Exercise class 5

Introduction to Programming and Numerical Analysis

Class 3 and 6

Annasofie Marckstrøm Olesen

Spring 2024

What is Git and GitHub?

Git is a **version control system**: it helps you keep track of the changes you make to your files.

You can use git to keep track of any type of file from Microsoft Word documents to code bases.

GitHub is a cloud based platform for hosting code - basically **Dropbox for code**.

Git repositories

A repository is a folder that Git keeps track of. You can create Git repositories by:

1. Initializing a repository in an existing folder on your computer
   - In VS Code: Ctrl+Shift+P and type Git: Initialize Repository
   - Choose the folder you want

2. Clone an existing repository from GitHub
   - ...like when you cloned the lectures repository etc.

Committing your changes

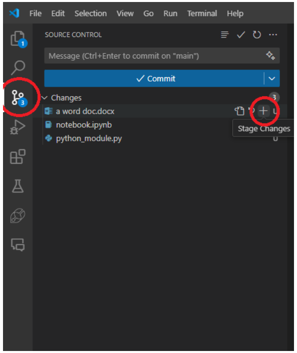
Git has several ”stages” of saving your files:

- **Working directory**: The current version of the files on your computer
- **Staging area**: changes that are ready to be committed
- **Committed changes**: Checkpoint or snapshot of the state of the repository.

Committing your changes

When you've made a change to a file, you want to keep:

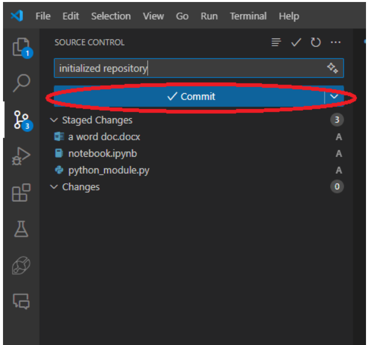
- Save the file (Ctrl+S) on your computer.
- Stage the file. In VS code:



Committing your changes

When you've made a change to a file, you want to keep:

- Save the file (Ctrl+S) on your computer.
- Stage the file.
- Commit the file. Remember to enter a commit message.



Uploading to GitHub

You can easily publish your local git repository directly to GitHub from VS Code (Ctrl+Shift+P and type Git: Publish Branch).

If you cloned your repository from a GitHub repository, it's already published, and you are ready to upload your changes (provided that you have writing access)!

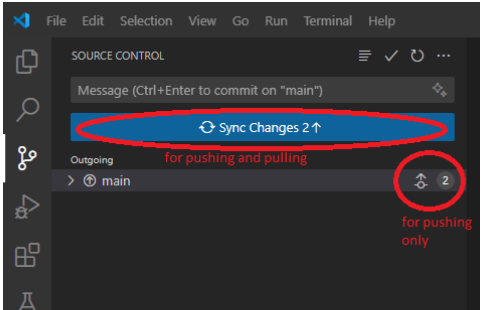
GitHub works the same way as eg. OneDrive or Dropbox: You store your files in the cloud, and you can **synchronize** them to your local computer.

But! Github doesn't sync automatically. You have to **manually** upload and download changes to files.

Uploading to GitHub

You can upload your changes to GitHub by Ctrl+Shift+P and typing Git: Push. This will push all you **committed** changes to the online GitHub repository.

You can also use the blue Sync button in VS code, which generally pushes AND pulls online changes. Or you can choose to push without also pulling - see picture.


File Edit Selection View Go Run Terminal Help
SOURCE CONTROL
Message (Ctrl+Enter to commit on "main")
Sync Changes 2
for pushing and pulling
Outgoing
main
2
for pushing only

Downloading from GitHub

You can pull new changes from GitHub by Ctrl+Shift+P and typing Git: Pull.

NOTE: You cannot do this if you have uncommitted changes in your local repository. Before you "Pull", you should always either **commit or discard your local changes**.

Bonus info: the "Pull"-command is actually a combination of two git commands, "Fetch" and "Merge". "Fetch" loads the changes from the cloud. "Merge" merges them into the working directory. Advanced users may sometimes prefer to perform these steps manually, but for now you can just use "Pull" to download changes.

Merge conflicts...

If your version of a file looks different than the corresponding file in the cloud, you can get a **merge conflict** when you try to sync!

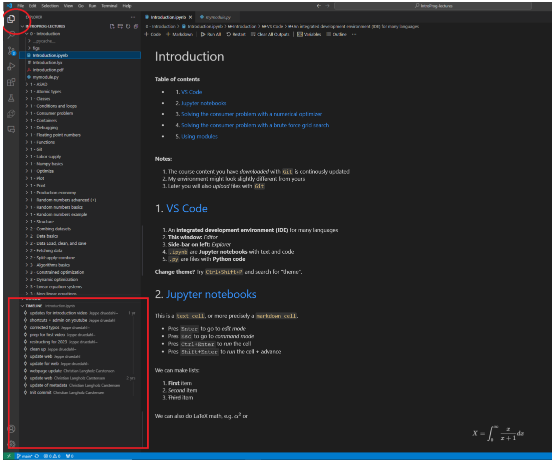
This can happen if two people are working on the same file at the same time.

If you get a merge conflict, you need to resolve it before you can complete the sync:

1. Choose the changes, you want to merge into your working directory.
2. Commit the merged changes.

Advanced: Version control

VS Code allows you to view the history of a file:


File Edit Selection View Go Run Terminal Help
IntroProg-lectures
EXPLORER
INTROPROG-LECTURES
Introduction.ipynb
mymodule.py
0 - Introduction > Introduction.ipynb > Introduction > VS Code > An integrated development environment (IDE) for many languages
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline
OUTLINE
main*
0 0
0

0 - Introduction
- __pycache__
- figs
- Introduction.ipynb
- Introduction.lyx
- Introduction.pdf
- mymodule.py

1 - ASAD
1 - Atomic types
1 - Classes
1 - Conditions and loops
1 - Consumer problem
1 - Containers
1 - Debugging
1 - Floating point numbers
1 - Functions
1 - Git
1 - Labor supply
1 - Numpy basics
1 - Optimize
1 - Plot
1 - Print
1 - Production economy
1 - Random numbers advanced (+)
1 - Random numbers basics
1 - Random numbers example
1 - Structure
2 - Combing datasets
2 - Data basics
2 - Data Load, clean, and save
2 - Fetching data
2 - Split-apply-combine
3 - Algorithms basics
3 - Constrained optimization
3 - Dynamic optimization
3 - Linear equation systems
3 - Non-linear equations

TIMELINE Introduction.ipynb

updates for introduction video	Jeppe druedahl~	1 yr
shortcuts + admin on youtube	Jeppe druedahl	
corrected typos	Jeppe druedahl~	
prep for first video	Jeppe druedahl~	
restructing for 2023	Jeppe druedahl~	
clean up	Jeppe druedahl~	
update web	Jeppe druedahl	
update for web	Jeppe druedahl~	
webpage update	Christian Langholz Carstensen	
update web	Christian Langholz Carstensen	2 yrs
update of metadata	Christian Langholz Carstensen	
Init commit	Christian Langholz Carstensen	

Introduction

Table of contents

- 1. VS Code
- 2. Jupyter notebooks
- 3. Solving the consumer problem with a numerical optimizer
- 4. Solving the consumer problem with a brute force grid search
- 5. Using modules

Notes:

1. The course content you have *downloaded* with `Git` is continously updated
2. My environment might look slightly different from yours
3. Later you will also *upload* files with `Git`

1. VS Code

1. An **integrated development environment (IDE)** for many languages
2. **This window:** *Editor*
3. **Side-bar on left:** *Explorer*
4. `.ipynb` are **Jupyter notebooks** with text and code
5. `.py` are files with **Python code**

Change theme? Try `Ctrl+Shift+P` and search for "theme".

2. Jupyter notebooks

This is a `text cell`, or more precisely a `markdown cell`.

- Pres `Enter` to go to *edit mode*
- Pres `Esc` to go to *command mode*
- Pres `Ctrl+Enter` to *run* the cell
- Pres `Shift+Enter` to *run* the cell + advance

We can make lists:

1. **First** item
2. *Second* item
3. ~~Third~~ item

We can also do LaTeX math, e.g. α^2 or

$$X = \int_0^\infty \frac{x}{x+1}dx$$

Advanced: Version control

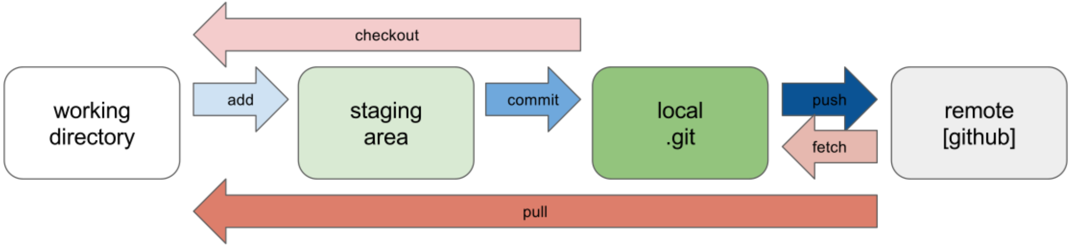
Git has LOADS of other functionalities. You can:

- Create different **branches** to work on specific tasks and then **merge** to the main branch.
- **Checkout** different branches and commits while preserving the state of your repository.
- **Revert** to previous commits, if you've made mistakes along the way.
- ...and much more.

There exist a bunch of apps to make it easier to interact with Git. I personally use GitHub Desktop .

Check out this video for how to do version control in VS code.

Git workflow



Important tips

If you're in a group, always **sync** before you make changes, and **try not to work on the same file at the same time** - otherwise you might get merge conflicts when you merge your work together.

Commit whenever you've added something to your code base. It's easier to navigate the commit history, when there are few changes per commit.

Summing up...

Skills you need:

- Staging and committing changes
- Pushing to GitHub
- Fetching and merging (pulling) from GitHub
- Handling merge conflicts

Other nice stuff you can checkout if you want:

- Reviewing commit history (git log)
- Checking out previous commits (git checkout)
- (Branches, merges, stashes and other version control tools)

Solving the consumer problem

The consumer problem generally has the form

$$\max_{\mathbf{c}\in\mathcal{C}} U(\mathbf{c})$$
$$s.t. \quad constraints$$

...so you need to be able to do **constrained optimization**.

See lectures on **Optimization** and **The Consumer Problem**, or Problem set 1. Or see the notebook on optimization and the consumer problem in our class repository.

Solving equilibrium models

Models with a **supply side** and a **demand side**.

Solution is a **set of prices** that makes supply equal to demand in all markets.

Demand for consumption goods usually stems from the solution to a **consumer problem.**

Supply of consumption goods can be eg. a production firm or consumers trading their endowments.

Standard approach

Solving these types of models can usually be done by:

1. Find demand as a function of prices.
2. Find supply as a function of prices.
3. Find excess demand (demand - supply) as a function of prices.
4. Find the point where excess demand is 0.

...and the solution if the price vector ensures point 4.

Remember, if you have N markets, you only need to do this for N-1 goods. The final one will clear by Walras' law.

See lectues on **Random numbers - example** and **Production economy**, or Problem set 2 (especially the extra problem with N>2 markets). Or the notebook on equilibrium models in our class repository.

The Inaugural Project

The inaugural project is about an Edgeworth economy with two agents and two goods.

Important skills for the inaugural project:

- Functions and classes.
- Optimization using grid search.
- Optimization using solvers, constrained and unconstrained.
- Printing and plotting results.
- General practical coding skills.

The Inaugural Project - Hand-in

Deadline for hand-in is **March 24th**, deadline for peer feedback is **March 31st**.

Hand-in by uploading your project to your GitHub repo:
github.com/NumEconCopenhagen/projects-YEAR-YOURGROUPNAME/inauguralproject

Your hand-in must include:

- A short README.md with introduction to project.
- A notebook presenting and discussing your results.
- A documented .py-file (you should base this on the provided file ExchangeEconomy.py)

The Inaugural project - Tips

The inaugural project asks you to:

1. Find market equilibrium in an exchange economy
2. Optimize utility under a bunch of different restrictions

So think about **constrained optimization**, the **consumer problem** and **equilibrium models**.

The questions are written in a way where **you can move on to the next question, if you get stuck**. So don't hesitate to do that if you need it.

General tips

Even thous this is a course on programming, don't forget to apply your **economic intuition!**

Always try to **print or illustrate** your results and **relate them to theory** - does the economic interpretation make sense to you?

Try to answer all the questions - if you cannot get the code to work, try to at least write down **what you tried to do and what you think went wrong**. This demonstrates your problem solving skills and can also give some points.

Make sure your notebook can run top to bottom **without errors** before handing in.

Next time

Video lectures

- Data and Pandas
- Loading, cleaning and saving data

Exercise:

- Problem set 3: Working with data from Statistics Denmark