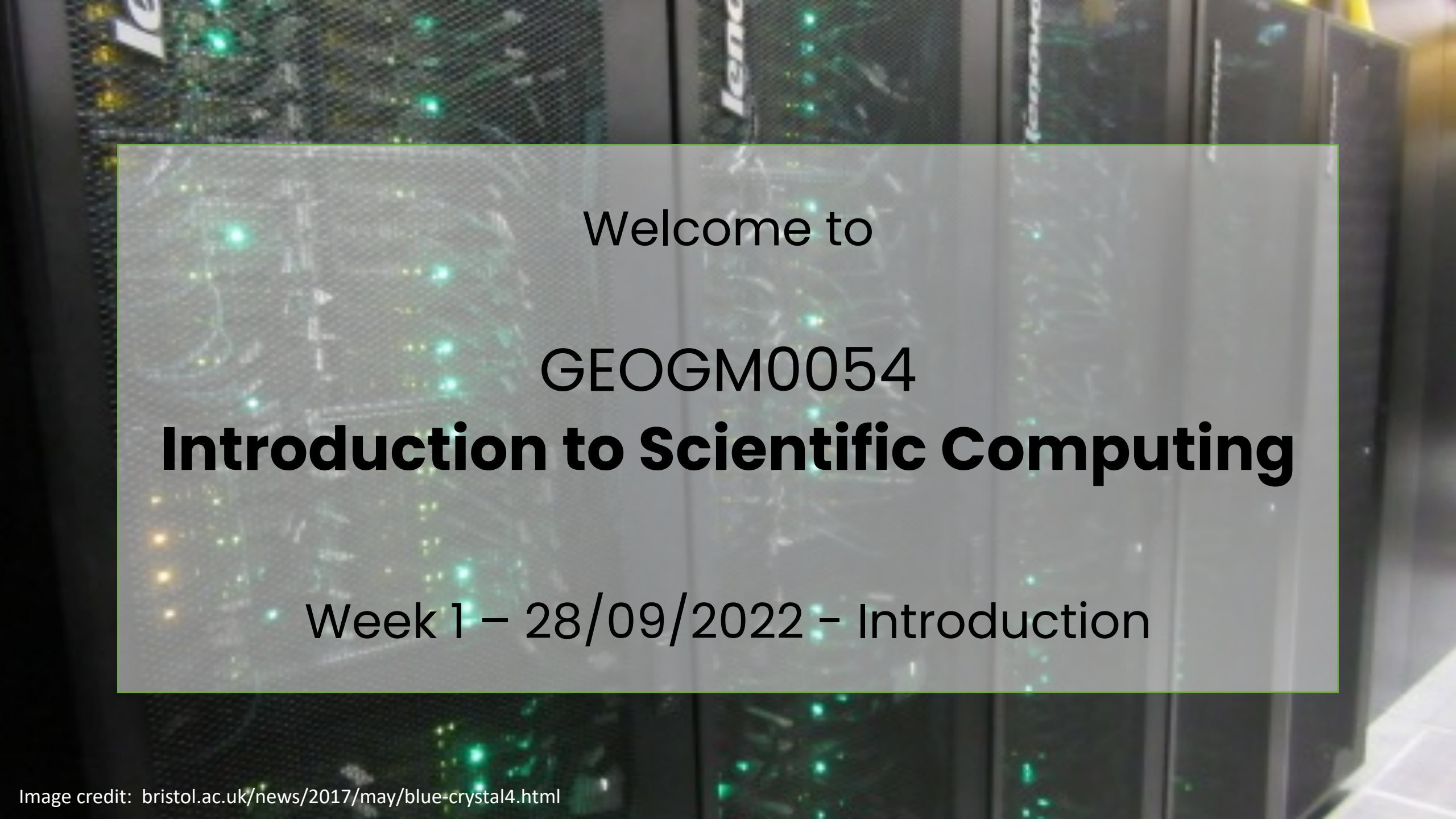


Welcome to

GEOGM0054

Introduction to Scientific Computing

Week 1 – 28/09/2022 - Introduction

Image credit: bristol.ac.uk/news/2017/may/blue-crystal4.html

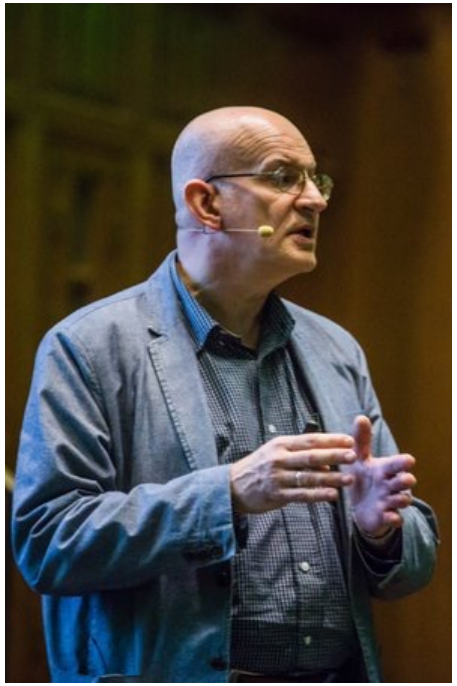
About us

Seb Steinig

sebastian.steinig@bristol.ac.uk
Office hours: Friday 11am-noon

Steph Cornford

s.l.cornford@bristol.ac.uk
Office hours: Monday 3-4pm



Tony Payne

a.j.payne@bristol.ac.uk
Office hours: Friday 10-11am

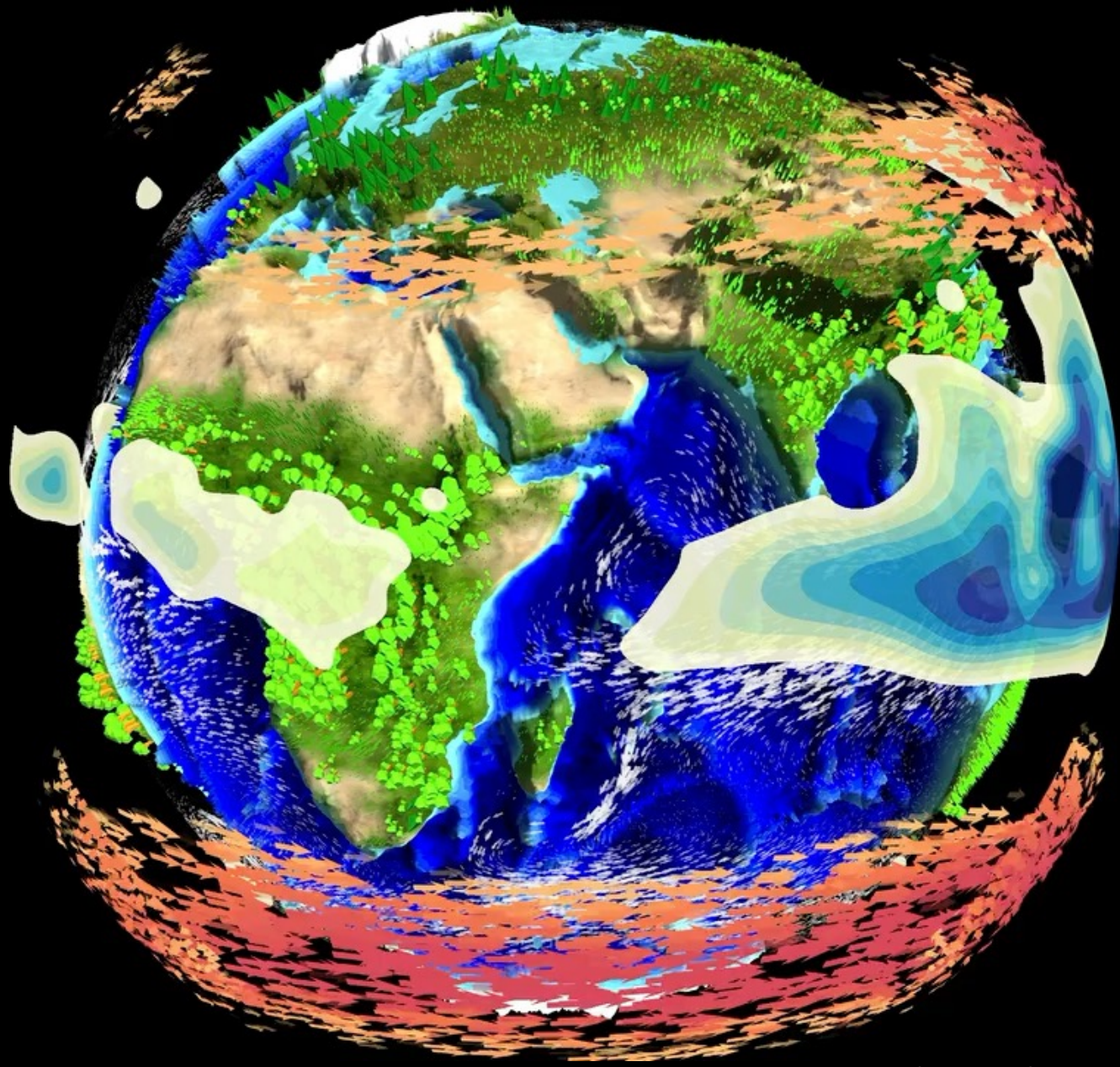
Claude de Rijke-Thomas

Demonstrator

Seb Steinig

sebastian.steinig@bristol.ac.uk
@sebsteinig

- Research on deep-time climate modelling (last 500 million years)
- Climate change on geologic timescales and past warm periods
- Trying to make climate model data more accessible
- Using Python for model data processing and visualisation

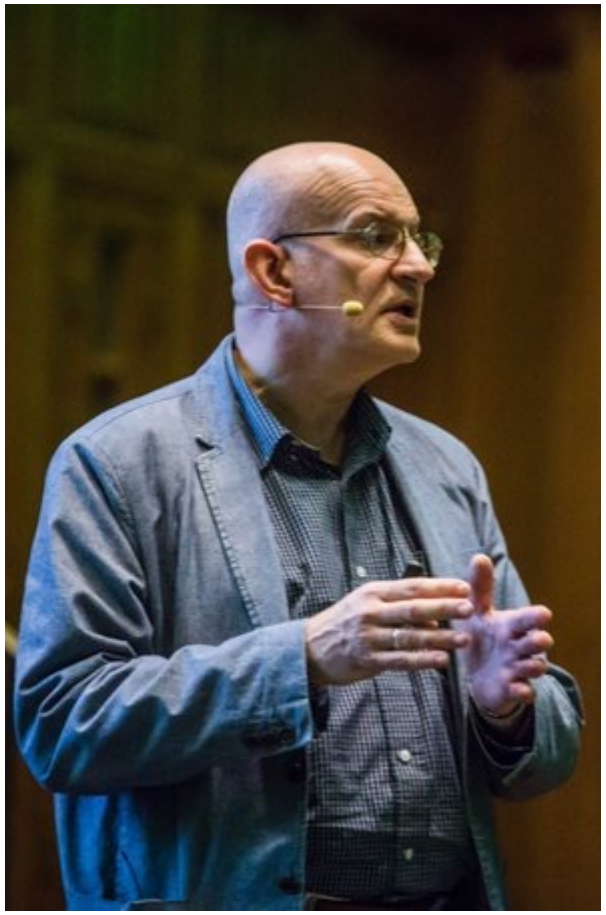


climatearchive.org

Steph Cornford

- Teaching on “Introduction to Scientific Computing” and unit convenor of “Environmental Modelling Applications” in TB2
- Research on computational glaciology, including projections of the Antarctic Ice Sheet under future ocean warming
- Developing techniques to represent processes on different length scales in adaptive mesh ice-sheet models
- s.l.cornford@bristol.ac.uk

Tony Payne



- 3rd year undergraduate teaching on ‘Sea Level past, present and future’
- Programme Director for MSc in Environmental Modelling and Data Analysis
- Research using regional coupled ice sheet-ocean models and global climate models to project sea level
- a.j.payne@bristol.ac.uk

Claude de Rijke-Thomas

- MSci Physics
- PhD Student in Physical Glaciology – estimates snow depths and properties over Arctic sea ice, using aircraft and satellite data (in Python^^)
- Demonstrating for this module
- Likes climbing (trees and rock), archery, kayaking, gaming and playing piano
- claude.derijke-thomas@bristol.ac.uk

Outline for today

- Overview of the course structure
- Assessment
- Why do we learn Python?
- How do we learn Python?
- Group exercise: algorithmic thinking
- Your first line of code!

Learning outcomes

After completing this unit, you should be able to:

- understand the complexity and the fundamentals of scientific computing
- use scientific computing to solve Geographic Data Science problems
- explain, understand, and employ analytical methods to analyse geospatial data sets
- produce reproducible workflows
- collaboratively develop and deploy scientific software with version control on Linux-based systems
- develop high-performance computing strategies to do data science at scale

part 1

Python for Geographic Data Science / Seb Steinig

- Week 1 (Wed, 28 September):
  - **Lecture 1: Introduction and unit overview**
- Week 2 (Wed, 5 October):
  - **Lecture 2: Introduction to Python**
- Week 3 (Wed, 12 October):
  - **Lecture 3: Code structuring**
- Week 4 (Wed, 19 October):
  - **Lecture 4: Numerical Python**
- Week 5 (Wed, 26 October):
  - **Lecture 5: Analysing geospatial datasets**
- Week 6 (Wed, 2 November):
  - **Lecture 6: Review of part 1 and introduction to assessment #1**

part 2

High-performance computing / Tony Payne & Steph Cornford

exact order might change at a later stage

- Week 7 (Wed, 9 November):
  - **Lecture 7: Profiling, debugging and IDEs (Steph)**
- Week 8 (Wed, 16 November):
  - **Lecture 8: Version control with git and GitHub (Steph)**
- Week 9 (Wed, 23 November):
  - **Lecture 9: Linux introduction (Tony)**
- Week 10 (Wed, 30 November):
  - **Lecture 10: Using high-performance computing (Tony)**
- Week 11 (Wed, 7 December):
  - **Lecture 11: Parallelising code (Steph)**
- Week 12 (Wed, 14 December):
  - **Lecture 12: Review of Part 2 and introduction to assessment #2**

Assessment

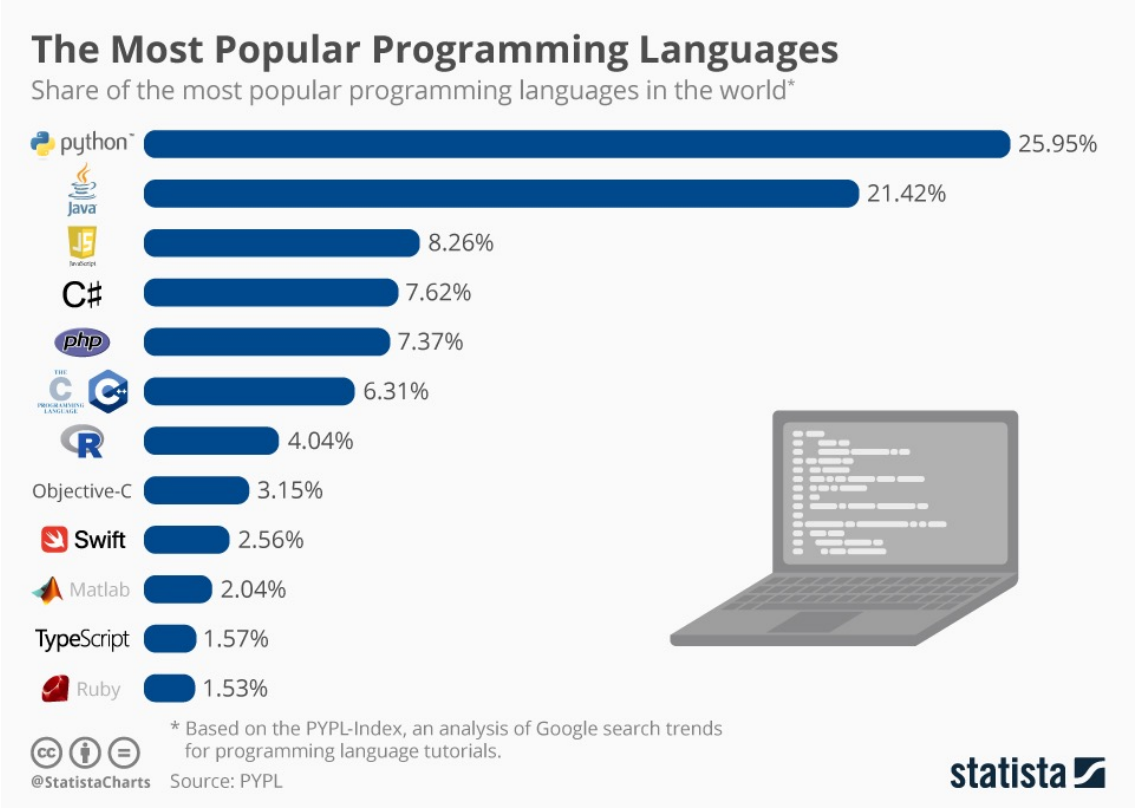
- **Part 1 (Python for Geographic Data Science):**
  - apply skills from first part of the unit to analyse a geospatial data set
  - notebook (Code+Figures) + 2000-word essay/practical report: 50%
  - **introduced in week 6 with hand-in date in mid-November**

- **Part 2 (High-performance computing):**
  - version-controlled code development and debugging
  - written code + 2000-word essay/practical report: 50%
  - **introduced in week 12 with hand-in date in mid-January**

exact hand-in dates will be announced in upcoming lecture and on Blackboard

Why do we learn Python?

- a programming language allows us to give a set of instructions to a computer
- so which is the best programming language?
  - counter question: Which is the best human language?
- it depends on the purpose, but Python is a very popular choice because:
  - general-purpose and open-source
  - high-level language
  - interpreted language

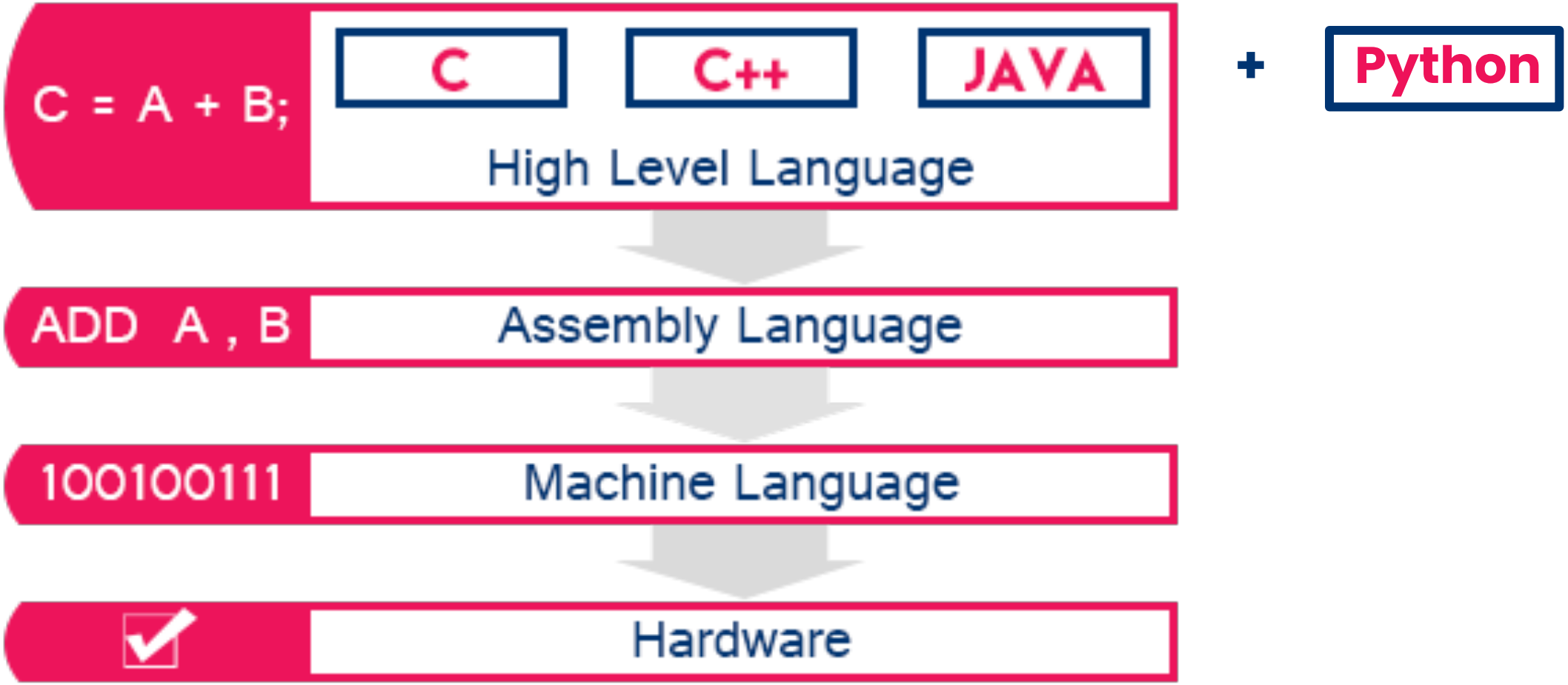


https://www.statista.com/chart/16567/popular-programming-languages/

1. Python is general-purpose and open-source

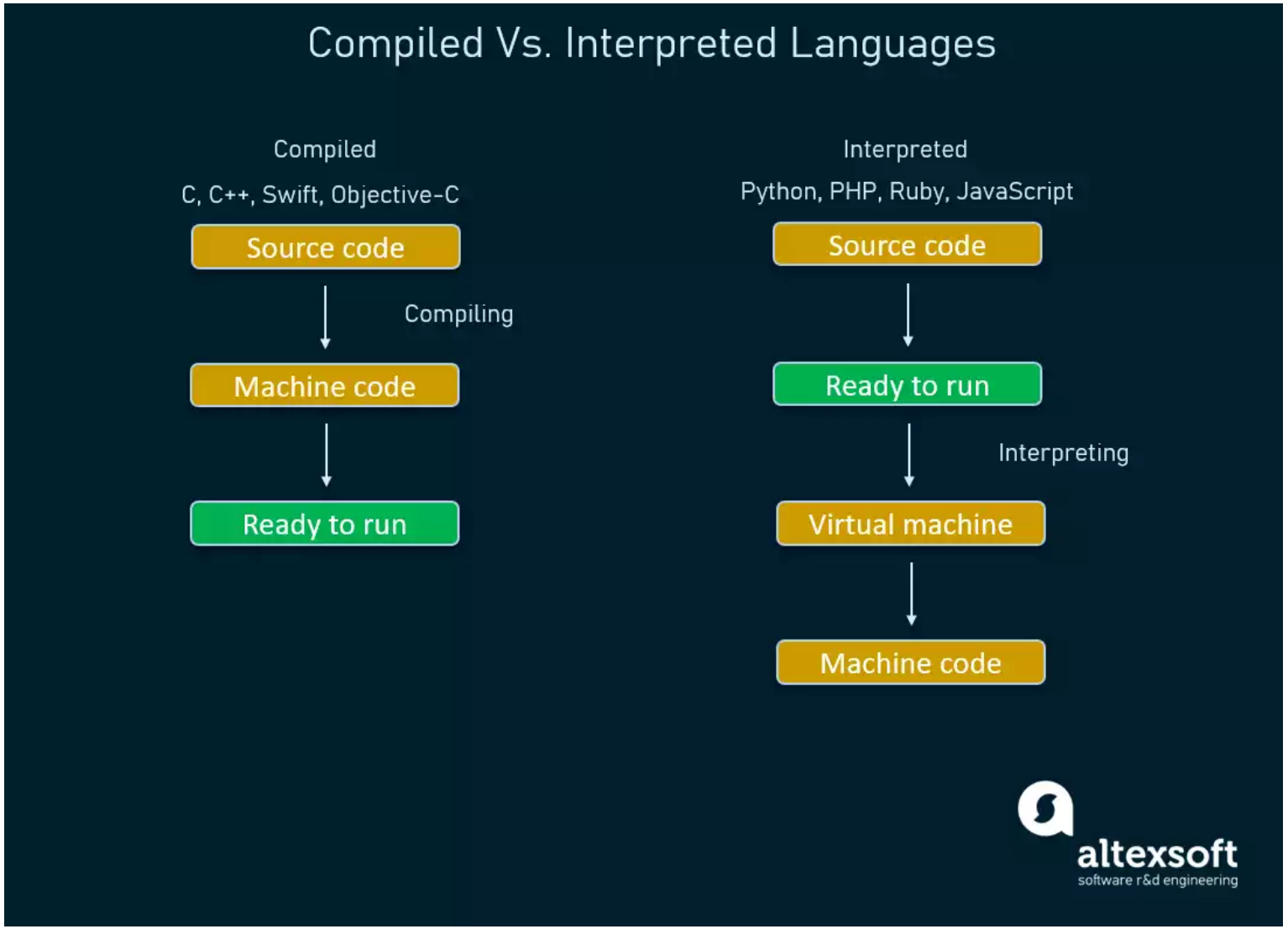
https://www.jetbrains.com/lp/python-developers-survey-2020/

2. Python is a high-level Language



http://www.btechsmartclass.com/c_programming/C-Computer-Languages.html

3. Python is an interpreted Language



https://www.altexsoft.com/blog/python-pros-and-cons/

Why do we learn Python?

1. general-purpose and open-source

- flexible and widely used with a large and active community
- de facto standard in data science and machine learning

2. high-level language

- simplified syntax (i.e. language rules) make it easy to learn for beginners
- high readability (e.g. no need to declare variable types = dynamic typing)

3. interpreted language

- saves you additional compilation step -> rapid development cycle
- code is interpreted/translated line-by-line -> easier to debug

How do we learn Python?

Three important steps : Practice, make mistakes, and practice more!

Joint Learning:

- new exercises/workbook uploaded to Blackboard every Monday (under Learning Materials)
- ~two-hour computer-lab based lecture on **Wednesdays at 9am in the Fry building in LG.21**
- introduction of new concepts by lecturer, but focus on solving exercises by writing your own code
- work at your own speed and finish exercises during the rest of the week
- you can bring your own laptop or work on the provided computers

Independent Learning:

- **technical unit**, i.e. time you would normally put into a unit reading relevant text books and literature is spent doing homework/coding exercises
- amazing, free online resources available; worth to try additional approach to learn coding
- links to online courses, textbooks and YouTube videos will be added to Blackboard (under Resource lists)
- maybe you already have small coding project in mind - now would be a great time to get started!

20 credit points unit -> ~200 hours to complete, made up of contact time, directed learning tasks, independent learning and assessment activity

Workspaces

Joint Learning:

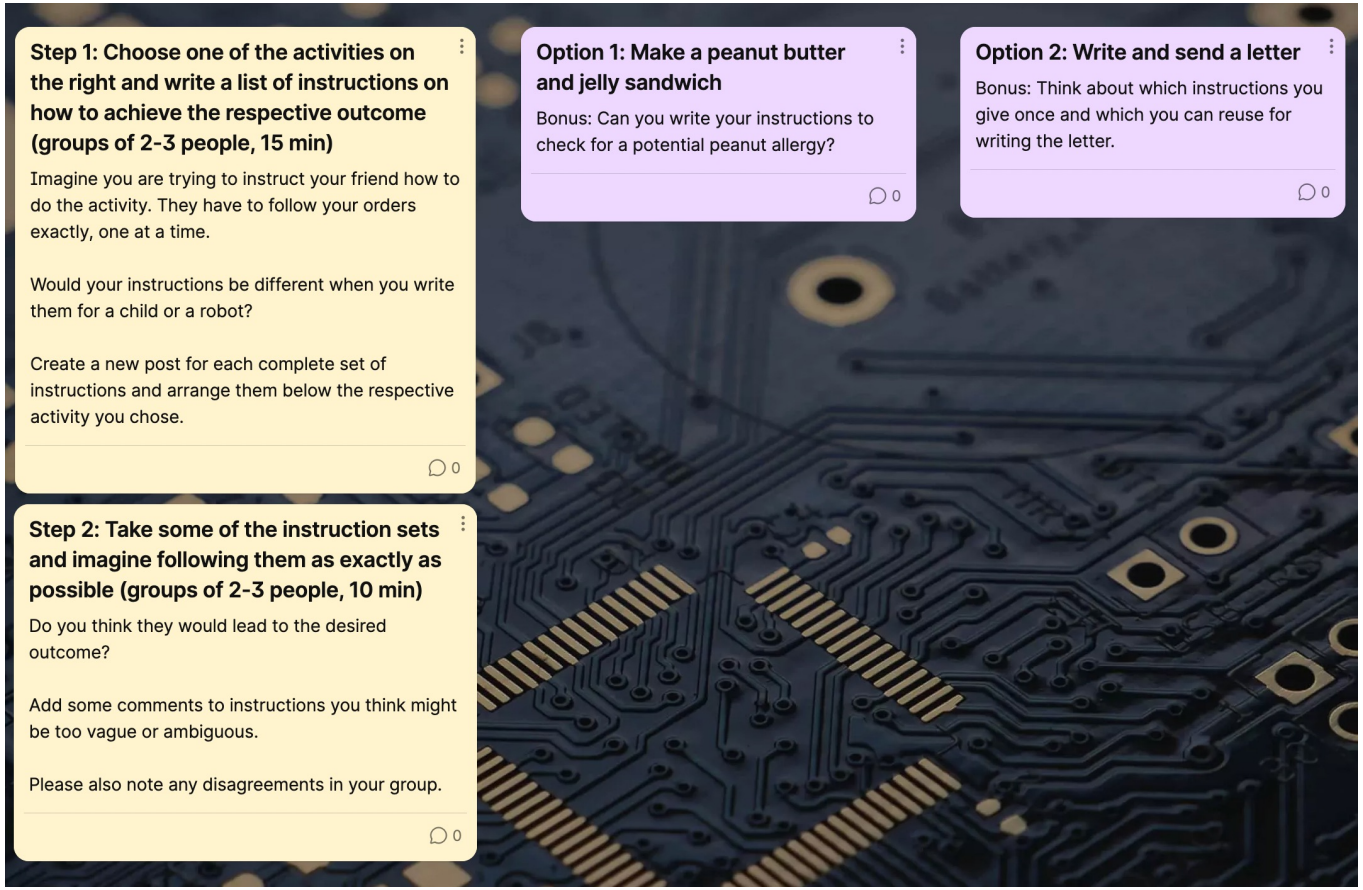
- ~two-hour computer-lab based lecture on **Wednesdays at 9am in the Fry building in LG.21**
- you can bring your own laptop or work on the provided computers

Independent Learning:

- the Fry computer lab LG.21 is for teaching only
- outside of lectures, you have access to identical PCs in:
  - Geography MSc study space (next to Fry computer lab, **Math building**)
  - Haggett Computer Lab (room 1.3N, **Geography building**)
  - Small Computer Room / Mini Hagget (room 1.4N, **Geography building**)
- or continue to work on your own laptop (all software is freely available for Windows, Mac and Linux)

Group exercise: algorithmic thinking

- before we start coding next week, let's try to start thinking like a programmer!
- **Task:** Write a set of instructions for a seemingly very easy activity.
  - use the Padlet link on the right to write your instructions (anonymous, no marking)
  - you can work and submit your instructions in groups of 2-3 people



https://uob.padlet.org/sebastiansteinig1/gztdm5ny99dfylxk

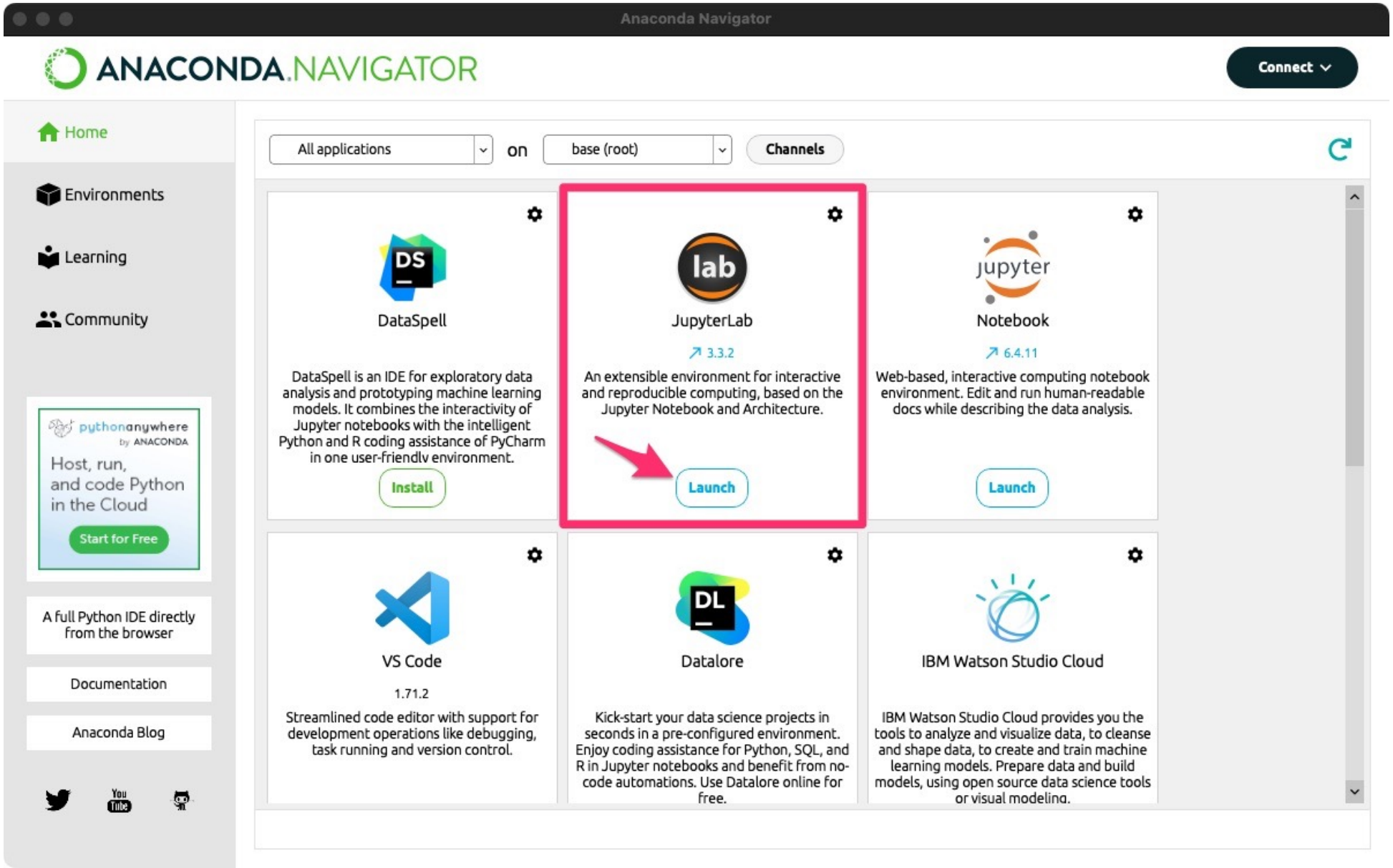
PW: GEOGM0054

https://youtu.be/cDA3_5982h8

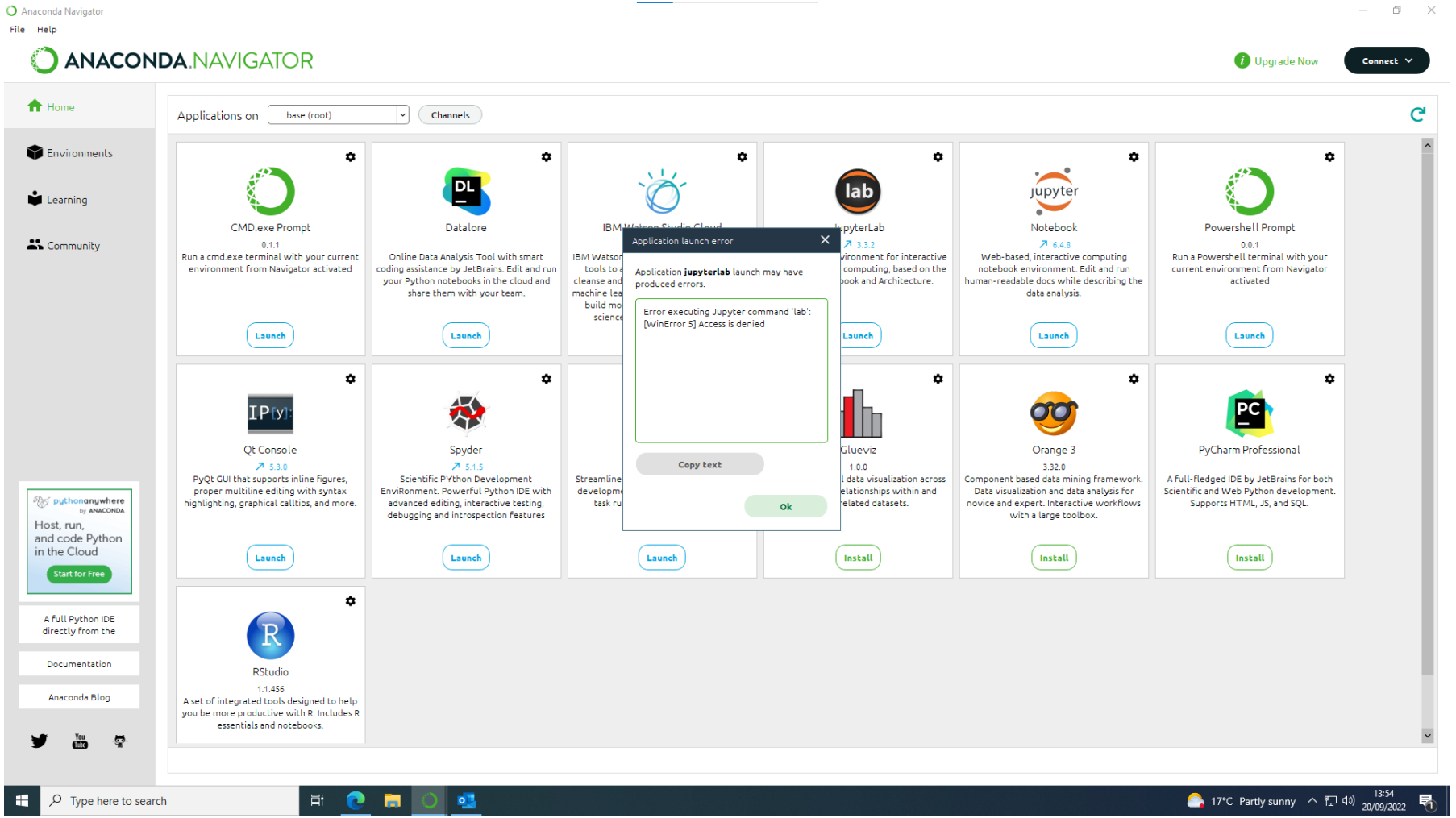
Getting started with Python!

- we will be using a free tool called JupyterLab which provides you with a local editor in your web browser where you can write and run Python code
- JupyterLab should be pre-installed on all UoB computers in the teaching lab, the MSc study space and the Geography Haggett Lab
- you can start it by launching the Anaconda Navigator from the Windows start menu
- if you want to work on your own laptop:
  - the easiest way to get access to JupyterLab is to install Anaconda 3, which is a piece of software which includes Python along with lots of other tools
  - it is freely available for Windows, Mac and Linux
  - go to https://www.anaconda.com/products/distribution and download the respective anaconda distribution for your operating system
  - open the Anaconda Navigator after the installation has finished

you should see the window below:

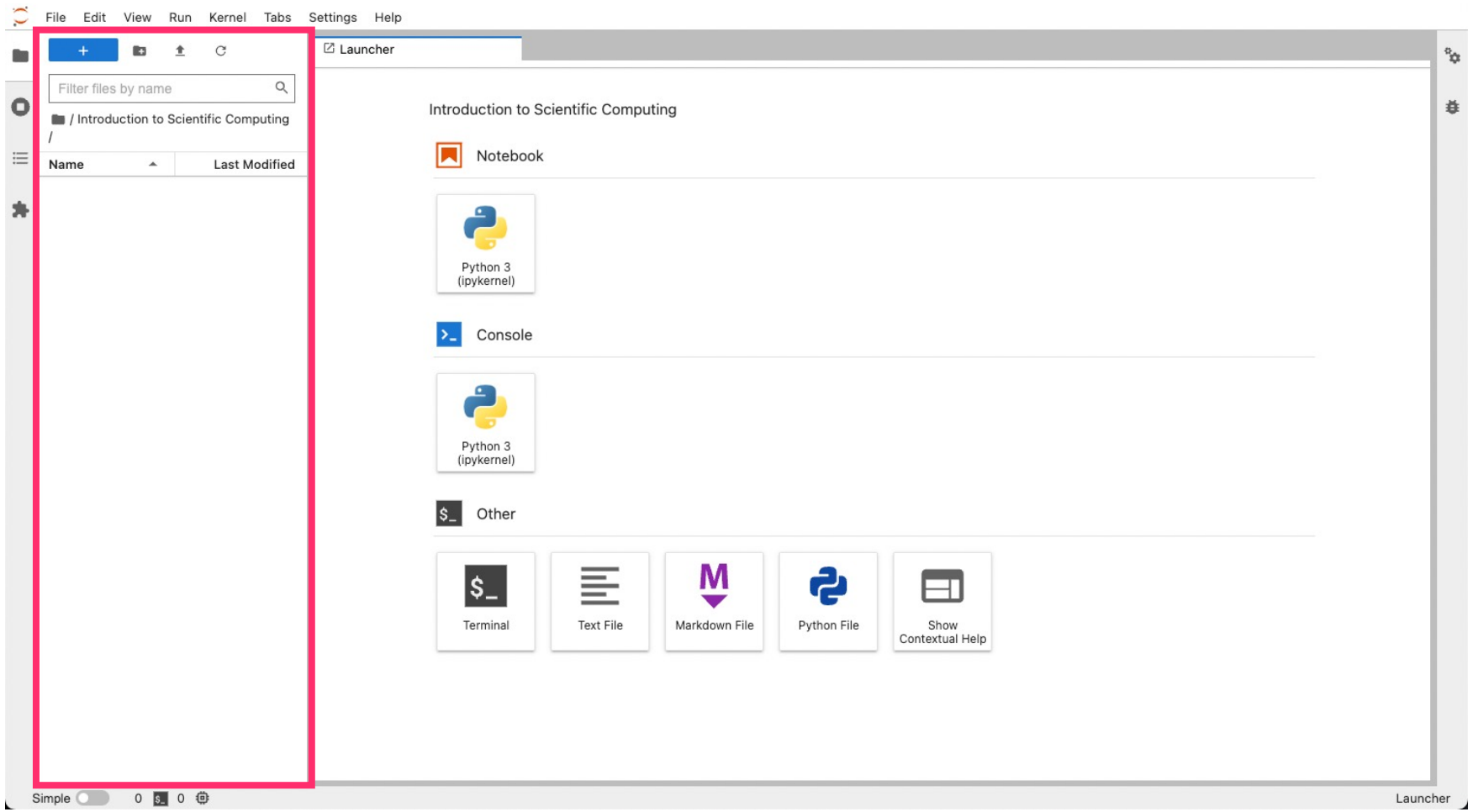


launching JupyterLab from an UoB computer might produce this error:



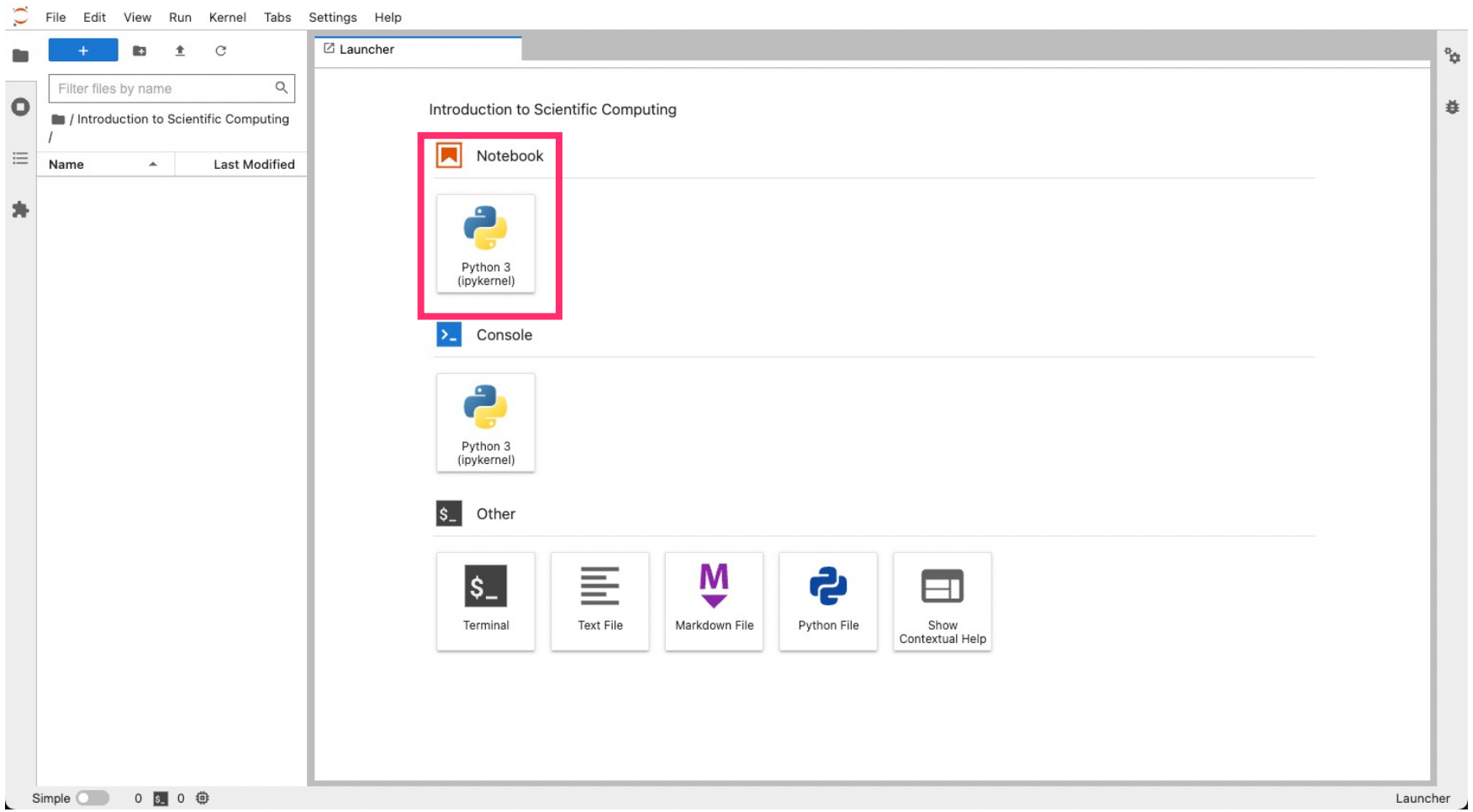
if this happens, just launch the ‘Notebook’ application instead and change the last part of the URL in your browser from ‘…/tree’ to ‘…/lab’

everybody should now get a browser window that looks similar to this:

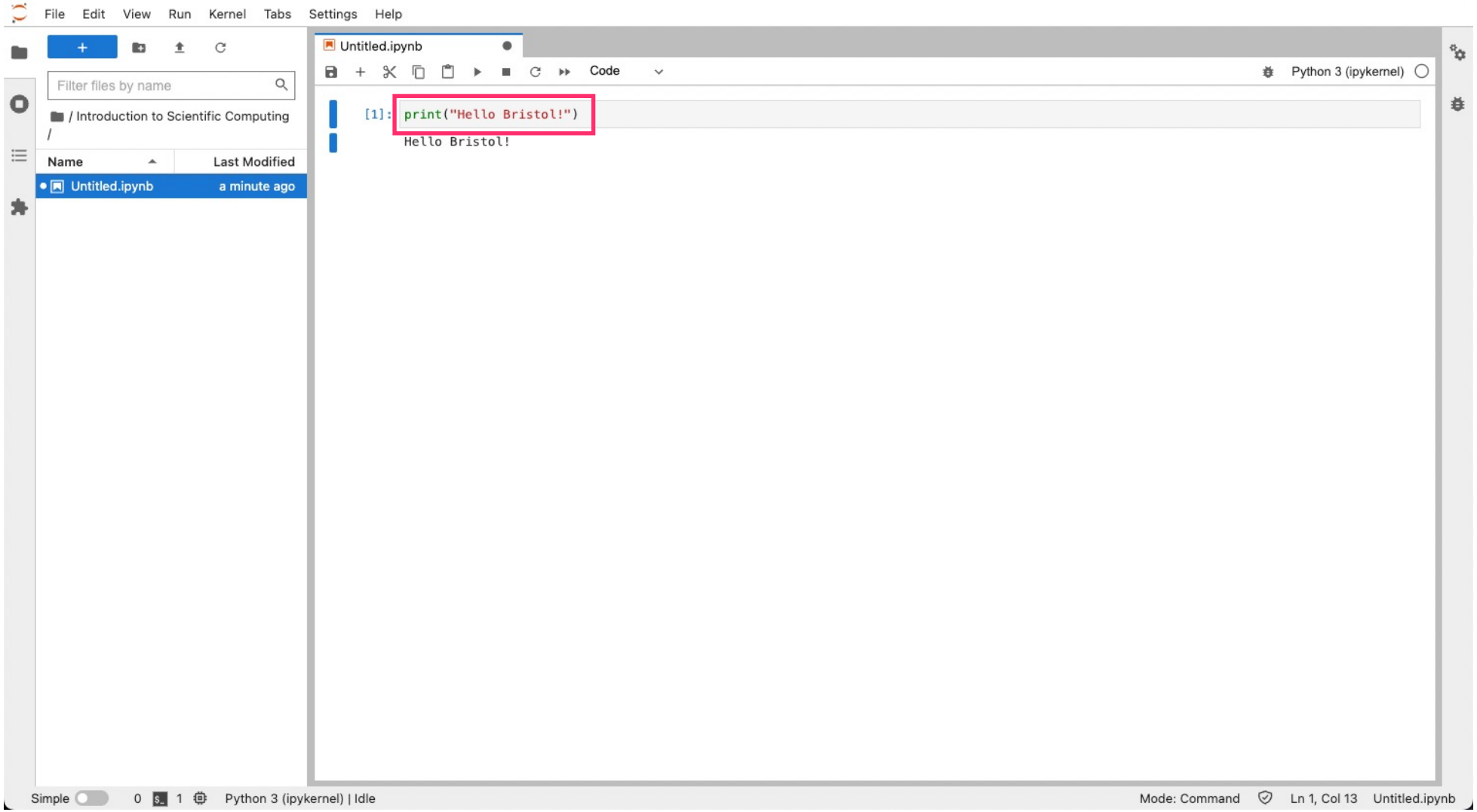


- let's create a new work directory that you will use for the rest of the course
- you can use the file browser on the left to navigate through your local files
- ideally, you want to create this within your OneDrive folder for backup and syncing

we can now use the launcher to open our first notebook …



… and write our first line of code!



- you can run the code by clicking the 'Play' button in the toolbar or by pressing Shift+Enter
- **congratulations, you just used the Python programming language to give instructions to the computer!**