Advanced Python

Module 7A.2. Part 2

1

1. What We'll Cover

Part 2 will explain…

- The Python Software Landscape
- The different Python programming environments.
- Setting up your own development environment.

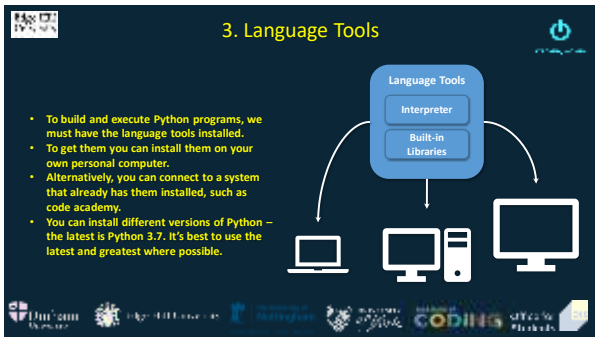
The aim: to get your ready to run the module activities, and setup your own development workflow.

2

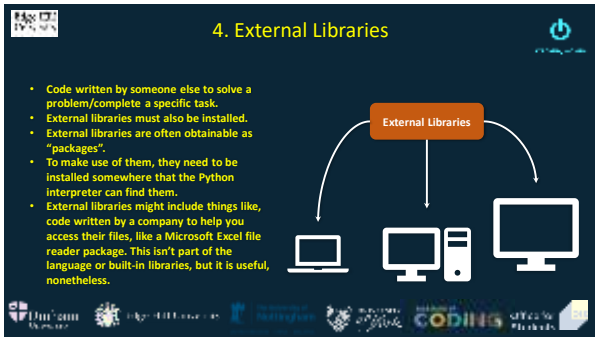
2. Understanding how things Fit

- Learning Python can be complicated because there are so many things to keep track of:
  - The language.
  - Coding in practice.
  - The interpreter.
  - Python libraries.
  - Code Editors
- Let's try to simplify the picture:
  - Language tools.
  - External Libraries
  - Tools

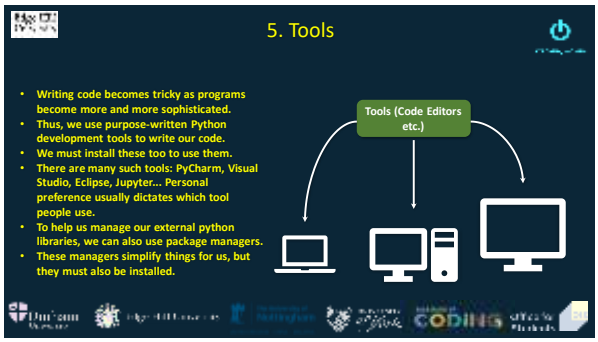
3


3. Language Tools
To build and execute Python programs, we must have the language tools installed.
To get them you can install them on your own personal computer.
Alternatively, you can connect to a system that already has them installed, such as code academy.
You can install different versions of Python – the latest is Python 3.7. It's best to use the latest and greatest where possible.
Language Tools
Interpreter
Built-in Libraries

4


4. External Libraries
Code written by someone else to solve a problem/complete a specific task.
External libraries must also be installed.
External libraries are often obtainable as "packages".
To make use of them, they need to be installed somewhere that the Python interpreter can find them.
External libraries might include things like, code written by a company to help you access their files, like a Microsoft Excel file reader package. This isn't part of the language or built-in libraries, but it is useful, nonetheless.
External Libraries

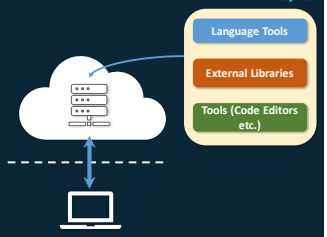
5


5. Tools
Writing code becomes tricky as programs become more and more sophisticated.
Thus, we use purpose-written Python development tools to write our code.
We must install these too to use them.
There are many such tools: PyCharm, Visual Studio, Eclipse, Jupyter… Personal preference usually dictates which tool people use.
To help us manage our external python libraries, we can also use package managers.
These managers simplify things for us, but they must also be installed.
Tools (Code Editors etc.)

6

6. Complex Software Ecosystem

- I Imagine this software ecosystem appears daunting at first – so many things to learn about! But we don't have to let this scare us off.
- Environments like this can be provided for us, with all these software systems pre-configured.
- Some have even packaged them up to be hosted from web-based computers.
- You can connect to these and run your code without issue.



7

7. Example: Google Colab

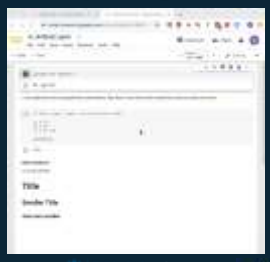
- Google Collaboratory is online environment that contains all the tools necessary to write and execute Python programs.
- To use the Collaboratory, all you'll need is a google account, e.g. a Gmail Account.
- When you login to the Collaboratory, it creates a computer just for you to work with.
- The environment will allow you to create Python code, and execute it.
- The environment is special, as it allows code to be executed in an interactive fashion. This means you don't need to write a whole source code file, before you can run some code.
- Instead you can execute individual commands one at a time. This is great for learning.
- Before proceeding, please create a google account if you don't already have one.

8

8. Example: Google Colab

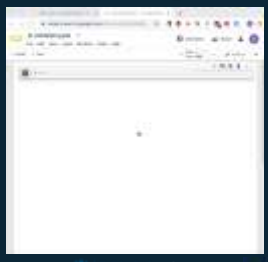
9

9. Example: Google Colab



10

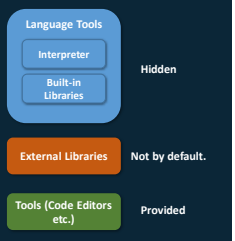
10. Example: Google Colab



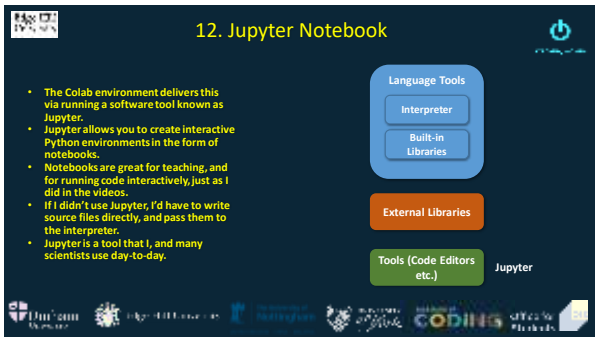
11

11. Understanding the Colab

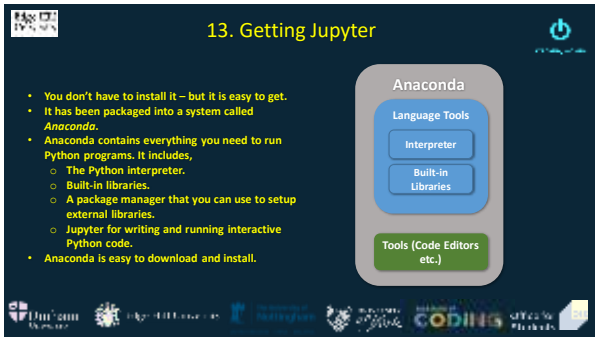
- In the Colab, the interpreter and built-in libraries are hidden.
- There are no external libraries loaded by default – but you can load them for yourself – this is an advanced topic that we will cover later.
- The Colab doesn't hide the tools at your disposal – it provides a friendly user interface via which you can interact with Python.



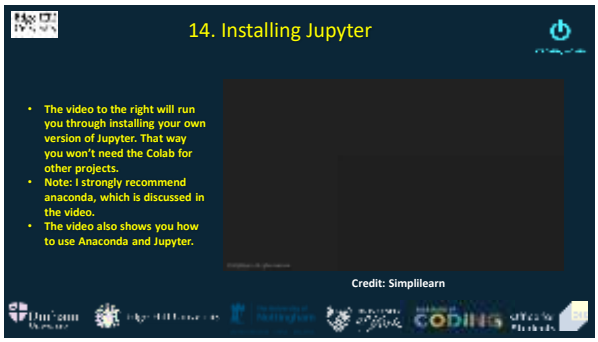
12


12. Jupyter Notebook
The Colab environment delivers this via running a software tool known as Jupyter.
Jupyter allows you to create interactive Python environments in the form of notebooks.
Notebooks are great for teaching, and for running code interactively, just as I did in the videos.
If I didn't use Jupyter, I'd have to write source files directly, and pass them to the interpreter.
Jupyter is a tool that I, and many scientists use day-to-day.
Language Tools
Interpreter
Built-in Libraries
External Libraries
Tools (Code Editors etc.)
Jupyter

13


13. Getting Jupyter
You don't have to install it – but it is easy to get.
It has been packaged into a system called Anaconda.
Anaconda contains everything you need to run Python programs. It includes,
The Python interpreter.
Built-in libraries.
A package manager that you can use to setup external libraries.
Jupyter for writing and running interactive Python code.
Anaconda is easy to download and install.
Anaconda
Language Tools
Interpreter
Built-in Libraries
Tools (Code Editors etc.)

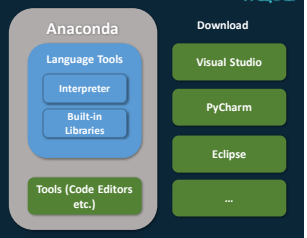
14


14. Installing Jupyter
The video to the right will run you through installing your own version of Jupyter. That way you won't need the Colab for other projects.
Note: I strongly recommend anaconda, which is discussed in the video.
The video also shows you how to use Anaconda and Jupyter.
Credit: Simplilearn

15

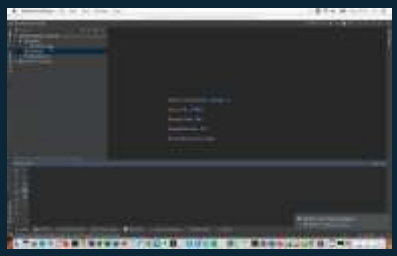
15. Another way

- If you installed anaconda, there is another way you can write and run Python code.
- You don't need to use all the tools Anaconda provides.
- You can download your own code editors and use those to create and run applications.
- This is how most software companies will create code.
- I'll show an example of this next.



16

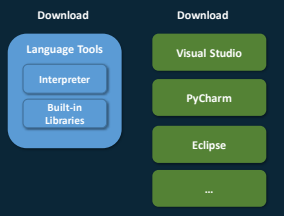
16. Example



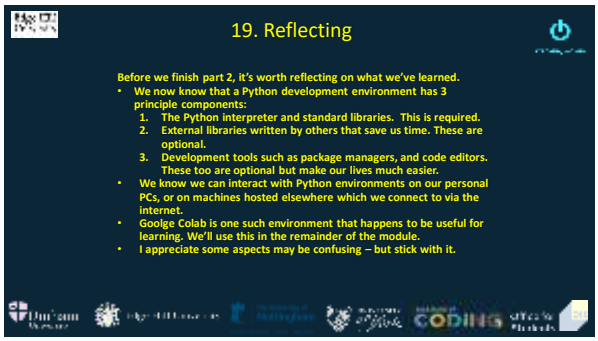
17

17. From Scratch

- You can always install Python without Anaconda.
- This requires more effort and can be complicated. You have to download all tools for yourself and manage the Python packages.
- I won't show you how this is done as I don't want to create confusion – it's best to just install Anaconda.
- For those that are adventurous, here's a video link that will walk you through the process:
  - For Windows: https://www.youtube.com/watch?v=ndrCfBJkkvE
  - For Mac: https://www.youtube.com/watch?v=TgA4ObrowRg



18


19. Reflecting
Before we finish part 2, it's worth reflecting on what we've learned.
• We now know that a Python development environment has 3 principle components:
1. The Python interpreter and standard libraries. This is required.
2. External libraries written by others that save us time. These are optional.
3. Development tools such as package managers, and code editors. These too are optional but make our lives much easier.
• We know we can interact with Python environments on our personal PCs, or on machines hosted elsewhere which we connect to via the internet.
• Goolge Colab is one such environment that happens to be useful for learning. We'll use this in the remainder of the module.
• I appreciate some aspects may be confusing – but stick with it.

19


20. Summary
Here we've introduced,
• The Python Software Landscape – comprised of the standard libraries, external modules, and development tools.
• Google Colab environment.
• Jupyter notebooks.
• Setting up your own development environment in a variety of ways.
Next we get back to the fun stuff – coding. For those that want a head start, here's a link to the Colab notebook we'll be working on next.
https://colab.research.google.com/drive/1JNwsQ6PM7IifWK2fXMEa0PPjwZRjxlMj

20


21. Links
Useful links:
• PyCharm community edition: https://www.jetbrains.com/pycharm/download/
• Eclipse IDE: https://www.eclipse.org/downloads/
• Visual Studio: https://visualstudio.microsoft.com/
• Anaconda: https://www.anaconda.com/
• Jupyter: https://jupyter.org/
• Another cool Python Course covering software installation to writing code:
https://www.youtube.com/watch?v=_uQrJ0TkZlc

21