

# Introduction

FIE463: Numerical Methods in Macroeconomics and Finance using Python

Richard Foltyn

NHH Norwegian School of Economics

January 9, 2025

- 1 Introduction to Python
  - Why Python?
  - Examples of Python in economics & finance
  - Python vs. other languages
  - Python ecosystem
- 2 Course outline & assessment
  - Course outline
  - Assessment
- 3 Software & tools
- 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:
  - Previously (and mostly forgotten): C/C++, Visual Basic, Java, Java Script, 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)
- May not be the fastest, but offers easy way to accelerate things (Cython, Numba, JAX, ML libraries)

## What can you do with Python?

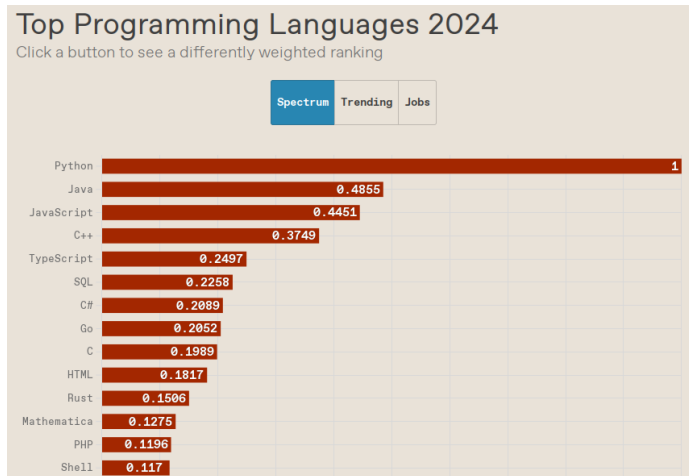
- Everything. The question is whether you should be using Python ...

## Why not Python?

- You already know another language that solves your problem well
- You want to use an estimator/algorithm that is implemented somewhere else (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 2024](#)

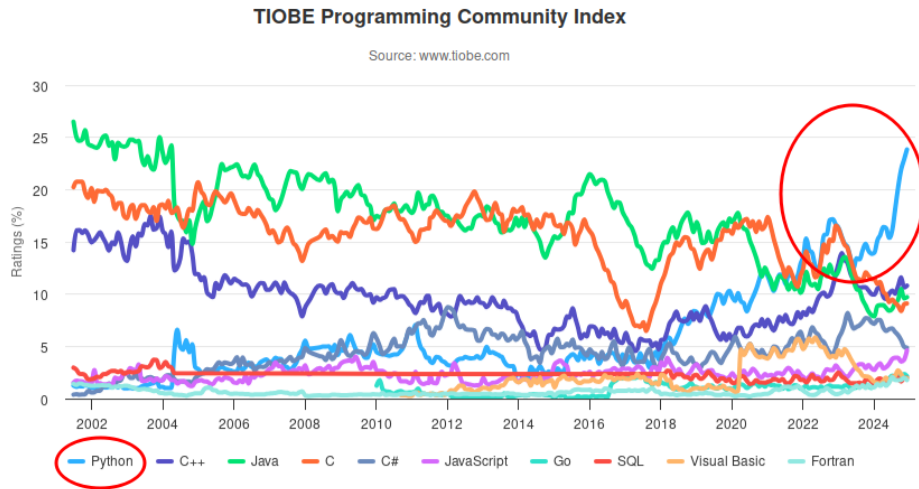
## Python popularity (2)

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



**Figure 2:** Source: [StackOverflow Developer Survey 2024](#)

## Python popularity (3)



**Figure 3:** Source: TIOBE Index for December 2024



## Python popularity (4)

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



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

# Examples of Python in economics & finance

## **Solving dynamic programming problems with Python + Numba**

- Olsson (2025): Solves Aiyagari model + extensive margin labour supply choice for single and couple households
- Foltyn (2024): Household finance model with portfolio choice and learning from experience

## **Econometrics (custom estimators with Python + Numba or JAX)**

- Foltyn and Olsson (2024): Custom Maximum Likelihood Estimator (MLE)
- Foltyn (2024): Custom MLE, uses JAX to run it on GPUs

## **Dynamic economic models solved with Python + ML**

- Maliar, Maliar, and Winant (2021): Solve dynamic problems with TensorFlow; [\[code\]](#)
- Duarte, Duarte, and Silva (2024): Continuous-time finance models with TensorFlow
- Duarte et al. (2021): Solve HH portfolio choice problem with 22 states using JAX

# Python vs. other languages (1)

## Matlab

- Proprietary, quite expensive
- Shipped as 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, less on general-purpose computing
- Substantially faster than Python, but Python can be accelerated to similar speed (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, less on general-purpose computing
- Large ecosystem of packages for statistics, econometric modelling, and machine learning

### **Stata**

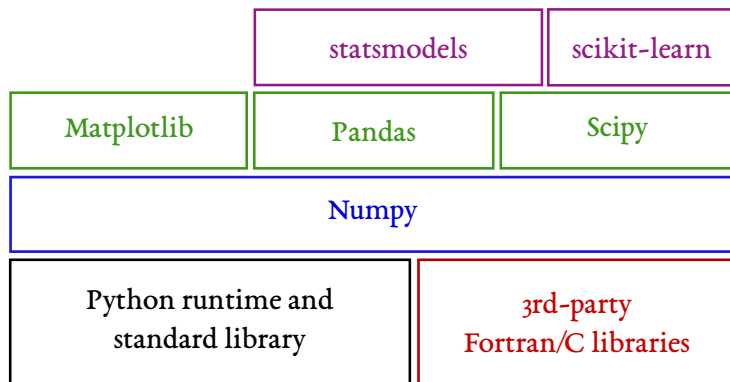
- Proprietary, quite expensive
- Focus on econometrics, in particular econometrics using large micro data sets
- Syntax was designed to run built-in commands, very inflexible for anything else
- If what you need is implemented, great! If not, it's very tedious to do it yourself (Mata is not great either).

# 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 3<sup>rd</sup>-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,  $\text{\LaTeX}$

## 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 vs. Python files

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

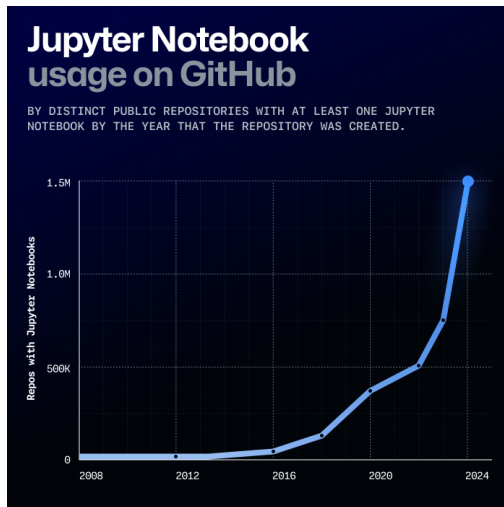


Figure 5: Source: [GitHub Blog](#), October 2024

# COURSE OUTLINE & ASSESSMENT

## Teaching approach

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

## 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-V25>

# Course outline (preliminary!)

## Part 1: Introduction to Python [ $\approx$ 4 weeks]

- Setting up a working environment
- Working with Visual Studio Code, Jupyter notebooks, git (optional)
- 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 [ $\approx$ 4 weeks]**

- 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 (income risk)
  - Simple asset pricing models

## **Part 3: Working with financial data [ $\approx$ 4 weeks]**

- 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](#): Algorithms and Computer Programming with Python
  - 2 [BAN401](#): Applied Programming and Data Analysis for Business
  - 3 [BAN436](#): Introduction to Python (1 week)
  - 4 [BAN438](#): Application Development in Python
  - 5 [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 6 at 9:00
- Submission date: Wednesday, February 12 at 12:00

- 1 **Group project #1** (“term paper”) [40%]
  - Hand-out date: Thursday, March 6 at 9:00
  - Submission date: Tuesday, March 18 at 12:00
- 2 **Individual peer review** of another group’s project #1 [5%]
  - Hand-out date: Tuesday, March 18 at 15:00
  - Submission date: Wednesday, March 26 at 12:00
- 3 **Group project #2** (“term paper”) [50%]
  - Hand-out date: Wednesday, April 2 at 9:00
  - Submission date: Wednesday, April 16 at 12:00
- 4 **Individual peer review** of another group’s project #2 [5%]
  - Hand-out date: Wednesday, April 16 at 15:00
  - Submission date: Wednesday, April 30 at 12:00

The peer reviews are intended to give you additional feedback on code style, structure and efficiency in a respectful, constructive manner.

TOOLS:  
GIT, GITHUB, AND VS CODE

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

- Python distribution: Anaconda
- Version control: git
- Code hosting: GitHub
- Editor: Visual Studio Code

## 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 **other's** code changes
- Allows for decentralized coding in teams
- Allows synchronizing of code across devices

## Why GitHub?

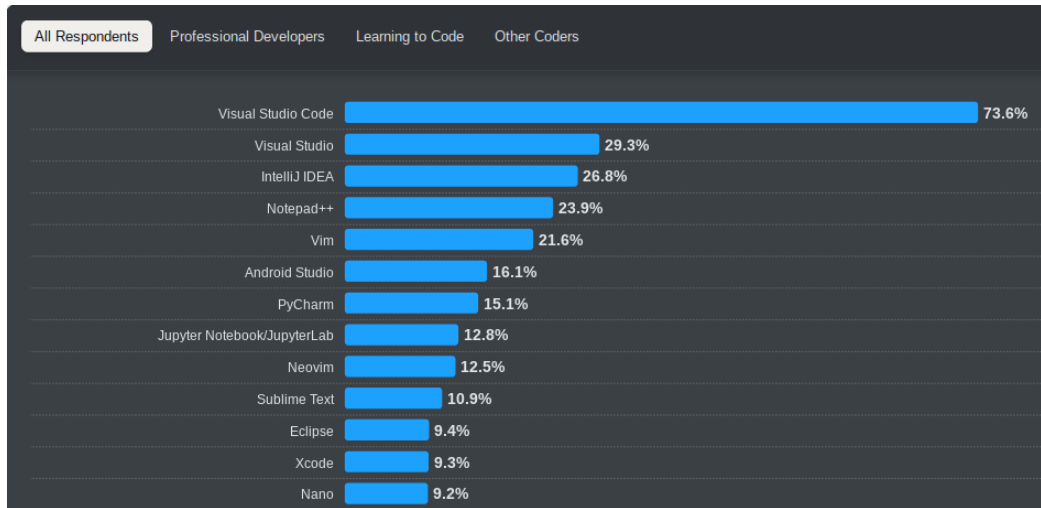
- Everyone uses it!
- Alternatives (less popular):
  - [GitLab](#)
  - [BitBucket](#)
- Offers many other services besides version control (issue tracking, Wiki, etc.)
- Register for free at <https://github.com/signup>

## Why Visual Studio Code?

- Has become the most widely used editor for most languages (see [StackOverflow Developer Survey 2024](#))
- 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)
- 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 did you use regularly over the past year?”*



**Figure 6:** Source: [StackOverflow Developer Survey 2024](#)



## 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

### 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

# References

- Duarte, Victor, Diogo Duarte, and Dejanir H Silva. 2024. Machine learning for continuous-time finance. **The Review of Financial Studies** 37 (11): 3217–3271.
- Duarte, Victor, Julia Fonseca, Aaron S Goodman, and Jonathan A Parker. 2021. **Simple Allocation Rules and Optimal Portfolio Choice Over the Lifecycle**. Working Paper, Working Paper Series 29559. National Bureau of Economic Research.
- Foltyn, Richard. 2024. Experience-based Learning, Stock Market Participation and Portfolio Choice.
- Foltyn, Richard, and Jonna Olsson. 2024. Subjective life expectancies, time preference heterogeneity, and wealth inequality. **Quantitative Economics** 15 (3): 699–736.
- Maliar, Lilia, Serguei Maliar, and Pablo Winant. 2021. Deep learning for solving dynamic economic models. **Journal of Monetary Economics** 122:76–101.
- Olsson, Jonna. 2025. Singles, couples, and their labor supply: long-run trends and short-run fluctuations. **American Economic Journal: Macroeconomics** 17 (1): 1–34.