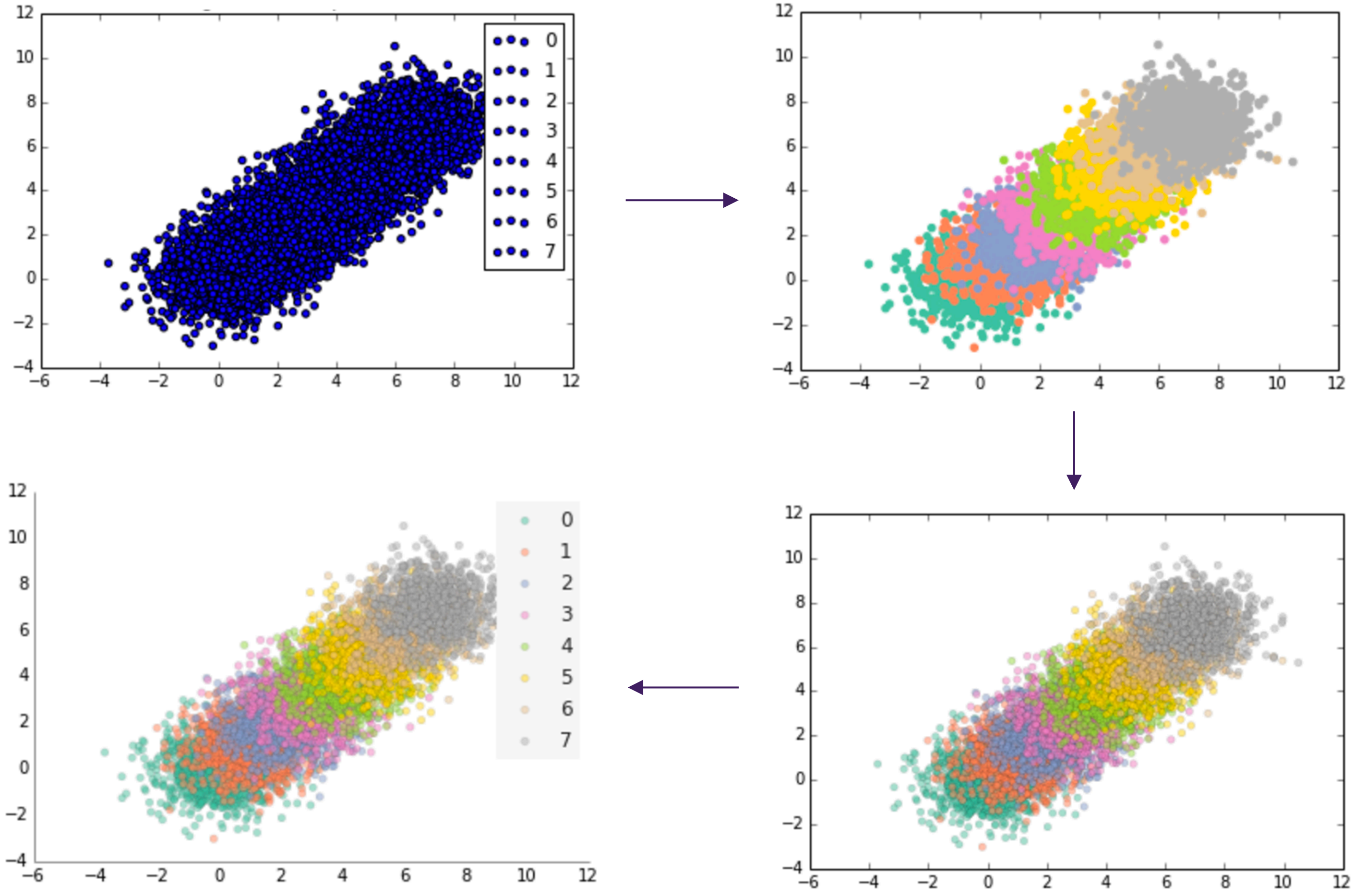
Step 5) Plot the data and the polynomial fit using matplotlib. Label the axes and give your plot a title.

Step 6) Save the figure as "[Your name here]-thermalexpansion.pdf" and send the it to `lucywhalley@gmail.com`.

Hint: It may help to split this work across several cells in your new notebook; errors will be easier to debug.

https://dx.doi.org/10.1039/C3TA10518K

Python Plots can be Beautiful



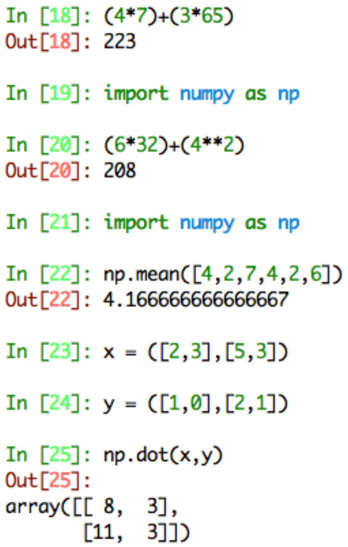
From:

http://blog.olgabotvinnik.com/blog/2013/08/21/2013-08-21-prettyplotlib-painlessly-create-beautiful-matplotlib/

Imperial College London

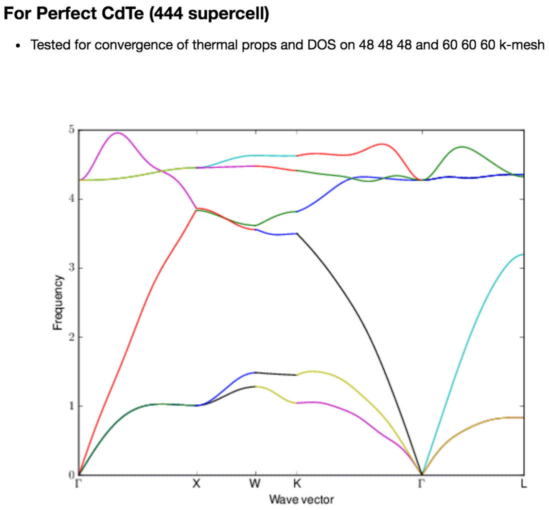
Next Steps

Use the terminal as a calculator



```
In [18]: (4*7)+(3*65)
Out[18]: 223

In [19]: import numpy as np

In [20]: (6*32)+(4**2)
Out[20]: 208

In [21]: import numpy as np

In [22]: np.mean([4,2,7,4,2,6])
Out[22]: 4.166666666666667

In [23]: x = ([2,3],[5,3])

In [24]: y = ([1,0],[2,1])

In [25]: np.dot(x,y)
Out[25]:
array([[ 8,  3],
       [11,  3]])
```

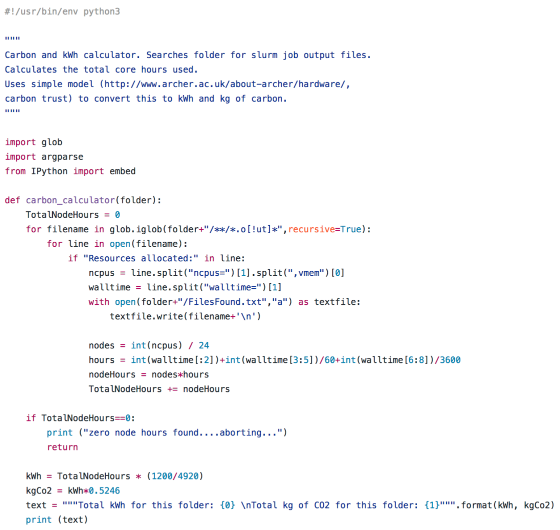
Keep a Jupyter lab book



For Perfect CdTe (444 supercell)

- Tested for convergence of thermal props and DOS on 48 48 48 and 60 60 60 k-mesh

Start a (mini?) project



```
#!/usr/bin/env python3

"""
Carbon and kWh calculator. Searches folder for slurm job output files.
Calculates the total core hours used.
Uses simple model (http://www.archer.ac.uk/about-archer/hardware/,
carbon trust) to convert this to kWh and kg of carbon.
"""

import glob
import argparse
from IPython import embed

def carbon_calculator(folder):
    TotalNodeHours = 0
    for filename in glob.iglob(folder+"/**/*.o[!ut]*",recursive=True):
        for line in open(filename):
            if "Resources allocated:" in line:
                ncpus = line.split("ncpus=")[1].split(",vmem")[0]
                walltime = line.split("walltime=")[1]
                with open(folder+"/FilesFound.txt","a") as textfile:
                    textfile.write(filename+'\n')

                nodes = int(ncpus) / 24
                hours = int(walltime[:2])+int(walltime[3:5])/60+int(walltime[6:8])/3600
                nodeHours = nodes*hours
                TotalNodeHours += nodeHours

    if TotalNodeHours==0:
        print ("zero node hours found....aborting...")
        return

    kWh = TotalNodeHours * (1200/4920)
    kgCo2 = kWh*0.5246
    text = """Total kWh for this folder: {0} \nTotal kg of CO2 for this folder: {1}""".format(kWh, kgCo2)
    print (text)
```

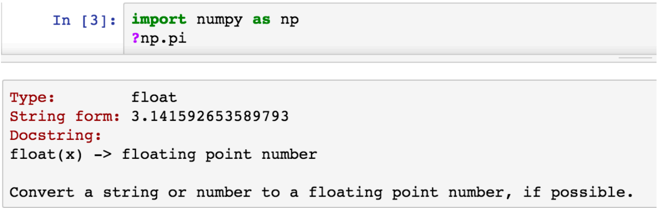
Tutorials

codecademy

Search online

stackoverflow

Use built-in help

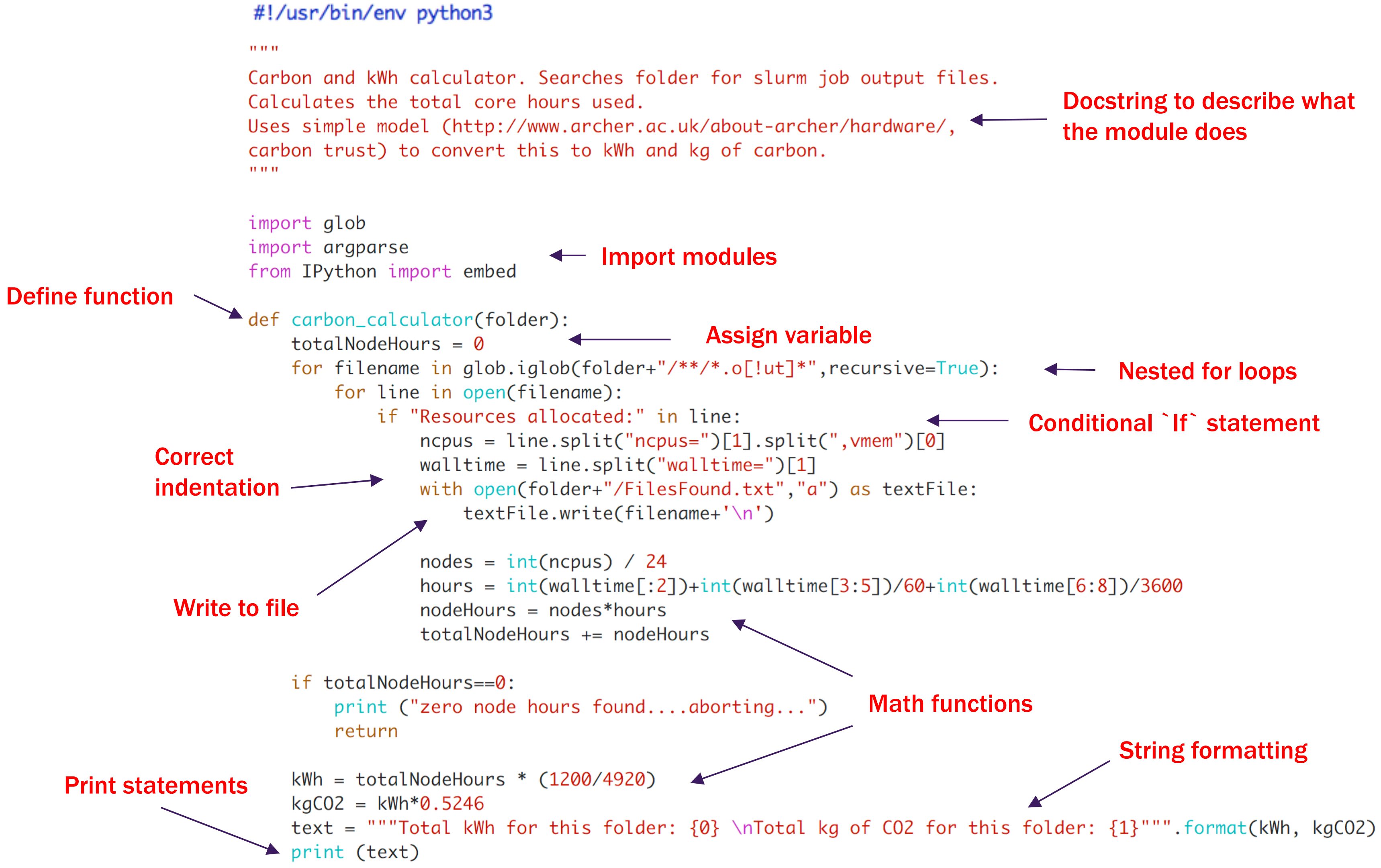


```
In [3]: import numpy as np
        ?np.pi
```

```
Type:        float
String form: 3.141592653589793
Docstring:
float(x) -> floating point number

Convert a string or number to a floating point number, if possible.
```

Imperial College London

Mini Project: carbon_calculator.py



```
#!/usr/bin/env python3

"""
Carbon and kWh calculator. Searches folder for slurm job output files.
Calculates the total core hours used.
Uses simple model (http://www.archer.ac.uk/about-archer/hardware/,
carbon trust) to convert this to kWh and kg of carbon.
"""

import glob
import argparse
from IPython import embed

def carbon_calculator(folder):
    totalNodeHours = 0
    for filename in glob.iglob(folder+"/**/*.o[!ut]*",recursive=True):
        for line in open(filename):
            if "Resources allocated:" in line:
                ncpus = line.split("ncpus=")[1].split(",vmem")[0]
                walltime = line.split("walltime=")[1]
                with open(folder+"/FilesFound.txt","a") as textFile:
                    textFile.write(filename+'\n')

                nodes = int(ncpus) / 24
                hours = int(walltime[:2])+int(walltime[3:5])/60+int(walltime[6:8])/3600
                nodeHours = nodes*hours
                totalNodeHours += nodeHours

    if totalNodeHours==0:
        print ("zero node hours found....aborting...")
        return

    kWh = totalNodeHours * (1200/4920)
    kgCO2 = kWh*0.5246
    text = """Total kWh for this folder: {0} \nTotal kg of CO2 for this folder: {1}""".format(kWh, kgCO2)
    print (text)
```

Docstring to describe what the module does

Import modules

Define function

Assign variable

Nested for loops

Conditional `If` statement

Correct indentation

Write to file

Math functions

String formatting

Print statements