Introduction to Programming and Numerical Analysis

Exercise Class 7
Exercise 5

Jonas Theodor Ø. Schmidt

UNIVERSITY OF COPENHAGEN

Today's Program

- 15:15 – 15:30: Introduction to Inaugural Project and Tips
- 15:30 – 16:00: Work on Inaugural Project
- 16:00 – 16:15: Break
- 16:15 – 16:25: Introduction to GitHub
- 16:25-16:55: Work on Inaugural Project
- 16:55-17:00: A Quick Review of the Most Asked Questions of Today

Introduction to Inaugural Project

Formal requirements

- Deadline for hand-in is March 24th and deadline for peer-feedback is March 31st

- Hand-in by uploading to your GitHub repository, i.e.:
  - github.com/NumEconCopenhagen/projects-YEAR-YOURGROUPNAME/inauguralproject

- Your hand-in must include:
  - A short README.md with a introduction to your project
  - A Jupyter notebook (.ipynb-file) that presents and discusses your results
  - A documented .py-file based on the provided file ExchangeEconomy.py

Question 1

Illustrate the set C in the Edgeworth Box

- Define functions for utility and demand of A and B
  - Remember that price 2 is numeraire, and that a function can return more than one output

- Find the set C and visualize the set in the Edgeworth plot
  - One may find inspiration in 'Lecture 1 – Conditions and Loops'
  - One may adjust the code for the Edgeworth box to plot the set C

Question 2 and 3

Calculate the error in the market clearing condition, and find the market clearing price

- Remember Walras' Law: If N – 1 markets clear, then all N markets clear
- One may find inspiration in ‘Lecture 1 – Optimizers’ and Problem Set 2
- Extra: Create a plot

Question 4A and 4B

Find the allocation if only prices in the set P1 can be chosen, and if any positive price can be chosen

- One may find inspiration in ‘Lecture 1 – Optimizers’ and Problem Set 1

- Extra: Create a plot

Question 5A and 5B

Find the allocation if the choice set is restricted to the set C, and if no further restrictions are imposed

- One may find inspiration in 'Lecture 1 – Optimizers' and Problem Set 1
  - A solution to one of the problems could involve constrained optimization

- Extra: Create a plot

Question 6A and 6B

Find the resulting allocation, and illustrate this and the former allocations

- One may find inspiration in ‘Lecture 1 – Optimizers’ and Problem Set 1
- Plot the allocations
- Important that you comment on and discuss the results, i.e., which allocations favours agent who?

Question 7 and 8

Draw a set W with 50 elements of wA = (w1A, w2A), and find the market equilibrium allocation for each endowment

- One may find inspiration in ‘Lecture 1 – Random Numbers’, ‘Lecture 1 – Optimizers’, Problem Set 1 and Problem Set 2

- Important! Set a seed

- Plot the allocations and comment on the plot – what does it illustrate?

All Questions

- Comment your code with #
- Use Markdown for introductions, explanations and discussions
- Plot a lot!
- ‘Clear All Outputs’, ‘Restart’ and ‘Run All’ in your Jupyter notebook after answering each question
- Rather hand-in your incorrect code and thoughts on a possible solution than to not answer a question

Break

What is Git and GitHub?

- Git is a version control system, which keeps track of file changes

- GitHub is a cloud-based platform for hosting code

Git Repository

- A Git repository is a folder that Git keeps track of.

- Git repositories can be created by:
  1. Initializing a repository of a existing folder on your computer
     - >Git: Initialize Repository
  2. Clone an existing repository from GitHub
     - >Git: Clone Repository

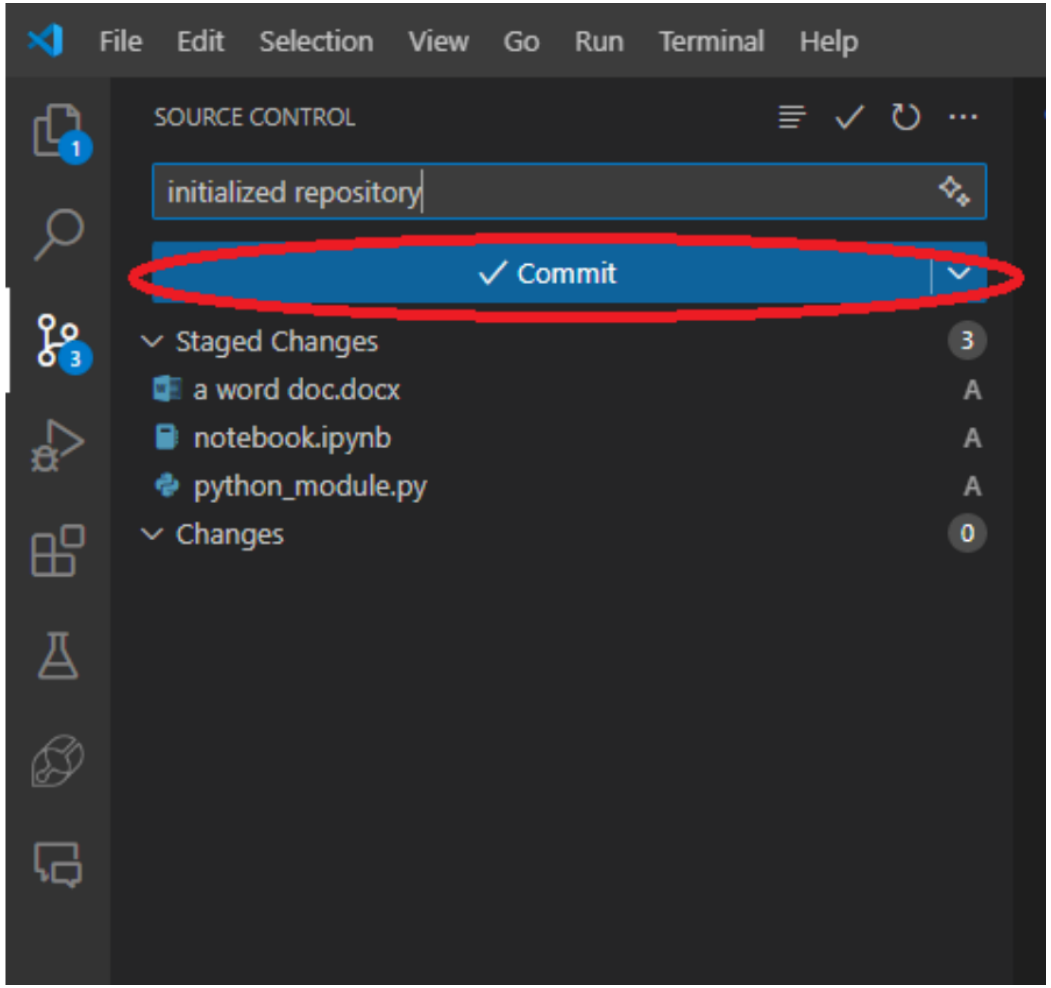
Committing Changes

Git has several "stages" of saving your files:

1. Working Directory: Current version of the files on the computer
   - Access by saving the files, i.e., Ctrl + S (PC) or Command + S (Mac)

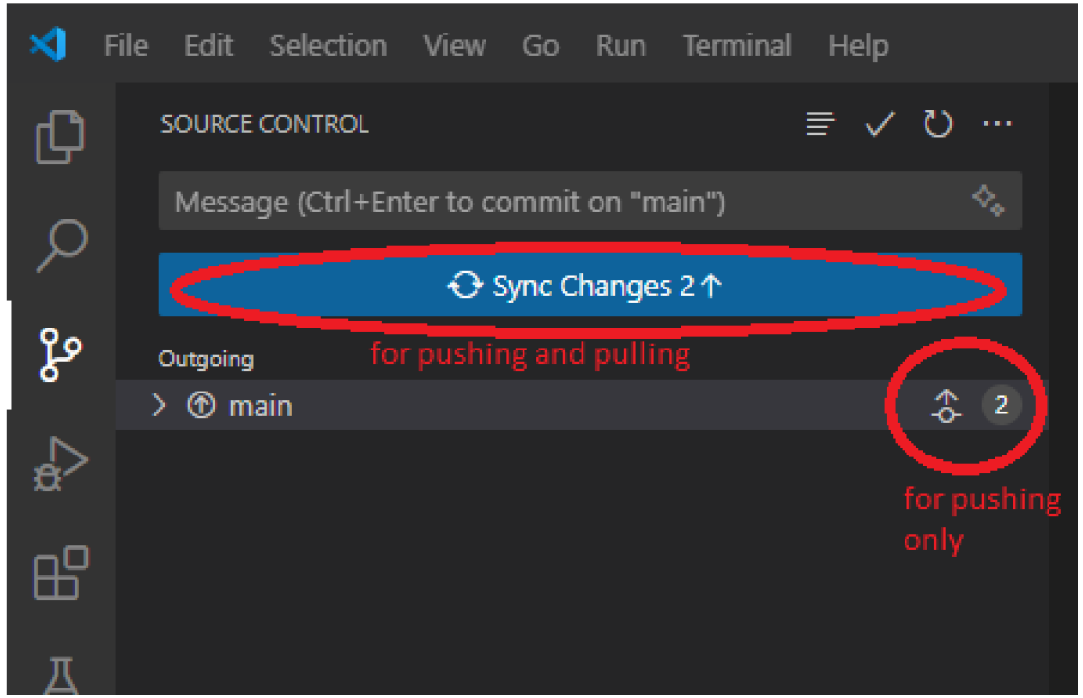
2. Staging Area: Changes that ready to be committed
   - Access by 'Source Control' in VSCode
   -

3. Committed Changes: Current state of repository
   - Access by clicking 'Commit' in 'Source Control' in VSCode



Upload to GitHub

- Publishing Your Repository
  - Publish your local git repository to GitHub via VS Code by >Git: Publish Branch
- Already Published Repositories
  - Upload to a repository cloned from (assuming write access)
- File Synchronization
  - GitHub allows for cloud file storage but does not automatically synchronize
    - Manual uploads and downloads are necessary for changes.
  - Upload changes to GitHub by >Git: Push or use the blue ‘Sync Changes’ buttom



Download from GitHub

- Pull new changes from GitHub by >Git: Pull

- Important Precaution
  - Pulling is not possible if there are uncommitted changes in your local repository

- Prepare to Pull
  - Commit or discard any local changes before attempting to pull from GitHub

Merge Conflicts

- Occurs if the computer file and the cloud version, usually occurs when edited by multiple people

- How to Resolve:
  - Select desired changes for your directory
  - Commit these changes to complete sync

Important for Cooperative Work In GitHub

- When working in a team, ensure you coordinate with your group members before modifying any files to avoid simultaneous edits that can lead to merge conflicts during integration

- Commit to your code base each time you make a noteworthy addition. Maintaining smaller, more frequent commits facilitates easier tracking and understanding of the commit history

Questions & comments?