

# Intro & Course Outline

## Introduction to Python Programming for Economics & Finance

Richard Foltyn

University of Glasgow

May 31, 2023

- 1 Introduction to Python
  - Why Python?
  - Examples of Python in economics
  - Python vs. other languages
  - Python ecosystem
- 2 Course outline
- 3 Course material & additional resources

# 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)

# 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 reasonably well
- You want to use an estimator/algorithm that is implemented somewhere else (Stata, R), but not in Python

# Examples of Python in economics

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

- Olsson (2023): Solves Aiyagari model + extensive margin labour supply choice for single and couple households [accepted at AEJ:Macro]
- Foltyn (2020): Household finance model with portfolio choice and learning from experience

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

- Foltyn and Olsson (2023): Implements MLE that keeps track of latent health states [R&R Quant. Econ.]

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

- Maliar, Maliar, and Winant (2021): Solve dynamic problems with TensorFlow; example code [here](#) [JME, 2021]
- Duarte (2018): Continuous-time finance models with TensorFlow [R&R Review of Financial Studies]
- Duarte et al. (2021): Solve HH portfolio choice problem with 22 states using JAX

# Comparing to other languages (1)

## Matlab

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

# Comparing to other languages (1)

## Matlab

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

## R language

- Free, open source
- Focus on statistics, less on general-purpose computing
- Large ecosystem of packages focus on statistics, econometric modelling, machine learning

## Comparing to other languages (2)

### Julia

- Free, open source
- Focused on numerical computations, less on general-purpose computing
- Substantially faster than Python, but Python can be accelerated to similar speed (using Numba)
- Smaller ecosystem, still under rapid development



## Comparing to other languages (2)

### Julia

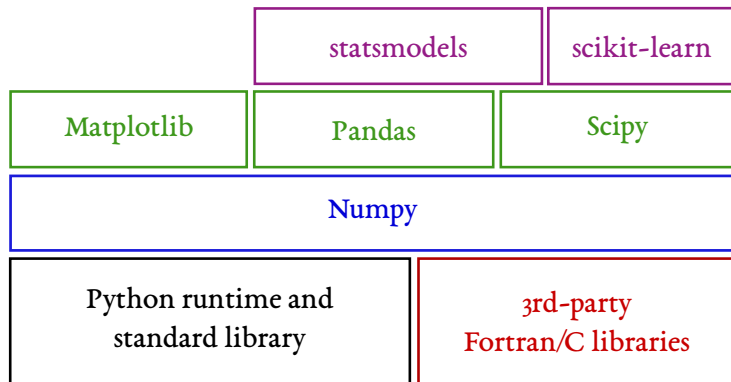
- Free, open source
- Focused on numerical computations, less on general-purpose computing
- Substantially faster than Python, but Python can be accelerated to similar speed (using Numba)
- Smaller ecosystem, still under rapid development

### Stata

- Proprietary, quite expensive
- Focused 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 software stack

How things fit together



# Python software stack (covered 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
- **SciPy**: Optimisation routines, sparse matrices, integration, interpolation, linear algebra, statistics
- **Matplotlib**: High-level plotting routines for visualisation
- **Pandas**: Containers to handle heterogeneous data & routines for data analysis
- **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 many (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

This course is mostly based on Jupyter notebooks, not “regular” Python scripts.

## Jupyter notebooks

- File extension: `.ipynb`
- Interactive, dynamic notebooks
- Run in web server, displayed in web browser
- 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
- Usually run locally in Python interpreter
- For “serious” programming
- For libraries, reusable code
- Not useful to share with others who don’t know Python

# COURSE OUTLINE

## ■ **Lecture 1**, 9:00–12:15 (15 min break), Room 305AB

### Unit 1: **Language and NumPy basics** [[PDF](#)]

- 1 Basic syntax
- 2 Built-in data types
- 3 NumPy arrays
- 4 Optional exercises with provided solutions (asynchronous)

### Unit 2: **Control flow and list comprehensions** [[PDF](#)]

- 1 Conditional execution
- 2 Loops
- 3 List comprehensions
- 4 Optional exercises with provided solutions (asynchronous)

### Unit 3: **Reusing code – Functions, modules and packages** [[PDF](#)]

- 1 Functions
- 2 Modules and packages
- 3 Optional exercises with provided solutions (asynchronous)

## ■ **Lab 1**, 13:30–15:00, Room 305AB

- Lab exercise for units 1–3

## ■ **Lecture 2**, 9:00–12:15 (15 min break), Room 305AB

### Unit 4: **Plotting** [[PDF](#)]

- 1 Line and scatter plots, categorical data
- 2 Labels and annotations
- 3 Multiple plots
- 4 Optional exercises with provided solutions (asynchronous)

### Unit 5: **Advanced NumPy** [[PDF](#)]

- 1 Creating and reshaping arrays
- 2 Advanced indexing
- 3 Numerical operations
- 4 Optional exercises with provided solutions (asynchronous)

## ■ **Lab 2**, 13:30–15:00, Room 305AB

- Lab exercise for units 4–5



■ **Lecture 3**, 9:00–12:15 (15 min break), Room 305AB

Unit 6: **Handling data with pandas** [[PDF](#)]

- 1 Creating and viewing DataFrames
- 2 Indexing
- 3 Aggregation and reduction operations
- 4 Working with time series data
- 5 Visualisation
- 6 Optional exercises with provided solutions (asynchronous)

Unit 7: **Data input and output** [[PDF](#)]

- 1 I/O with NumPy
- 2 I/O with pandas
- 3 Retrieving macroeconomic / financial data from the web

■ **Lab 3**, 13:30–15:00, Room 305AB

- Lab exercise for units 6–7

## ■ **Lecture 4**, 9:00–12:15 (15 min break), Room 305AB

### Unit 8: **Random number generation and statistics** [[PDF](#)]

- 1 NumPy's RNG routines
- 2 Statistics functions in SciPy
- 3 Optional exercises with provided solutions (asynchronous)

### Unit 9: **Introduction to unsupervised learning** [[PDF](#)]

- 1 Principal component analysis (PCA)
- 2 Introduction to scikit-learn

### Unit 10: **Introduction to supervised learning** [[PDF](#)]

- 1 Linear regression models
- 2 Ridge regression
- 3 Lasso
- 4 Hyperparameter tuning

## ■ **Lab 4**, 13:30–15:00, Room 305AB

- Lab exercise for units 8–10

- **Lecture 5**, 9:00–12:15 (15 min break), Room 305AB

Unit 11: **Solving models for macroeconomics and household finance** [[PDF](#)]

- 1 Setting up consumption/savings household problems
- 2 Solving deterministic problems with VFI
- 3 Solving stochastic problems with VFI

- **Lab 5**, 13:30–15:00, Room 305AB

- Lab exercise for unit 11

## COURSE MATERIAL & ADDITIONAL RESOURCES

- All code can be downloaded from GitHub repository  
<https://github.com/richardfoltyn/python-intro-PGR>
- Interactive notebooks can be launched directly in the browser (see [setup guide](#))

## Additional resources

- Scipy Lecture Notes (<http://scipy-lectures.org/index.html>)  
Online lecture notes for quantitative work with Python
- Numpy quick start tutorial (<https://numpy.org/doc/stable/user/quickstart.html>)
- Numpy tutorial for Matlab users  
(<https://numpy.org/doc/stable/user/numpy-for-matlab-users.html>)
- QuantEcon lectures: mostly Python but also Julia (<https://quantecon.org/lectures/>)
- QuantEcon library for Python (<https://quantecon.org/quantecon-py/>)  
Collection of routines and tools for economics
- QuantEcon repository for contributed code solving economic problems in Python  
(<https://notes.quantecon.org/>)
- Dive Into Python 3 (<https://diveintopython3.net/>)  
Freely available online book on general Python programming (no focus on scientific computing)
- scikit-learn user guide ([https://scikit-learn.org/stable/user\\_guide.html](https://scikit-learn.org/stable/user_guide.html))

# References

- Duarte, Victor. 2018. **Machine learning for continuous-time finance**. Technical report. Tech. rep., Gies College of Business.
- 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. 2020. Experience-based Learning, Stock Market Participation and Portfolio Choice. **Available at SSRN 3543442**.
- Foltyn, Richard, and Jonna Olsson. 2023. **Subjective Life Expectancies, Time Preference Heterogeneity, and Wealth Inequality**. Technical report.
- Maliar, Lilia, Serguei Maliar, and Pablo Winant. 2021. Deep learning for solving dynamic economic models. **Journal of Monetary Economics** 122:76–101.
- Olsson, Jonna. 2023. Singles, couples, and their labor supply: long-run trends and short-run fluctuations. **American Economic Journal: Macroeconomics**.