

# Introduction

FIE463: Numerical Methods in Macroeconomics and Finance using Python

Richard Foltyn

NHH Norwegian School of Economics

January 15, 2026

# Contents

## 1 Introduction to Python

- Why Python?
- Python vs. other languages
- Python ecosystem

## 2 Software & tools

## 3 Course outline & assessment

- Course outline
- Assessment

## 4 Additional resources

- Books & websites
- Video tutorials

# About me

- Undergraduate studies in software engineering (& economics), PhD in Economics
- Research fields: Quantitative Macroeconomics & Household Finance
- 20+ years of programming experience:
  - Languages used previously (and mostly forgotten):  
C/C++, Visual Basic, Java, JavaScript, PHP, Perl, SQL, Matlab, R
  - These days:  
Python, Fortran, Unix shell scripts, Stata

## Contact

- Email: [richard.foltyn@nhh.no](mailto:richard.foltyn@nhh.no)
- Office: D231 (SAM, 2<sup>nd</sup> floor in the new building)

# INTRODUCTION TO PYTHON

# Why Python? ... and why not?

## Why Python?

- Free and open-source
- Easy to learn, yet powerful and flexible syntax
- General-purpose language that can be used to solve many different problems
- Huge ecosystem of libraries and tools
- By now the most popular language overall
  - Most popular in machine learning
  - One of the two most popular in data science (together with R)
- It may not be the fastest, but offers easy ways to accelerate things (Cython, Numba, JAX, ML libraries)

## What can you do with Python?

- Everything. The question is whether you should be using Python ...

## Why not Python?

- You already know another language that solves your problem well
- You want to use an estimator/algorithim that is implemented elsewhere (Stata, R), but not in Python

# Python popularity (1)

Since its creation in the 1990s, Python has climbed to the top of almost any programming language ranking.

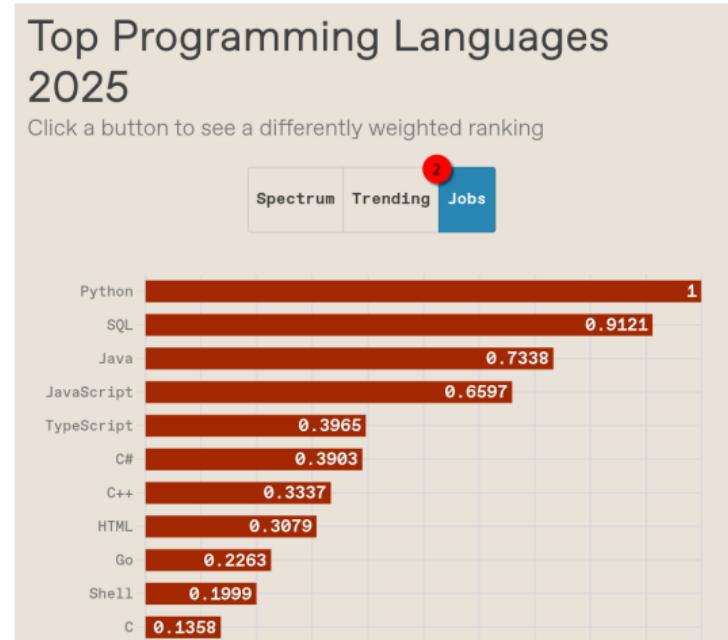
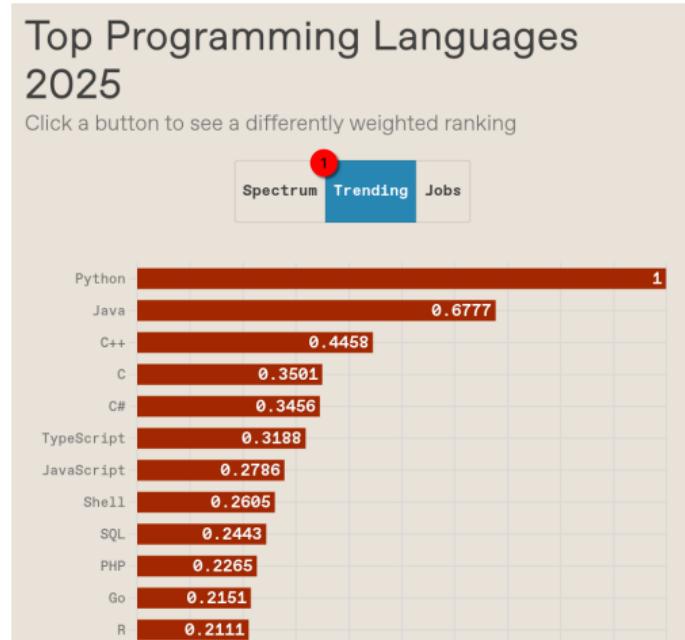


Figure 1: Source: IEEE The Top Programming Languages 2025

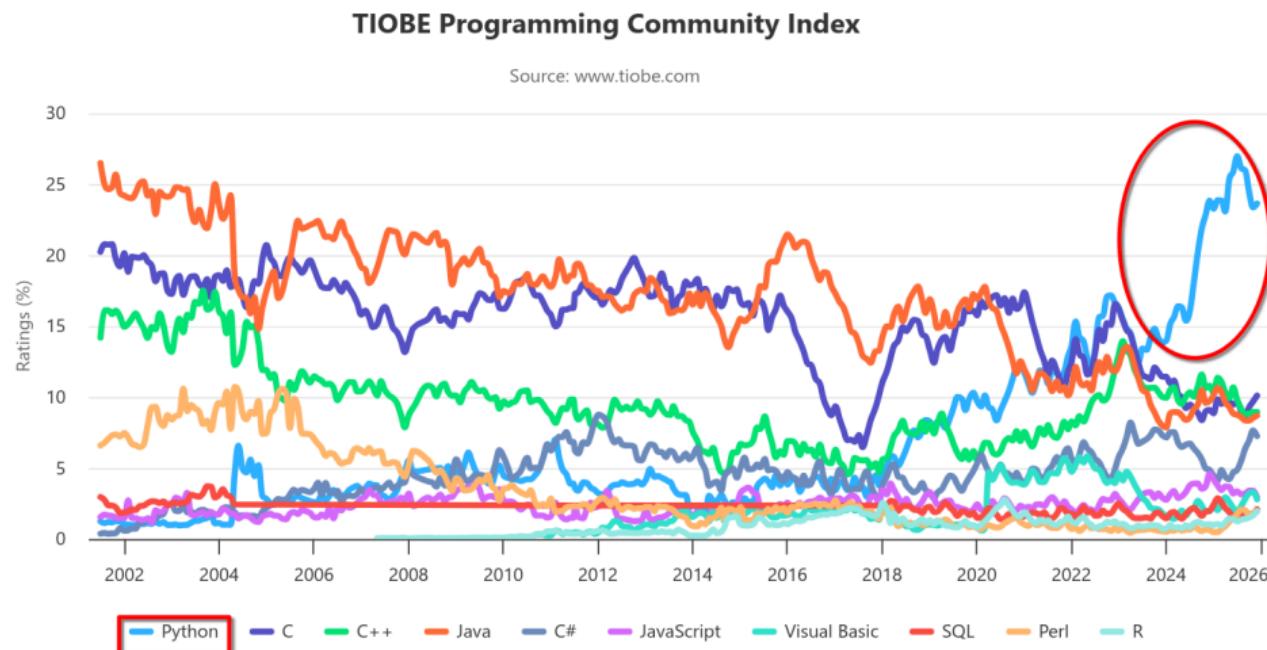
## Python popularity (2)

*“Which programming, scripting, and markup languages have you done extensive development work in over the past year?”*



Figure 2: Source: StackOverflow Developer Survey 2025

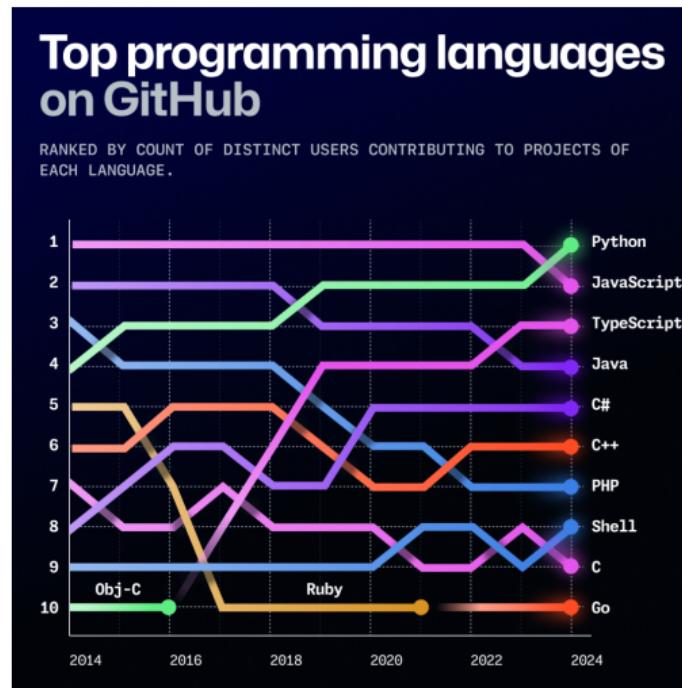
## Python popularity (3)



**Figure 3:** Source: TIOBE Index for December 2025

## Python popularity (4)

GitHub: “AI leads Python to top language as the number of global developers surges”



**Figure 4:** Source: [GitHub Blog](#), October 2024

# Python vs. other languages (1)

## Matlab

- Proprietary, quite expensive
- Shipped as a complete software package from one vendor (plus optional toolboxes)
- (Legacy) industry standard, widely used
- Substantially less powerful syntax
- Pure Matlab is somewhat faster than pure Python, but Python is easier to accelerate

## Julia

- Free, open source
- Focused on numerical computation, not on general-purpose computing
- Substantially faster than Python, but Python can be accelerated to similar speeds (using Numba)
- Popular among younger academics doing quantitative work
- Smaller ecosystem & less mature
- Not widely used or supported by Big Tech

## Python vs. other languages (2)

### R language

- Free, open source
- Focus on statistics, not on general-purpose computing
- Large ecosystem of packages for statistics, econometric modeling, and machine learning

### Stata

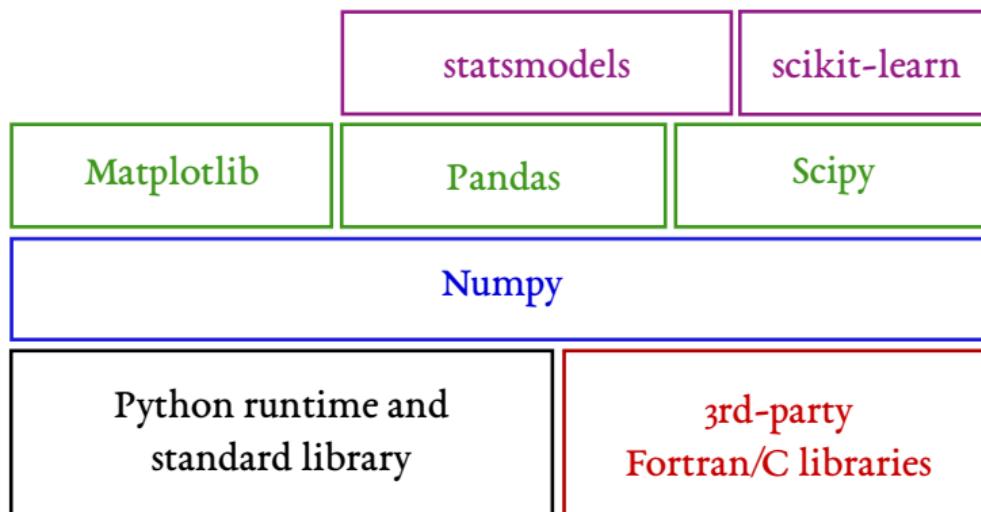
- Proprietary, quite expensive
- Focus on econometrics, in particular econometrics using large micro datasets
- Syntax was designed to run built-in commands, very inflexible for anything else
- If what you need is implemented, great! If not, it's very tedious to do it yourself  
(Mata is not great either)

# PYTHON ECOSYSTEM

# Python software stack

How things fit together

- “Python” is the language & standard library supported by the [Python Software Foundation](#)
- For numerical applications, we need additional third-party packages such as [NumPy](#), [SciPy](#), etc.



# Python software stack (used in this course)

## Core libraries for quantitative work

- [Python](#) language, runtime and standard libraries (“Python”)
- [NumPy](#): implements  $n$ -dimensional arrays, linear algebra routines, random number generators
- [Matplotlib](#): High-level plotting routines for visualization
- [Pandas](#): Containers to handle heterogeneous data & routines for data analysis
- [SciPy](#): Optimization routines, sparse matrices, integration, interpolation, linear algebra, statistics
- [scikit-learn](#): routines used for machine learning (Ridge regression, Lasso, elastic net, etc.)

# Python software stack (**not** covered in this course)

## Econometrics & Machine learning

- [statsmodels](#): routines for estimating (linear) models
- [TensorFlow](#): ML library maintained by Google with Python API
- [JAX](#): Low-level API for automatic differentiation and accelerated linear algebra used to build ML models, developed by Google
- [PyTorch](#): Python interface to ML libraries originally developed by Facebook

## Frameworks to speed things up

- [Numba](#): compiles Python code to machine code using LLVM
- [Cython](#): converts pseudo-Python to C code (advanced, don't use this)

# Jupyter notebooks vs. Python files

This course often uses Jupyter notebooks, not “regular” Python scripts.

## Jupyter notebooks

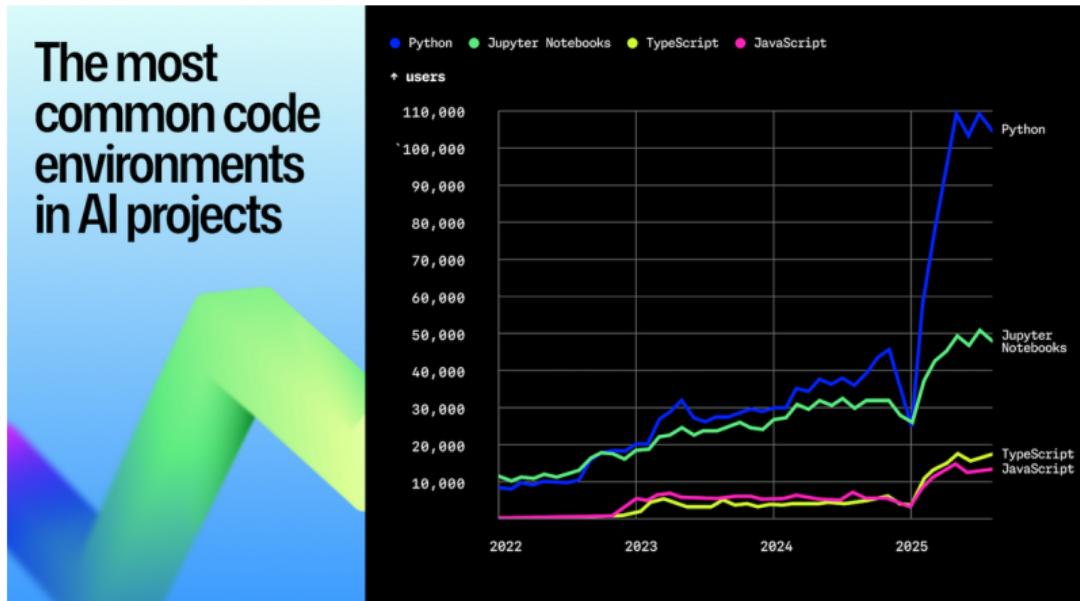
- File extension: .ipynb
- Interactive, dynamic notebooks
- Good for exploratory work
- Easy to share work with others, in particular if they are *not* data analysts or programmers
- Can be exported to other formats, e.g., PDFs, L<sup>A</sup>T<sub>E</sub>X

## Python scripts

- File extension: .py
- Interactive only in debugger
- For “serious” programming
- For libraries, reusable code
- Not useful to share with others who don’t know Python

# Jupyter notebooks

Explosive growth of Jupyter notebooks on GitHub that are used for data science, data visualization, and AI.



**Figure 5:** Source: [GitHub Blog](#), October 2025

## SOFTWARE & TOOLS

## Software & tools

**Goal:** learn to use industry-standard tools for programming in Python

- Python distribution: [Anaconda](#)
- Version control: [git](#)
- Code hosting: [GitHub](#)
- Editor: [Visual Studio Code](#)
- AI-assisted coding: [GitHub Copilot](#) and possibly other tools (e.g., [Gemini Code Assist](#), [Gemini CLI](#))

## Why git? (and GitHub)

- Because everyone uses it: almost completely replaced all other version control systems over the last 19 years

Examples:

- Python: <https://github.com/python/cpython>
- NumPy: <https://github.com/numpy/numpy>
- SciPy: <https://github.com/scipy/scipy>
- Pandas: <https://github.com/pandas-dev/pandas>
- Matplotlib: <https://github.com/matplotlib/matplotlib>
- PyTorch (Meta's ML library): <https://github.com/pytorch/pytorch>
- TensorFlow (Google's ML library): <https://github.com/tensorflow/tensorflow>

- Keeps history of **your** code changes (and can restore previous versions)
- Keeps history of **others'** code changes
- Allows for decentralized coding in teams
- Allows synchronizing of code across devices

## Why GitHub?

- Everyone uses it!
- Alternatives (less popular):
  - GitLab
  - BitBucket
- Offers many other services besides version control (issue tracking, Wiki, etc.)
- Register for free at <https://github.com/signup>
- Students can sign up for GitHub Pro for free (comes with higher GitHub Copilot usage quotas)

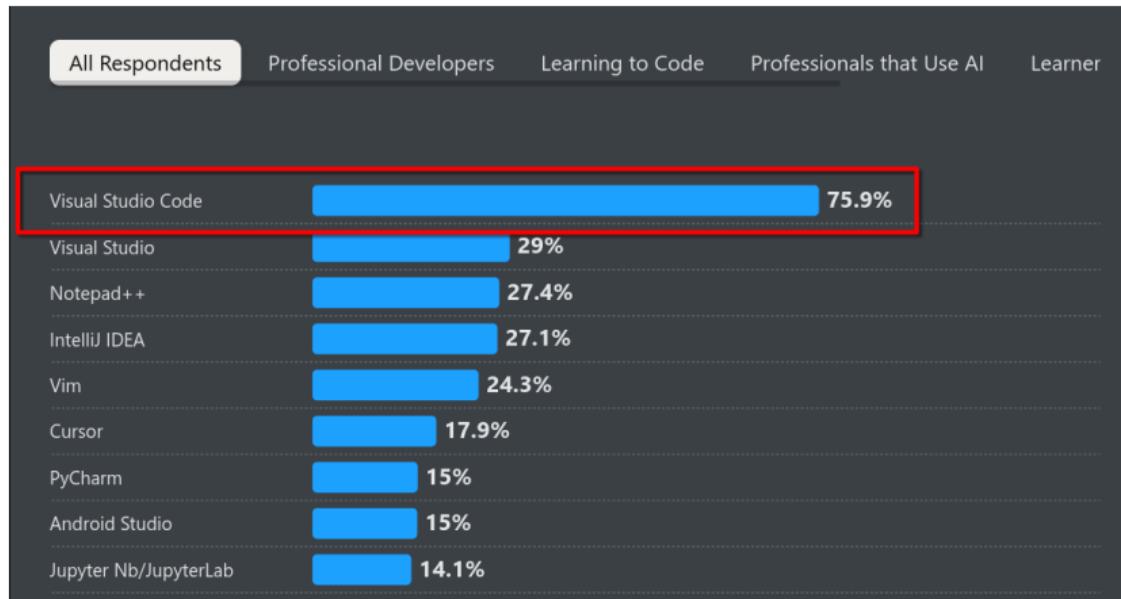
# Visual Studio Code

## Why Visual Studio Code?

- Has become the most widely used editor for most languages (see [StackOverflow Developer Survey 2025](#))
- Free & open source
- Good support for almost any programming language and file format (e.g., Jupyter Notebooks) via extensions
- Natively supports git & GitHub (unlike older editors)
- Natively supports AI integration (GitHub Copilot)
- Almost any other recent "AI editor" (Cursor, Windsurf, Google Antigravity) is a clone of VS Code
- Alternative: PyCharm by JetBrains (free community edition is available, free professional edition for students)
- Note: [Visual Studio Code](#) completely independent of [Visual Studio](#), a commercial IDE from Microsoft for Windows development

# VS Code is the most popular editor

*“Which development environments and AI-enabled code editing tools did you use regularly over the past year?”*



**Figure 6:** Source: StackOverflow Developer Survey 2025

## COURSE OUTLINE & ASSESSMENT

# Course details

## Teaching approach

- 1 Lectures: introduce new concepts [Tuesday, 12:15–14:00]
- 2 Workshops: practice concepts from previous lecture [Thursday, 8:30–10:00]

## Prerequisites

- No Python knowledge required
- Previous exposure to other programming languages (R, Julia, Matlab) is helpful

## Course material

- Available on GitHub: <https://github.com/richardfoltyn/FIE463-V26>

# Part 1 – Introduction to Python

Teaching weeks 3–6, January 20 – February 12

## Contents

- Setting up a working environment
- Working with Visual Studio Code, Jupyter notebooks, git
- Basic programming concepts (syntax, data types, NumPy arrays)
- Control flow (conditional execution, loops)
- Functions and modules
- Random number generation
- Plotting with matplotlib
- Applications:
  - Maximization using grid search
  - Portfolio choice
  - Consumption-savings problems (partial equilibrium)

# Part 2 – Applications to models in macroeconomics & finance

**Teaching weeks 7–10, February 17 – March 12**

## Contents

- Advanced NumPy and SciPy
- Maximization and root-finding using standard algorithms
- Applications:
  - Labor supply & consumption-savings problems (including general equilibrium)
  - Portfolio choice
  - Overlapping-generations models (OLG)
  - Stochastic processes and simulation
  - Solving models with uncertainty

## Part 3 – Working with financial data

**Teaching weeks 11–14, March 17 – April 9**

- Introduction to pandas
- Processing data from various sources
- Introduction to unsupervised and supervised learning with scikit-learn
- Applications:
  - Obtaining macroeconomic & financial data from the internet
  - Predicting house prices, stock prices, or similar

# Courses using Python at NHH

- 1 [FOR14](#) (BØA): Algorithms and Computer Programming with Python
  - 2 [BAN401](#): Applied Programming and Data Analysis for Business
  - 3 [BAN405](#): Python Programming for Data Science
  - 4 [BAN436](#): Introduction to Python (1 week) [discontinued]
  - 5 [BAN438](#): Application Development in Python [discontinued]
  - 6 [BAN442](#): From data to value: Machine Learning with Python (1 week)
- 
- FIE463 overlaps with these in the first few weeks (intro to Python)
  - Parts 2 and 3 focus on applications in macroeconomics & finance, unlike the courses above

## Course approval

- **Individual programming assignment**
- Grading: Pass/Fail — Pass required to receive a grade in this course
- Hand-out date: Thursday, February 12 at 9:00
- Submission date: Wednesday, February 18 at 12:00

# Assessment

**1 Group project #1 (“Term paper”) [40%]**

- Hand-out date: Thursday, March 12 at 9:00
- Submission date: Wednesday, March 18 at 12:00

**2 Individual peer review of another group’s project #1 [5%]**

- Hand-out date: Wednesday, March 18 at 15:00
- Submission date: Tuesday, March 24 at 12:00

**3 Group project #2 (“Term paper”) [50%]**

- Hand-out date: Thursday, April 9 at 9:00
- Submission date: Wednesday, April 22 at 12:00

**4 Individual peer review of another group’s project #2 [5%]**

- Hand-out date: Wednesday, April 22 at 15:00
- Submission date: Tuesday, April 28 at 12:00

- All 4 grade components must receive a passing grade to pass the course
- Term papers are written in groups of 2–3. They cannot be done individually without good reason and prior approval

# AI guidelines

**Goal:** Use AI as a tool, not a crutch. You need to understand code to verify it.

## Rules for this course

- **Part 1:** AI use is **discouraged**. Focus on building your own understanding.
- **Allowed:**
  - Coding assistant in lectures and workshops
  - Group projects and peer reviews (usage **must** be documented)
- **Forbidden:**
  - Handing in AI-generated output directly (copy/paste)
  - All text and analysis must be your own work

*Note: We will have a dedicated workshop on GitHub Copilot (and possibly Gemini CLI/Gemini Code Assist) in Part 2.*

# Goal for today

Everyone should have a working environment up and running.

## Steps:

- 1 Download and install Anaconda
- 2 Download and install git
- 3 Download and install Visual Studio Code
- 4 Create the FIE463 environment in the Terminal (Windows: Anaconda Prompt):  
`conda create -f environment.yml`
- 5 Clone the course repository from GitHub in VS Code:  
<https://github.com/richardfolty/FIE463-V26.git>
- 6 Run the notebook  
`lectures/lecture00/test-environment.ipynb`  
to verify that everything is working

## ADDITIONAL RESOURCES

# Additional resources – Books

## Books

- [Think Python](#) by Allen B. Downey  
General intro to Python, chapters are available as Jupyter notebooks.
- [Python for Everybody](#) by Charles R. Severance  
General intro to Python with a focus on data analysis, available as PDF.

## Online courses

- [QuantEcon lectures](#)  
Python programming for economics & finance
- [Introduction to Programming and Numerical Analysis](#)  
Python course at the University of Copenhagen, focusing on applications in macroeconomics

# Additional resources

## User guides and documentation

- [Numpy quick start tutorial](#)
- [Numpy tutorial for Matlab users](#)
- [scikit-learn user guide](#)

## Code

- [QuantEcon library](#)

Collection of routines and tools for economics

- [QuantEcon repository](#)

Contributed code for solving economic problems in Python

## Additional resources – Videos

### Introduction to the command line / terminal:

- Absolute BEGINNER Guide to the **Mac OS** Terminal [17 min]  
<https://youtu.be/aKRYQsKR46I>
- Git Bash - Simplest command line program for **Windows** [7 min]  
<https://youtu.be/yoZ910JQzrg>

### Introduction to using git

- Git for dummies [20 min] <https://youtu.be/mJ-qvsxPHpY>
- Git and GitHub Tutorial for Beginners [46 min] <https://youtu.be/tRZGeaHPoaw>
- Git Essentials in VS Code [30 min] <https://youtu.be/twsYxYaQikI>  
Focuses on interacting with git and GitHub through VS Code