



# Python in the Advanced Labs

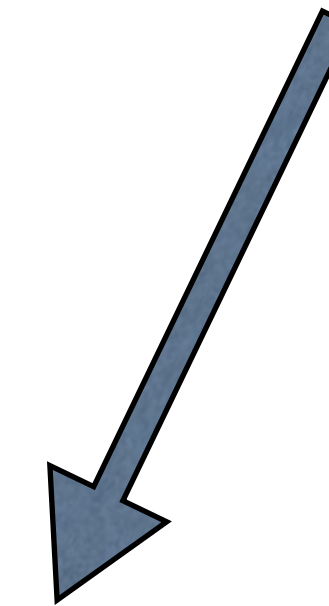
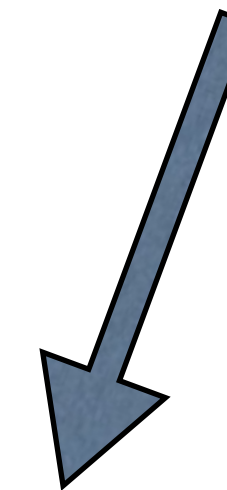
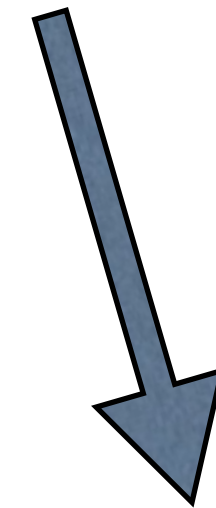
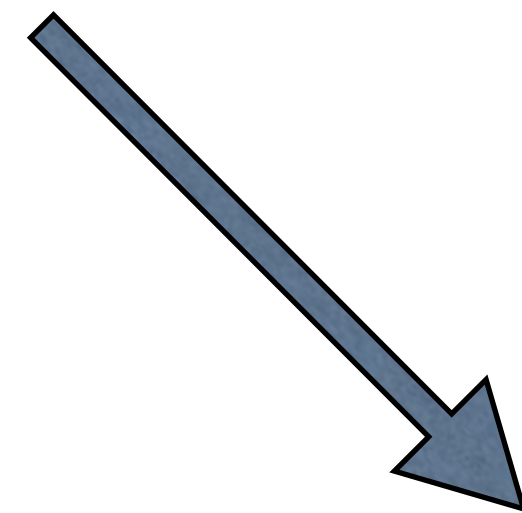
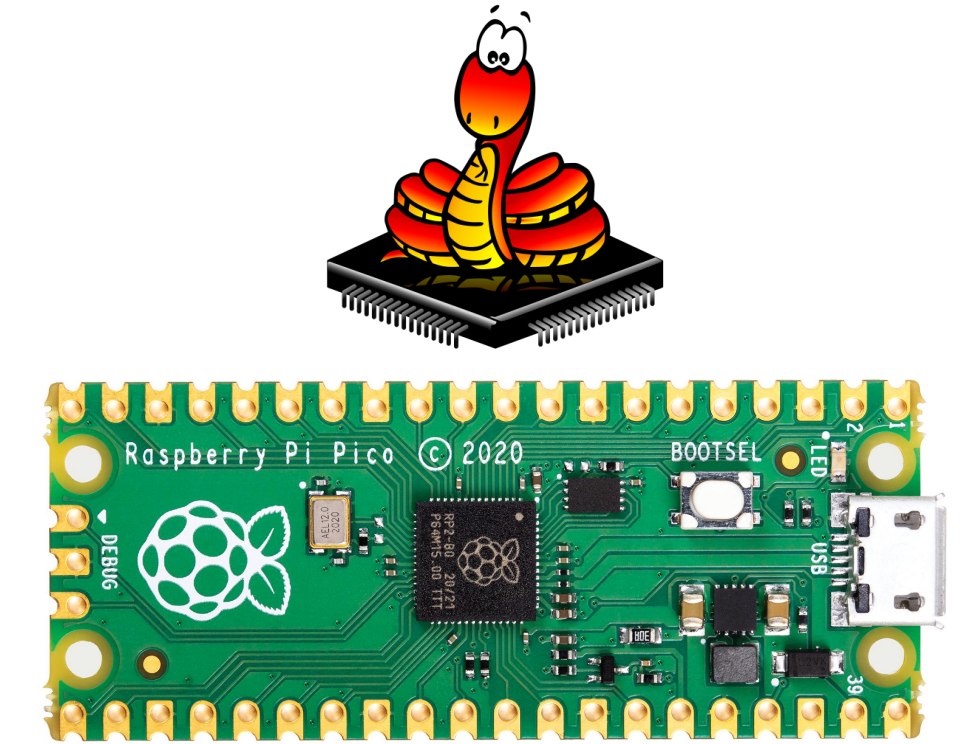
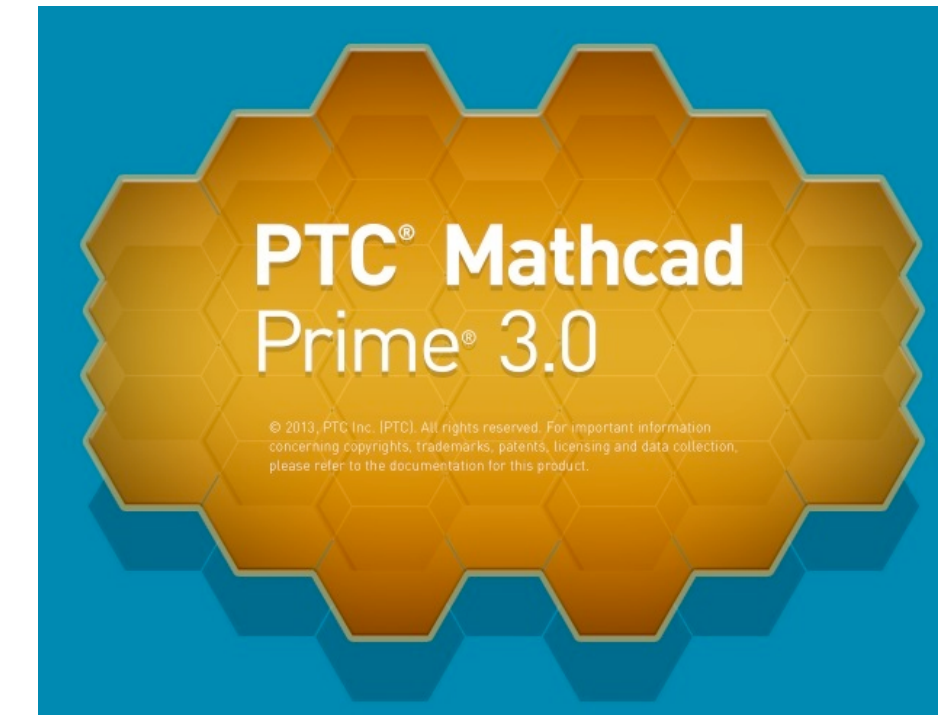
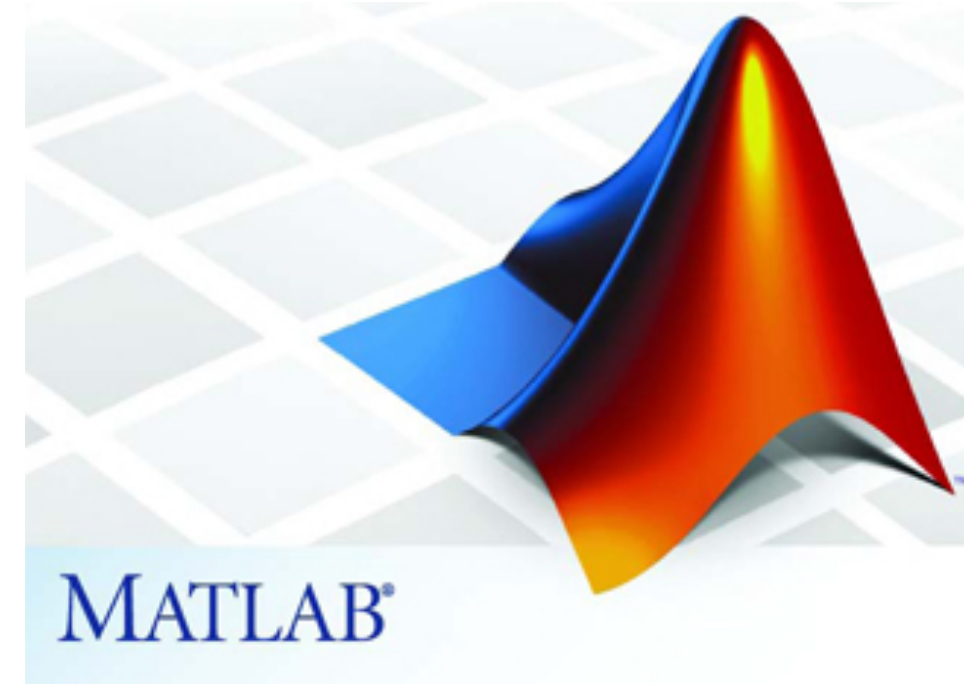
<https://physicslabs.ucd.ie>

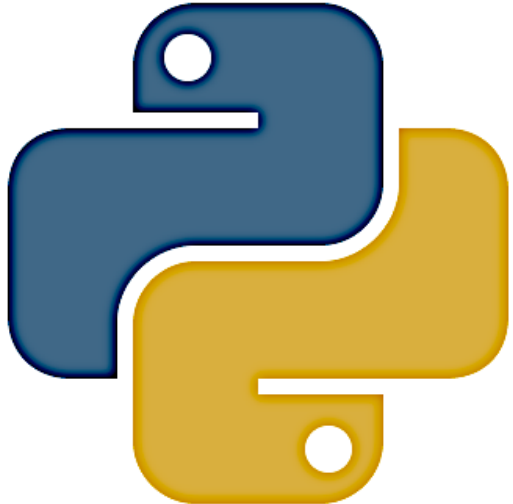
---

John Quinn

# A Unified Environment

C++



 python

Python is the programming framework for Data Acquisition, Data Analysis and Computational Physics in the Advanced Laboratories.

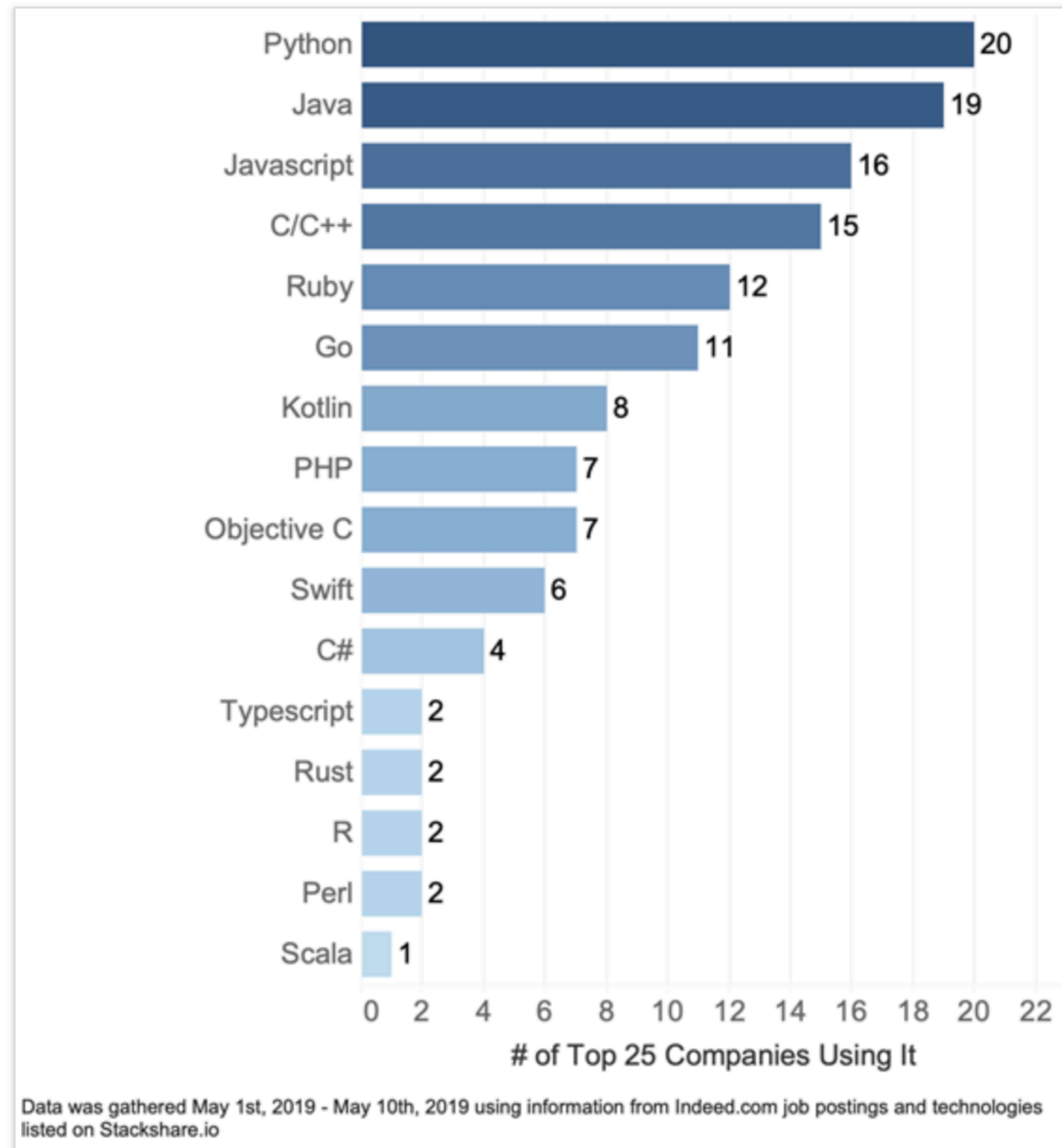


# Why Python?

---

- **Unified approach**: only one language to learn
- Python is a **mature high-level language** that is **easy to learn** (widely used for teaching programming) but with many **advanced features**.
- “**Batteries Included**”: many many packages and libraries, especially scientific, are available (and installed by default with Anaconda).
- Python and libraries are open source.
- Python is **increasingly used in Physics, Astronomy and Industry/Finance especially Data Analytics - Python skills are in demand!**
- However Python **is not perfect for everything** and in cases where efficiency and speed are needed, code is still written in C/C++ and called from Python.
- However, **Python is the most generally-useful programming language for Physics today**, due to the scientific modules that are available and the ease of development!

# Python in demand!



[source](#)

MUST READ: [Microsoft's top lawyer: Trump's Huawei ban makes no sense](#)

## Programming language popularity: Python tightens its grip at the top

Python retains its top spot as the most popular language for electrical and electronics engineers.

By [Liam Tung](#) | September 10, 2019 -- 10:55 GMT (11:55 BST) | Topic: [Developer](#)

[source](#)

## The top programming languages of 2019: Python is number one, say engineers

### Practical Deep Learning - 5 Day Training Course

Deep learning training for engineers & programmers using Python, TensorFlow, and Keras. [doulos.com](#)

OPEN

by [Nick Heath](#) in [Software](#) on September 9, 2019, 5:13 AM PST

The most popular languages according to the world's largest organization for engineering and applied science.

[source](#)

- Google: “most popular programming language scientific”



# Python issues

---

- It can be slow (there are ways to speed it up - generally not an issue in the labs.)
- It is a **growing and evolving language**:
  - new features appear steadily
  - there can be several ways to do something simple (e.g. string formatting)
  - you **never feel like you understand more than some small fraction of the language**
- The **number of libraries available is enormous**:
  - which one to choose and learn for a given task?
  - In the **Advanced Labs. we support standard Python standard libraries, Numpy, Scipy (some of!).**
    - **we do not support Pandas**
    - staff cannot be expected to know every library
    - If you feel you need to use some other library for your analysis please consult with the staff member in charge of the experiment.

# Python Programming

---

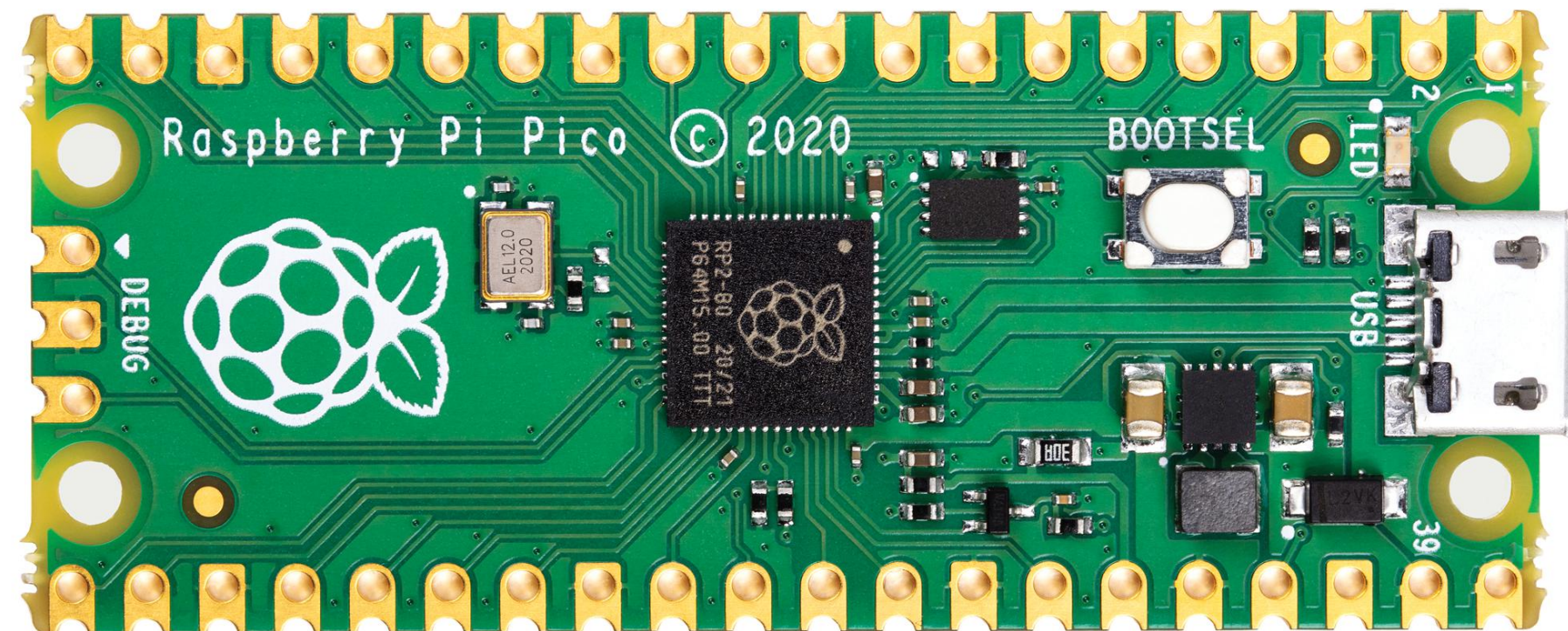


IP[y]:  
IPython

- All you need is a **text editor to type your programs** and you **can run them from the command line**.
- For data analysis etc, where a document record is useful, **Jupyter Notebook** is often the best choice.
- However, sometimes interactive iPython prompt or Python on the command line may be more appropriate.
- There are also interactive development environments (IDEs) such as Thonny, Spyder ...

# When would you not want a Jupyter Notebook?

- Libraries: code reuse
- Scripting and automating tasks
- Interactivity
- Server-side processing
- Embedded systems programming:



# Which Python?

- We use **Python 3** (3.8 is installed on lab. computers)
- **Anaconda** Scientific Python Distribution from Continuum Analytics:
  - Python 3.11 is now the default
  - Completely **free!**
  - Available for Linux, Mac OS X and Windows
  - Includes hundreds of the most popular Python packages including **Scipy**, **Numpy**, **Matplotlib**, **iPython** and **Jupyter Notebook**.  
<https://docs.anaconda.com/anaconda/packages/pkg-docs/>
  - Get from: <https://www.anaconda.com/download/>
- New! **Google Colab**: <https://colab.research.google.com>







# How to learn Python?

---

- We assume **everyone has basic Python from 2Y (Computational Science module)** but we recognise there will be a large range of abilities and experiences.
- We used to do intensive introductory sessions but mixed success - students felt too overwhelmed with new material at the start of Stage 3.
- **New approach:**
  - Documentation and examples online
  - HowTo's (developed by a Stage 3 student on a summer internship) in Google Colab. + ongoing development and improvement (feedback welcome!)
- **Informal tutorials can be given as needed:**
  - Please coordinate (Class reps) and let me know.



# Python Basics

- Basic Python language features you should be(come) familiar with:
  - **variables**: numbers (int, float), strings
  - **containers**: **lists, tuples and dictionaries**
    - indexing and slicing - also relevant for Numpy
  - **logical conditions**: **if... else...**
  - **looping**: **for** loops, **while** loops
  - **string formatting** (new in Python 3.6: **f-strings** - highly recommended you use!)
  - ~~writing data to a file~~ Use NumPy!
  - **functions**
  - **classes** (using them)

- **Scientific Python:**
  - NumPy
  - Matplotlib
  - SciPy



# Observations of Student Code

- Operator precedence catches people out - example:  $z = e^{-\frac{h\nu}{kT}}$

```
z = np.exp(-h*nu/k*T)
```

Incorrect: this is  $z = e^{-\frac{h\nu T}{k}}$

```
z = np.exp(-h*nu/(k*T))
```

correct

```
z = np.exp(-(h*nu)/(k*T))
```

correct

```
z = np.exp(-((h)*(nu))/((k)*(T)))
```

correct but hard to read!

- Please make yourself aware of operator precedence (order in which calculation are done) and strive to write correct and readable code!
- <https://docs.python.org/3/reference/expressions.html#operator-precedence>



# Observations of Student Code

---

- Code **not commented** and **poorly structured**
  - Long lines extended beyond edge of page ...
    - check what you submit that it is not truncated!
    - split code across multiple lines
  - **Variables poorly named**
- Code **overly complicated!**
  - strive for simplicity and readability
  - Don't try to automate a complicated analysis so it automatically analysed multiple data files
    - when something goes wrong it is difficult to track down!

# Recommendations

---

“Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.” (Kernighan’s Law)

- **Keep it simple!**
  - straightforward easy-to-read and follow code is much more understandable and maintainable than complicated code that is difficult to follow.
  - strive to improve - experience and practice is critical!
  - don’t be overwhelmed
    - ask for help and advice
  - be critical of results returned by google (e.g. stackoverflow)
- **Python Coding Guidelines:** [https://physicslabs.ucd.ie/docs/python/coding\\_guidelines/](https://physicslabs.ucd.ie/docs/python/coding_guidelines/)



# Code Plagiarism

- Copying code directly and including it is Plagiarism
- The UCD Plagiarism checker will find matches between your code and each other's, and online sites such as Stackoverflow.
- If you use a piece of code from the internet (beyond the trivial example of seeing how to call a function), justify why and reference it!
  - Googling for solutions (not reference documentation) should be a last resort not a first port of call.
    - be critical of the results returned by Google - is the source trustworthy?
  - the aim of the Advanced Lab is to get you THINK for yourselves and to be able to tackle new problems you have not met before, not to test your Googling skills!
- New for this academic year - how to deal with AI (ChatGPT, Code Lama, Google Bard, Github Co-Pilot...)
  - Use AI to learn but work you submit must be your own and you must be able to explain, stand over and defend every line of code you submit.



# [physicslabs.ucd.ie](http://physicslabs.ucd.ie)

---

- Over to web site....