Introduction

Programming for Economists

Jeppe Druedahl
Brigitte Hochmuth

Plan

1. Intended learning goals
2. Programming in practice
3. Infrastructure
4. Work-flow
5. Projects
6. Summing up

Intended learning goals

Intended learning goals

- **In a nutshell:**
  1. Summarize and visualize empirical data
     ⇒gain flexibility and automation compared to Excel
     ⇒build upon Descriptive Economics
     ⇒working with »big data«
  2. Numerically solve and simulate economic models
     ⇒less strict assumptions than with math and easy visualization
     ⇒improve your understanding of Micro and Macro
     ⇒ from model-user to *model-builder*

Intended learning goals

- **In a nutshell:**
  1. Summarize and visualize empirical data
     ⇒gain flexibility and automation compared to Excel
     ⇒build upon Descriptive Economics
     ⇒working with »big data«
  2. Numerically solve and simulate economic models
     ⇒less strict assumptions than with math and easy visualization
     ⇒improve your understanding of Micro and Macro
     ⇒ from model-user to *model-builder*
- **Tool:** *Python* - general purpose programming language (open source)
  1. Web development (not covered)
  2. Data science and machine learning (covered)
  3. Scientific computing and research (main focus)

Active learning

- **Active learning:** To learn programming you need to work on actual problems yourself
  - We give your a broad overview
  - We give you set of important tools
  - You have to experiment on your own
  - You have to be able to acquire new tools on your own (studies+work)

Active learning

- **Active learning:** To learn programming you need to work on actual problems yourself
  - We give your a broad overview
  - We give you set of important tools
  - You have to experiment on your own
  - You have to be able to acquire new tools on your own (studies+work)
- **Use copilot and LLMs such as ChatGPT**
  - From words to code
  - Less need to remember syntax
  - Answers often *easily verifiable* (if you can understand what it does...)

Active learning

- **Active learning:** To learn programming you need to work on actual problems yourself
  - We give your a broad overview
  - We give you set of important tools
  - You have to experiment on your own
  - You have to be able to acquire new tools on your own (studies+work)
- **Use copilot and LLMs such as ChatGPT**
  - From words to code
  - Less need to remember syntax
  - Answers often *easily verifiable* (if you can understand what it does...)
- **High level:** Few (if any) econ bachelor programs provide education on programming and numerical analysis on the level you will get

Scientific programming

- **Numerical analysis** is a ***complement*** not a substitute for **math**

Scientific programming

- **Numerical analysis** is a ***complement*** not a substitute for **math**
- **Three central steps:**
  1. mathematical problem $\rightarrow$ construct algorithm
  2. translate algorithm $\rightarrow$ write code
  3. run code $\rightarrow$ present results

Scientific programming

- **Numerical analysis** is a ***complement*** not a substitute for **math**
- **Three central steps:**
  1. mathematical problem $\rightarrow$ construct algorithm
  2. translate algorithm $\rightarrow$ write code
  3. run code $\rightarrow$ present results
- **The set of potential errors is infinite:**
  A good work-flow is very important
  1. Clear structure reduces the number of bugs
  2. Testing helps discovering bugs
  3. Documentation helps removing bugs

Scientific programming

- **Numerical analysis** is a ***complement*** not a substitute for **math**
- **Three central steps:**
  1. mathematical problem $\rightarrow$ construct algorithm
  2. translate algorithm $\rightarrow$ write code
  3. run code $\rightarrow$ present results
- **The set of potential errors is infinite:**
  A good work-flow is very important
  1. Clear structure reduces the number of bugs
  2. Testing helps discovering bugs
  3. Documentation helps removing bugs
- **Programming is more than writing code:** Structuring, testing, documenting and collaborating on code projects is a central aspect of this course

You and your teachers

- **Jeppe Druedahl**, Associate Professor
  web: sites.google.com/view/jeppe-druedahl/
  e-mail: jeppe.druedahl@econ.ku.dk

- **Brigitte Hochmuth**, Assistant Professor
  web: brigittehochmuth.net/
  e-mail: bhoc@econ.ku.dk

First time

- **Previous course:** Introduction to Programming and Numerical Analysis
  1. Slightly higher level
  2. Almost fully video-based (few physical lectures)
- **First time for this course**
  1. *All of your feedback is very important for improving the course!*
  2. From next year the course is obligatory on the 3rd semester
  3. Goal is to be more video-based in future

Programming in practice

Python distribution and VSCode

- We work with **Python 3.13**
- **Suggested environment:**
  1. **Distribution:** Anaconda
  2. **Editor/IDE**: VSCode (with free copilot, GPT 4.1+)
- **Example:** Show how to use *VSCode*
  1. Run python *code* and *notebooks*
  2. Import US GDP data from online API and plot
  3. Solve the consumer problem from microeconomics

Infrastructure

Getting started

- **Web-page:** The course is organized around https://sites.google.com/view/numeconcph-progecon/
- **DataCamp:** Online courses on Python (requires no installation)
  $\Rightarrow$ you get 6 months free access (see e-mail with details)
- **Installation of Python:** Follow the installation guide

GitHub.com (code hosting platform)

- **All course materials will be shared on GitHub**
- **Organization:** www.github.com/NumEconCopenhagen

  Repositories:

  1. **ProgEcon-lectures**: slides, course plan, guides etc.
  2. **ProgEcon-exercises:** problem sets, solutions etc.

- **Git:** A version-control system for tracking changes in files and coordinating work $\Rightarrow$ integrated in VSCode

Download course content guide

1. Follow the installation guide
2. Open *VScode*
3. Windows: Pres *Ctrl+Shift+P* to *command control palette*
   Mac: Press Cmd+Shift+P
4. Write *Git: Clone*
5. Use *https://github.com/NumEconCopenhagen/ProgEcon-lectures*
6. Repeat with
   https://github.com/NumEconCopenhagen/ProgEcon-exercises
7. Create copies of the folder to work in
8. You can update later with *Git: Sync*

Lectures, classes and exam

- **Lectures:** 36, 39-41, 43-50 (see calendar)
- **Exercises:** 37-41, 43-41

Lectures, classes and exam

- **Lectures:** 36, 39-41, 43-50 (see calendar)
- **Exercises:** 37-41, 43-41
- **Exam requirements** (see deadlines)
  1. Sep: Basic programming test (on DataCamp.com, see e-mail)
  2. Oct: Project 1: Data project
  3. Nov-Dec: Project 2: Model project
- **Exam:** Portfolio of revised projects (40%)
  + 48 hours home assignment (60%)
- **Grading:** Pass or fail
- **Groups:** All projects can be done in *fixed* groups (maximum of 4)

Course plan - lectures

- **Week 37-38: Basics** (types, containers, views vs. copies, conditionals, loops, functions, floating point numbers, classes, numpy)
  $\Rightarrow$ *online videos instead of physical lectures*
- **Week 39-41: Data** (print, plot, save/load, online data, descriptive economics, structure and documentation, workflow and debugging, git)
- **Week 43-44: Tools** (optimization, root-finding, random numbers, simulation)
- **Week 45-48: Models** (Solow, Walras, AS-AD, calibration/estimation)
- **Week 49-50: Perspectives** (algorithms, speed, parallization, dynamic programming, artificial intelligence)

Lectures are recorded (and uploaded raw without editing)

More screen capture videos from *Introduction to Programming and Numerical Analysis* and are extra material

Course plan - exercises

- **Week 37-39:** DataCamp + Problem Set 0: Getting started
- **Week 40:** Problem set 1. Printing and plotting
- **Week 41:** Problem set 2. Descriptive economic analysis
- **Week 43:** Work on data project + intro to git
- **Week 44:** Problem set 3. Solving consumer problems
- **Week 45:** Problem set 4: Simulating economic dynamics (+ feedback)
- **Week 46:** Problem set 5: Solving and simulating Solow models
- **Week 47:** Problem set 6. Solving exchange and production economies
- **Week 48:** Problem set 7. Simulating AS-AD models
- **Week 49-51:** Work on your model analysis project + feedback + exam prep

Learning phases

1. **Basics:** A million new concepts and syntax.
   How does it all fit together!?!?! I will never understand this.

Learning phases

1. **Basics:** A million new concepts and syntax.
   How does it all fit together!?!?! I will never understand this.
2. **Data:** Arghhh! It is also hard to use in practice.
   But I begin to be able to do some cool stuff.

Learning phases

1. **Basics:** A million new concepts and syntax.
   How does it all fit together!?!?! I will never understand this.
2. **Data:** Arghhh! It is also hard to use in practice.
   But I begin to be able to do some cool stuff.
3. **Models:** Hard to understand math and code simultaneously.
   But the combination is really powerful.

Learning phases

1. **Basics:** A million new concepts and syntax.
   How does it all fit together!?!?! I will never understand this.
2. **Data:** Arghhh! It is also hard to use in practice.
   But I begin to be able to do some cool stuff.
3. **Models:** Hard to understand math and code simultaneously.
   But the combination is really powerful.
4. **Perspectives:** I have learned a lot! And so many possibilities ahead for working with data and models.

Work-flow

Your work-flow

- **Lectures:**
  1. Attend physical lectures and *participate actively*
  2. Watch videos
  3. *Try out the code yourself*

Your work-flow

- **Lectures:**
  1. Attend physical lectures and *participate actively*
  2. Watch videos
  3. *Try out the code yourself*
- **Exercises:** Work on problem sets and ask questions to your fellow students and the teaching assistants
  1. Solve tasks and problems
  2. Fill the missing code
  3. Find the error
  4. Solve full problem

  Note: OK to peak at answers, but write the solution yourself

Your work-flow

- **Lectures:**
  1. Attend physical lectures and *participate actively*
  2. Watch videos
  3. *Try out the code yourself*
- **Exercises:** Work on problem sets and ask questions to your fellow students and the teaching assistants
  1. Solve tasks and problems
  2. Fill the missing code
  3. Find the error
  4. Solve full problem

  Note: OK to peak at answers, but write the solution yourself
- **In between classes and lectures:**
  1. Go through lecture notebooks (curriculum)
  2. Solve the problem set
  3. Experiment with your own ideas

Getting help

- **Observation:** Programming is the slow and painful removal of tiny errors in your code – one at a time

Getting help

- **Observation:** Programming is the slow and painful removal of tiny errors in your code – one at a time
- **Everybody often forgets the correct syntax** $\Rightarrow$ trial-and-error and testing is central, never a single correct approach

Getting help

- **Observation:** Programming is the slow and painful removal of tiny errors in your code – one at a time
- **Everybody often forgets the correct syntax** $\Rightarrow$ trial-and-error and testing is central, never a single correct approach
- **Many sources for answers:**
  1. Documentation
  2. Copilot/LLMs
  3. Fellow students
  4. Teachers (in person or e-mail)

Getting help

- **Observation:** Programming is the slow and painful removal of tiny errors in your code – one at a time
- **Everybody often forgets the correct syntax** $\Rightarrow$ trial-and-error and testing is central, never a single correct approach
- **Many sources for answers:**
  1. Documentation
  2. Copilot/LLMs
  3. Fellow students
  4. Teachers (in person or e-mail)
- **Help each other!!** You will learn a lot.
  Remember to be constructive and polite!

Projects

Basic programming test

- **You must complete the following courses on DataCamp**
  1. Introduction to Python
  2. Intermediate Python
  3. Introduction to Functions in Python
  4. Intermediate Object-Oriented Programming in Python
     (chapter: Overloading and Multiple Inheritance)
  5. Introduction to NumPy
- **First 3 exercise classes:** Reserved for your work on DataCamp

Data project

- **Objectives:**
  1. Apply data cleaning and data structuring methods
  2. Apply data analysis methods
  3. Structure a code project
  4. Document code
  5. Present results in text form and in figures
  6. Use GitHub
- **Content:**
  1. Import data from an online source
  2. Present the data visually
  3. Apply some method(s) from descriptive economics
- **Structure:**
  1. A self-contained single notebook presenting the analysis
  2. Fully documented python files
- **Hand-in:** Create and commit folder called "dataproject" in your GitHub repository

Model project

- **Objectives:**
  1. Apply model analysis methods
  2. Structure a code project
  3. Document code
  4. Present results in text form and in figures
- **Content:**
  1. Describe an algorithm on how to solve a simple economic model
  2. Solve (and perhaps simulate) a simple economic model
  3. Visualize results across e.g. parametrizations
  4. Analyze one or more extensions of the baseline model
- **Structure:**
  1. A self-contained single notebook presenting the analysis
  2. Fully documented python files
- **Hand-in:** Create and commit folder called "modelproject" in your GitHub repository

Summing up

Summing up

- **I hope your have an idea of:**
  1. Why learning programming and numerical analysis is important
  2. What you will learn in this course
  3. Why active participation is required
  4. How you will qualify for and pass the exam

Your to-do list

1. **First priority:** Login to DataCamp.com (see info in e-mail)

Your to-do list

1. **First priority:** Login to DataCamp.com (see info in e-mail)
2. **Second priority:** Install **Anaconda**, **Git** and **VSCode** (see the installation guide)

Your to-do list

1. **First priority:** Login to DataCamp.com (see info in e-mail)
2. **Second priority:** Install **Anaconda**, **Git** and **VSCode** (see the installation guide)
3. **Third priority:** Download slides and exercises with git (see »Download course content guide«-slide)

Your to-do list

1. **First priority:** Login to DataCamp.com (see info in e-mail)
2. **Second priority:** Install **Anaconda**, **Git** and **VSCode** (see the installation guide)
3. **Third priority:** Download slides and exercises with git (see »Download course content guide«-slide)
4. **Fourth priority:** Run the example code from this lecture yourself