BA Course "6,222 Introduction to Programming"

Dr. Franziska Bender, Dr. Aurélien Sallin

20.02.2026

Overview

Learning Objectives: After completion of the course, students will be able to design and execute projects in Economics and Econometrics in a systematic and reproducible way. The course has two core objectives: to develop intermediate programming skills and to apply collaborative, reproducible code development practices. First, students will understand and apply essential programming practices in Python, including writing modular code, documenting functions and implementing error handling. They will also be able to use object oriented programming by creating their own classes to structure larger projects. Second, students will be able to collaborate on projects using Git and version control, manage shared code and work with branching and pull requests. They will know how to design relational database schemas and interact with relational data from Python. They will also understand central concepts of DevOps, including continuous integration workflows and automated testing. Throughout the course they will learn to produce code that is reusable, maintainable and easy to share.

Philosophy of the course: The course aims to teach and coach young economists to code and work in a collaborative, structured and impactful way. The focus is on both traditional teaching and practical work, with guided exercises and project coaching in settings that reflect real work in research and industry.

Target audience: The course is mandatory for students in the Bachelor program in Economics and is offered in the second semester. Typically, participants have completed Data Handling and Statistics in the previous semester and are taking Introductory Econometrics in parallel. They already have solid working knowledge of R, including basic data manipulation and exploratory analysis.

Structure of the course:
- 12 units of 90′ for the main lecture;
- 10 units of 90′ for the exercises (week 1, 2, 3, 4, 6, 8, 9, 10, 11, 12);

Prerequisites:
- Elementary programming skills (see syllabus “Data Handling” course)
- Knowledge of basic Data structures (see syllabus “Data Handling” course)
- Statistics

Examination overview

The student assessment consists of two main components:

1. **Final Exam (30% of the grade)**: This exam will cover the core theoretical concepts of the course. Form: digital exam, BYOD, duration tbd. Multiple-choice questions.
2. **Group Project (70% of the grade)**: This is the primary applied assessment. Students will work in groups of 5 to design and build a 'software tool' that applies the skills learned in the course.

Tutorials

- The course includes 10 tutorial sessions to accompany the lectures.
- A team of 4 Teaching Assistants (TAs) working on each exercise session.
- First 8 tutorials: (Skill Building)
  - These are classic, hands-on sessions for students to practice the concepts from the lectures.
  - They will be held as a single, combined session for all students. Or
  - 1 TA will lead the instruction, while the other 3 TAs provide hands-on support and answer individual questions.
- Final 2 tutorials: (Project Support)
  - These sessions are structured as "office hours" for the group projects.
  - Purpose: This time is dedicated for groups to meet with TAs to get help, troubleshoot problems, and receive direct guidance on their project.

Detailed Lecture Plan

Lecture 1	The big picture. Set up your environment.	Aurélien & Franziska	20.02.2026
Lecture 2	Working together: intro to version control and git	Aurélien	27.02.2026
Lecture 3	Working together: more about version control and git	Aurélien	06.03.2026
Lecture 4	Intro to python	Franziska	13.03.2026
Lecture 5	Python for Data: Pandas and Matplotlib	Franziska	20.03.2026
Lecture 6	Python for Data: Pandas and Matplotlib	Franziska	27.03.2026
Break			
Lecture 7	Python: Introduction to Classes (OOP)	Franziska	17.04.2026
Lecture 8	Error Handling, documentation, Debugging	Franziska	24.04.2026
Lecture 9	Short overview of databases and relational database management systems	Aurélien	01.05.2026
Lecture 10	Devops, Continuous Integration	Aurélien	08.05.2026
Lecture 11	Conclusion, Q&A,Case Study	Aurélien & Franziska	15.05.2026
Lecture 12	Exam	Aurélien & Franziska	22.05.2026

Lecture 1: The Big Picture. Set up your environment *F, A*

Lecture 1 defines the objectives of the course and motivates why programming in a shareable, reproducible way is crucial for Economists. The first part presents the course structure, the team, the examination format, and the group project expectations.

The second part (01b) introduces the tools students will use throughout the course. Students learn what a terminal is, how to navigate the file system using the command line (bash on Mac, Command Prompt on Windows), and how to work with files and paths. Students are introduced to Visual Studio Code as their IDE, and to Python as the main programming language. The lecture explains why Python projects require isolated virtual environments (unlike R's global library approach) and introduces uv as the tool for managing Python versions, packages, and project environments.

Exercise week 1: Students set up their course directory structure, install VS Code and extensions, install uv, create their first Python project with a virtual environment, and run their first Python script. They also install git for the next lecture.

Sources:

- "Research Software Engineering", Matt Bannert
- "Basics of Computing Environments for Scientists", Physics department of ETH Zurich

Lecture 2-3: Working Together: Intro to Version Control and Git *A*

The lecture introduces version control as a central element of collaborative data work. It covers the main Git concepts such as repositories, commits, staging, branching, and merging. Students learn how to create and configure a local repository and how to connect a project to GitHub. The session discusses branching strategies for small teams, the use of pull requests, and practical conflict resolution. It also addresses project organization with clear folder structures and naming conventions. It then covers Github and how to work with a remote repository.

The lecture includes a hands on part in which students create a repository, push and pull changes, work with branches, merge them and resolve conflicts.

Exercise weeks 2-3: Students will have time to experiment with git and github.

Sources:

- "Introduction to git" from Jam Simson
- "Git Version Control", "Research Software Engineering", Matt Bannert

Lecture 4: Intro to Python *F*

Lecture 4 provides the essential foundation for the course. This lecture will cover Python's core syntax, its fundamental data types (like strings and integers), and data structures (like lists, dictionaries, …). We will also walk through control flow (loops and conditionals) and demonstrate how to write functions. Throughout the lecture, we will highlight the differences between R and Python (such as 0-indexing and indentation) to help students adapt their existing programming knowledge.

Exercise week 4: Students apply basic concepts in Python learned in the course.

Lectures 5 and 6: Python for Data: Pandas and Matplotlib *F*

This lecture introduces students to Pandas, along with some NumPy essentials. We will cover the workflow for data handling: reading data into a DataFrame, inspecting its structure, and managing missing data. Students will then learn the core tools for data manipulation, including filtering rows, grouping data, aggregating results, and merging different datasets. The goal is to show how to perform basic (descriptive) analysis using interesting, economics-related data. Finally, we will introduce basic visualization tools, demonstrating how to generate simple plots using matplotlib to visually inspect results.

Optional: Time permitting, we may also provide a brief outlook on modern high-performance alternatives like Polars or tensor-based frameworks like PyTorch.

Exercise week 6: Students apply classes and work with data in Python.

Lecture 7: Introduction to Classes (OOP) *F*

This lecture introduces students to the fundamental concepts of Object-Oriented Programming (OOP). We will explain how OOP is used to structure code by bundling data and functionality into reusable blueprints. Students will learn the complete process of how to define their own custom class, covering the essential components: writing the constructor, defining attributes, and creating methods. The goal is

to ensure students can confidently build and use their own custom objects in Python. After covering these basics, we will explore some practical examples to showcase what can be done with a class, helping to spark ideas for the final group projects.

Lecture 8: Error Handling, Documentation *F*

This lecture focuses on the critical skills needed to transform a basic script into a reliable and shareable tool. We will cover the principles of Error Handling, teaching students how to anticipate common problems (like bad data inputs) and provide clear, helpful feedback to the user. The second half focuses on Documentation, explaining how to write clear descriptions for functions and classes. The goal is to ensure that the tools built in the group projects are robust, easy to use, and understandable by others.

Lecture 9: Short overview of databases and relational database management systems *A*

This lecture introduces students to databases and relational database management systems. Using PostgreSQL, students learn what are database systems in order to manage and process large data sets. They will be introduced to creating, manipulating, and querying databases using SQL.

Exercise weeks 7-8: Students run simple sql queries and set up a basic relational database management system.

Sources:
- "Big Data Analytics" from Ulrich Matter
- "Databases", "Research Software Engineering", Matt Bannert

Lecture 10: DevOps *A*

This lecture introduces students to basic principles of DevOps, like containerization and docker, Continuous Integration - Continuous Development (CI/CD). As an example, we'll run a simple API in docker and access the database in a container with postgres, or we will use GitHub Actions to update the api automatically.

Exercise weeks 9-10: tbd.

Sources:
- Survey with a Database Backend
- "Github Action Demo", Matt Bannert

Lecture 11: Conclusion, Q&A, Case Study *F, A*

Lecture 11 is a practical, application-focused session that demonstrates how to combine several of the tools learned in the course. We will walk through the process of building a custom Python class from scratch. The goal of this class will be to solve a specific data-driven problem.

Lecture 12: Exam *F, A*

This lecture concludes the semester with the final exam.

Tutorials

The course features 10 tutorial sessions designed to bridge the gap between lecture theory and practical application. These are supported by a team of teaching assistants.

Tutorial Structure

- First 8 tutorials: (Skill Building)
  - These are classic, hands-on sessions for students to practice the concepts from the lectures.
  - They will be held as a single session for all students, where one TA will lead the instruction while the other TA's will provide hands-on support and answer individual questions
- Final 2 tutorials: (Project Support)
  - These sessions are structured as "office hours" for the group projects.
  - Purpose: This time is dedicated for groups to meet with TAs to get help, troubleshoot problems, and receive direct guidance on their project.

Examination

Examination Part 1: Group Project

Project Overview

The goal of this project is to move beyond simple scripts and build a reusable Python tool for economic analysis. Working in groups, students will identify an economic question, source a relevant dataset, and build a Python class that automates the cleaning, analysis, and visualization of that data.

Students won't just be submitting code; they will be submitting a collaborative Git repository and two professional Quarto (`.qmd`) documents that communicate their findings and explain how their tool works.

The target audience for the project is other economists who want to use the tool. The project should be designed with reusability and shareability in mind, following best practices for code organization, documentation, and version control.

The Deliverables

1. The Collaborative Repository Students' project must be hosted on Git. We expect to see a clean commit history that reflects contributions from all group members. The repository should be organized and include a .gitignore file to keep the environment clean.

2. The Python Class The data analysis tool you will create is a class in a `.py` file in the repository. It should have different methods to (i) clean the data, (ii) perform some analysis /calculations (iii) visualize results.

3. Report: The Economic Analysis This is students' "Research Note." It should briefly explain:

- What is the topic of interest/ the 'research question'?
- What data they use?
- What are the key findings?

It should contain examples of the results / figures that students created using their class. In this report students don't have to explain the class or use all the methods, it should be an example of an interesting question that they can answer using their class.

4. Documentation: The User Manual This is the "Technical Guide" for other economists who want to use students' code. It must include:

- An explanation of the class
- A "Quick Start" section showing how to import students' class and run a basic analysis in three lines of code.
- For every method:
  - An explanation of what the method does
  - A description of the arguments (inputs) and return values (outputs).
  - Example Code: A few lines of code that show an example of how to use the method.

Examination Part 2: Final Exam