# “Dynamic programming and structural estimation” mini course
# HSE, St.Petersburg, January 2020

## Instructions on how to set up your own or lab computer, access and submit your course assignments

### Step 1. Register at GitHub

If not already a registered GitHub user, go to https://github.com/ and follow signup procedure.
You can get additional benefits by applying for the Student developer pack https://help.github.com/en/github/teaching-and-learning-with-github-education/applying-for-a-student-developer-pack

### Step 2. Download GitHub Desktop

https://help.github.com/en/desktop/getting-started-with-github-desktop/installing-github-desktop
Github desktop is a  GUI for the Git software that makes interactions with version control system much more productive. There are many other GUI for Git https://git-scm.com/downloads/guis/

### Step 3. Install Python environment

Follow instructions at https://python.quantecon.org/getting_started.html
Make sure that Python, Jupyter Notebook and scientific libraries are installed on your computer. All necessary components are provided by Anaconda installation. https://www.anaconda.com/download

## Step 4. Make sure Jupyter Notebooks run as expected

Continue with the instructions at
https://python.quantecon.org/getting_started.html

# How to submit your assignments

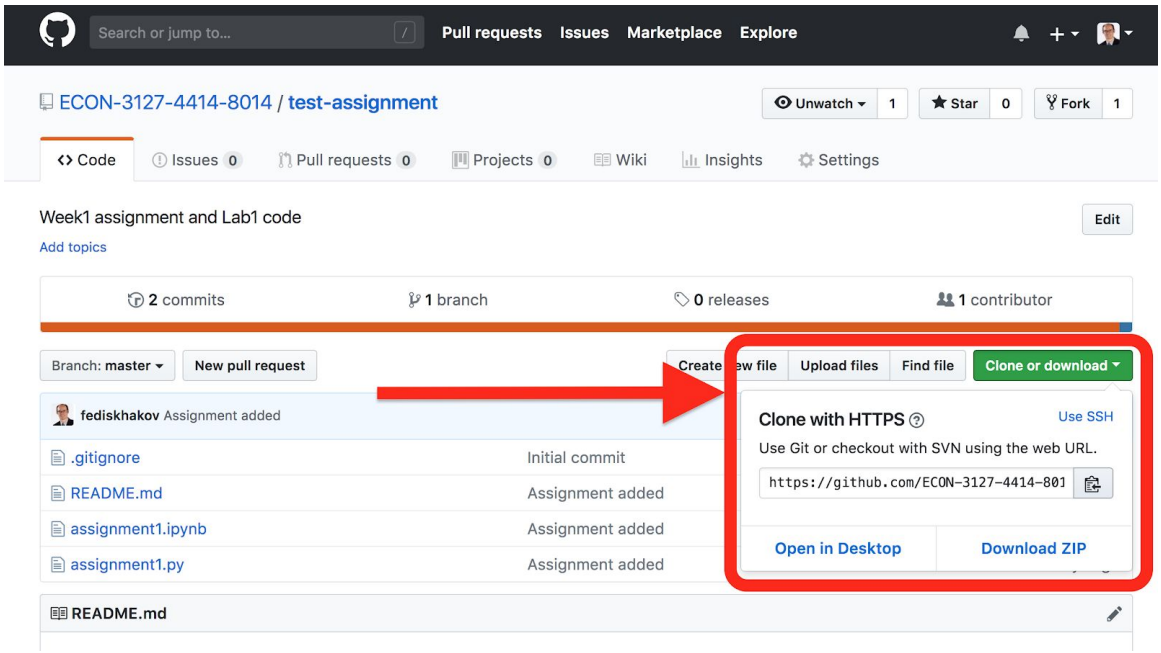
The instructions below are written for the command line Git interface, GitHub Desktop and other GUI have corresponding functionality implemented with buttons and menu commands.

## Step 1. Finding the assignment repository

Follow the *provided* link to the assignment, it will take you to Github repository (sign up and then sign in is required).
Links to the assignments will be distributed for each assignment separately.

## Step 3. Cloning the assignment repository

Clone the repository created for you to complete the assignment to your local computer. Click **Cone of download**, and copy the *https:// link*



After that run in your local Git shell in the folder where you want the assignment to appear

```
git clone <copied link>
```

Alternatively, in Git GUI or other program run the **Clone existing repository** command and choose the folder where you want the assignment to appear.

## Step 4. Working on the assignment

Find the folder with the assignment with your normal file browsing tools (Windows Explorer, Mac Navigator), and start working on the assignment.

When starting Jupyter Notebook, make sure that it starts up in the folder where all your assignment repositories are, so that you can browse them easily. Do not *upload* a notebook as it will save it outside of your Git working folder.

HowTo: number one, number two in case the first one does not work

When working on the pure python files use an appropriate text editor, and run python programs in the appropriate shell using command

```
python <name of your file>
```

Commonly used text editors which are good for coding can be found at from the common knowledge source. Sublime Text (paid) and Atom (free) are good alternatives, as well as Visual Studio Code and PyCharm (complete IDE).

## Step 5. Committing your changes

Don't forget to regularly commit your work (changes in your working directory) by running

```
git add *
git commit -m "informative commit message"
```

or using an appropriate commit button in your graphical Git interface.

Do not underestimate the usefulness of the git command **`git status`** and **`git log`** for monitoring your progress. These are implemented visually in graphical interfaces.

## Step 6. Submitting your work

To submit your work you have to push the local commits up to Github. To do this run (optional arguments may be needed in non-standard cases)

**`git push [origin master]`**

or run the **Push** command in your graphical Git interface.

The assignments will be graded using the last commit that is made before the submission deadline, but you can continue working on it and continue using the assignment code later as well, perhaps in your own research projects.

## Appendix A. Keeping up to date with lecture materials

Clone the lecture repository at
https://github.com/CompEconCourse/CourseMaterials
and to be up to date periodically run in your local Git folder

**`git pull origin master`**

If you change something in the lecture notebooks and the merge command is interrupted, you can commit local change to your local repository, or delete them by running

**`git reset --hard`** (this completely deletes all your changes)

# Appendix B. Working in groups

Some assignments may require you to work in groups. The groups correspond to “teams” on GitHub, and we are using the tools provided by GitHub Classroom to manage the student groups.

When you click on the link for midterm assignment, you’re prompted to join one of the existing groups of create a new one. Please, follow the groups defined in Wattle to either join your group or create your group if it’s not yet there. If your group is not yet on the list, create it using the name from the first column in the table on Wattle, otherwise just choose your group. Ideally, you are doing this together with your group partner, on both of your computers in the same time.

The provides starter code has all topics for the midterm assignment. You should only work on your prescribed topic. Feel free to delete other notebooks for easier grading of your work.

Each team will have a separate repository on GitHub that holds the work performed by the team. Each team member has access to the join repository. This is used to share the code that you collaborate on.  Each member of the team should have a local cloned version of the team repository on their local computer. Cloning is done in the usual way (see above).

Working in a group requires additional Git commands that you’ll have to use to share your work.  Assuming there is a central repository on GitHub that you cloned from, and local repositories set up for each team member, these are the needed commands.  They have corresponding buttons or settings in graphical interfaces as well.

- To get the changes made on the project by other team members, and merge them with your own work (***do this each time before starting***

***working on your local version***)
**`git pull`**
https://git-scm.com/docs/git-pull

- The same result can be achieved by first getting to know what the changes are by
  **`git fetch`**
  https://git-scm.com/docs/git-fetch
- Followed by merging the work by others into your repository with
  **`git merge`**
  https://git-scm.com/docs/git-merge
- The local commits (after you have done some work, and committed) can be send to the central online repository with usual
- **`git push (origin master)`**
  https://git-scm.com/docs/git-push

It is important for each team member to make regular commits with informative commit messages, because the individual grades for the midterm assignment may be dependent on the individual contributions documented by the commit message history in the assignment repository.

It is also important to keep in mind that if the local changes contradict the changes committed to the central repository, then both **`git pull`** and **`git merge`** will result in a merge conflict. The conflict can be resolved automatically “using theirs” or “using mine” in the graphical interface, or manually by editing the conflicting files using plain text editor (the Jupyter Notebook will not work if conflicted, because Git marks conflicting lines, and thus breaks the structure of Notebook file).
It is advisable to avoid conflicts by deleting all current changes (assuming they are not needed) with

- **`git reset --hard`**
  https://git-scm.com/docs/git-reset

In case of questions, please first read https://lab.github.com/.