

Introduction to GitHub

Python-do-ECARES

Fabrizio Leone

Introduction

Git and GitHub

- ▶ Git is a version control system, i.e. a way to keep track of the whole history of things you do on a file. It is useful to save, manage and edit all the different versions of your project.
- ▶ GitHub is a web service that allows to conveniently work with Git. It allows you to create your own directories, see projects of other people and collaborate with them.
- ▶ GitHub offers four different [subscription plans](#). We will work with a free one, which means that *everybody* can see what we do. However, with an academic account, you can also create private repositories.

GitHub: What It Does And Does Not Do

- ▶ No matter which subscription plan you choose, GitHub offers very limited storage space (you cannot upload files $> 100\text{MB}$). Therefore, it is **not** suitable for storing large files (e.g. datasets). **GitHub is not a substitute for a cloud.**
- ▶ GitHub is a platform where to upload mostly **source files** (e.g. .tex, .txt, .m, .R, .do, .py, .doc,...) and light pdf.
- ▶ You can read more about Git and GitHub [here](#) and [here](#).

GitHub Desktop

- ▶ In this course, we interact with GitHub mostly through the GitHub Desktop application.
- ▶ GitHub Desktop provides a simple yet powerful desktop interface to GitHub.
- ▶ You can download GitHub desktop [here](#).
- ▶ You can also interact with GitHub using the [Terminal](#) (not covered in this class).

Realistic Workflow

Suppose you are starting out your new project. If you use GitHub, you can

1. Create a local folder with your favourite sub-folders (e.g. code, slides, paper, literature, data,...).
2. Publish some sub-folders on GitHub (e.g. code, slides, paper).
3. Work on your code/paper/slides locally and then save different versions of them on GitHub. (No more: paper_v1, paper_v1.A, paper_v2.89.x7%,...).
4. Scroll through different versions of your files when you need. Collaborate with other people. (No more: paper_v1_myedition, paper_v1_youredition,...).

Introduction
○○○○○

First Steps With GitHub Desktop
●○○○○○○○○○○○○○○○○○○

Commit and Push
○○○○○

Browse Through History
○○○○○○

More Functionalities
○○

First Steps With GitHub Desktop

First Steps With GitHub Desktop

- ▶ This section shows how to clone a repository, create **your own branch** and how to commit and push changes to it.
- ▶ Make sure to familiarise with these steps. You will do this many times in the future.

Step 1. Clone Repository

Go to the Classes repository on GitHub and click on the "Clone or download" button.

Python-do-ECARES / Classes

Watch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

This repository contains codes and materials of each meeting. [Edit](#)

[Manage topics](#)

8 commits 1 branch 0 packages 0 releases 1 contributor 0 apps 2.0

Branch: master New pull request

Create new file Upload files Find file **Clone or download**

Fab993 Create Example.ipynb Latest commit 259f246 1 minute ago

Session_1 Create Summary.md 2 months ago

Session_2 Create Example.ipynb 1 minute ago

LICENSE Initial commit 2 months ago

README.md Create README.md 1 hour ago

README.md

Class Schedule

This repository contains the folders with the content of each class. Please check them for the references of each meeting. The schedule goes as follows (always 10-12AM in room R42.3.103)

Session	Date	Discussant	Topic	References
Class 1	11/02/2020	Glenn	Reproducible research, automation and introduction to the course	

Step 1.A. Open Repository

Choose "Open in Desktop".

The screenshot shows the GitHub web interface for the repository 'Python-do-ECARES / Classes'. The 'Clone or download' dropdown menu is open, showing options for cloning via SSH or HTTPS, and a button to 'Open in Desktop' which is highlighted with a red box. The repository page also shows a list of files and a 'Class Schedule' table.

Python-do-ECARES / Classes

Watch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

This repository contains codes and materials of each meeting. [Edit](#)

[Manage topics](#)

8 commits 1 branch 0 packages 0 releases 1 contributor Apache-2.0

Branch: master New pull request

Create new file Upload files Find file Clone or download

Fab993 Create Example.ipynb

- Session_1 Create Summary.md
- Session_2 Create Example.ipynb
- LICENSE Initial commit
- README.md Create README.md

README.md

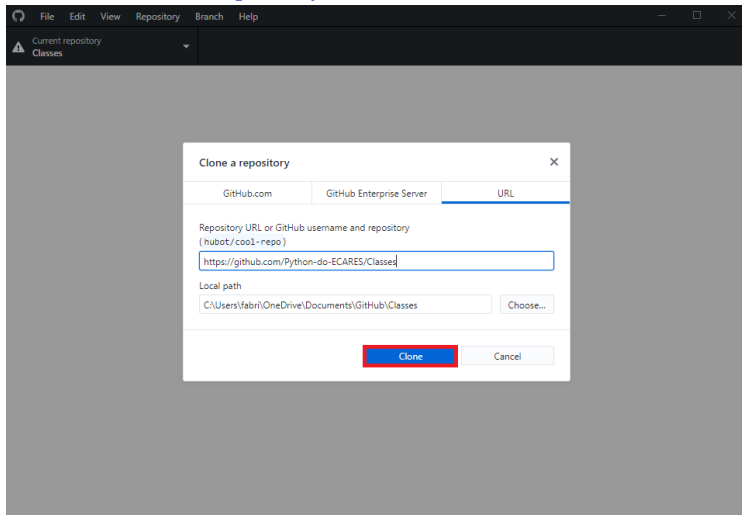
Class Schedule

This repository contains the folders with the content of each class. Please check them for the references of each meeting. The schedule goes as follows (always 10-12AM in room R42.3.103)

Session	Date	Discussant	Topic	References
Class 1	11/02/2020	Glenn	Reproducible research, automation and introduction to the course	

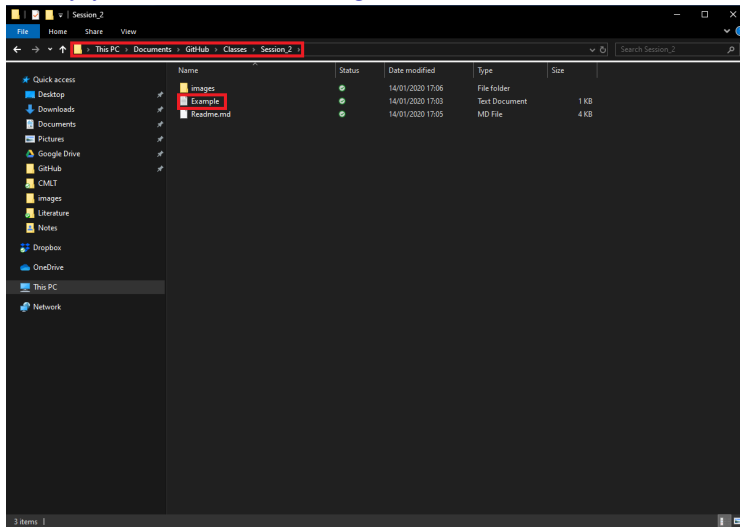
Step 1.B. Clone Repository

Check the local path where to clone the repository and hit "Clone".



Step 1.C. Open Local Directory

Under the chosen directory, you will see the following files.



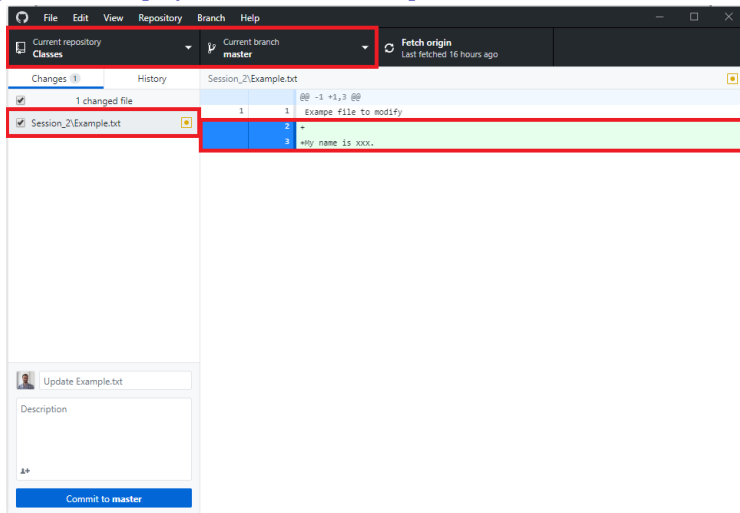
Step 2. Make Changes

Open *Example.txt*. Add a comment line with your name.



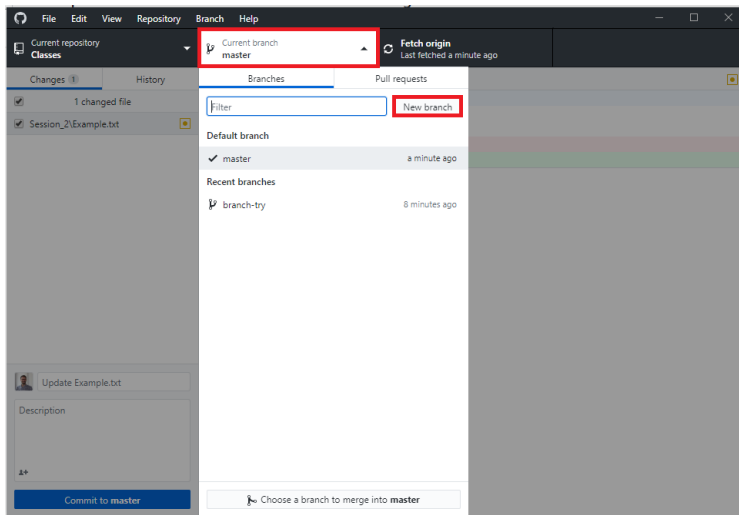
Step 2.A. Save Changes

Save the file. Changes will be displayed in GitHub Desktop as follows.



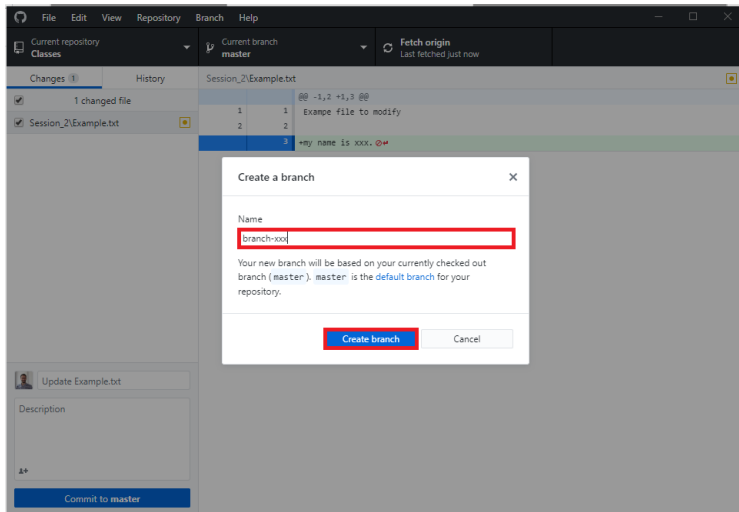
Step 2.B. Create Your Own Branch And Commit Changes

Select "Master" and hit "New Branch".



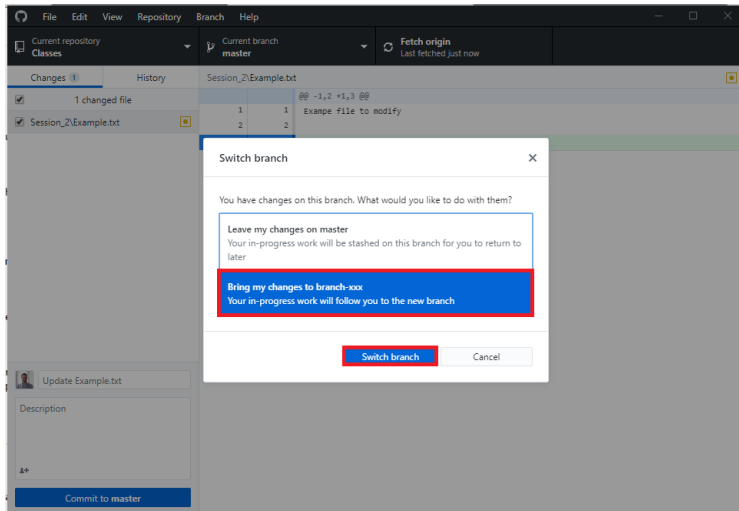
Step 2.C. Name Branch

Give the new branch your name. Then click "Create Branch".

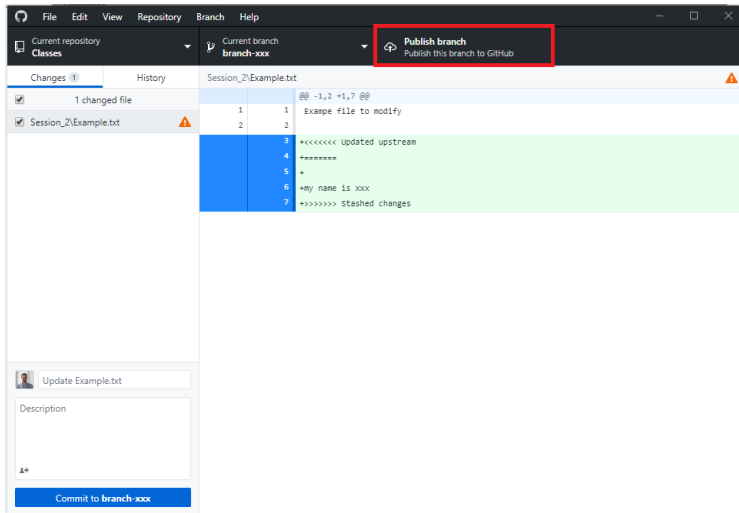


Step 2.D. Switch Branch

Switch changes to your own branch.

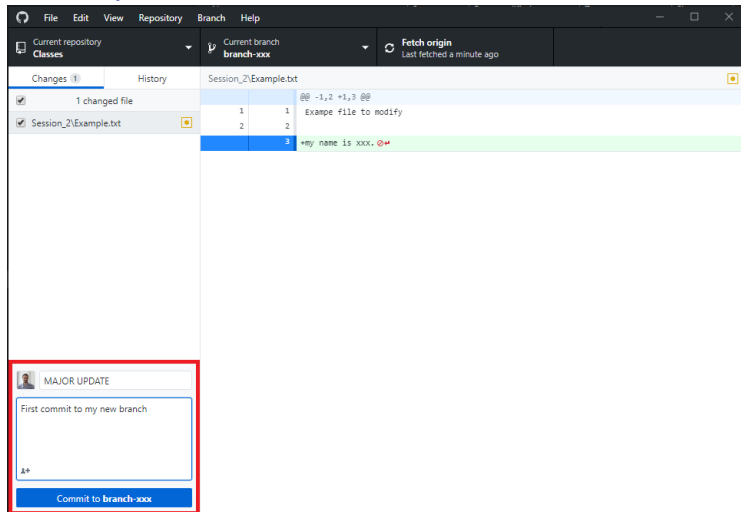


Publish your branch online.



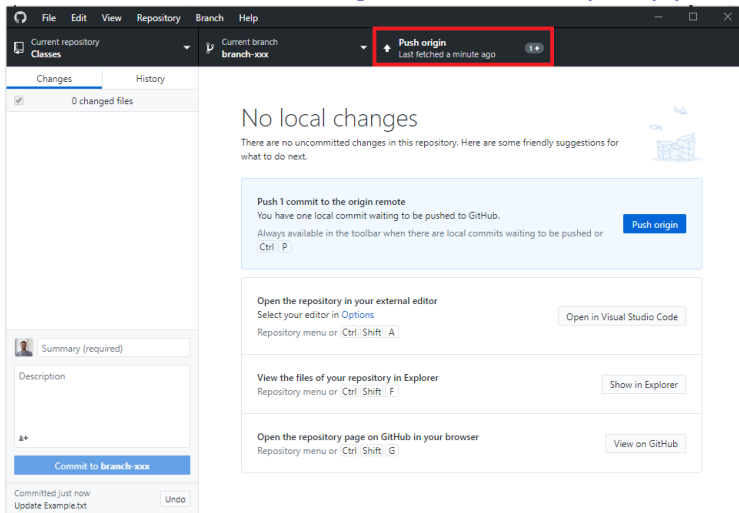
Step 2.F. Commit to Branch

Give Description and summary. Then commit.



Step 2.G. Push to Branch

Push changes to your branch. You are not allowed to push to the master. If you try, you will get an error.



Step 3. Next Steps

Go to "Classes" page. Notice that now there are 2 branches. Click on "branches".

Python-do-ECARES / Classes

Watch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

This repository contains codes and materials of each meeting. [Edit](#)

[Manage topics](#)

49 commits **2 branches** 0 packages 0 releases 1 contributor Apache-2.0

Your recently pushed branches:

branch-xxx (1 minute ago) [Compare & pull request](#)

Branch: master New pull request

Create new file Upload files Find file Close or download

Feb993 Update README.md Latest commit 5ec9f91 27 seconds ago

Session_1	Create Summary.md	2 months ago
Session_2	Update README.md	27 seconds ago
LICENSE	Initial commit	2 months ago
README.md	Create README.md	18 hours ago

README.md

Class Schedule

This repository contains the folders with the content of each class. Please check them for the references of each meeting. The schedule goes as follows (always 10-12AM in room R42.3.103)

Session	Date	Discussant	Topic	References
Class 1	11/02/2020	Glenn	Reproducible research, automation and introduction to the course	

Step 3.A. Done!

Your directory will now appear below "branches" as follows.

The screenshot displays the GitHub web interface for the repository 'Python-do-ECARES / Classes'. The 'branches' tab is selected, showing a list of branches. The 'Your branches' section is highlighted with a red box, indicating the current state of the repository. It shows a branch named 'branch-xxx' updated 5 minutes ago by Fab993, with a progress bar and a 'New pull request' button. The 'Default branch' section shows 'master' as the default branch, updated 1 minute ago by Fab993, with a 'Change default branch' button. The 'Active branches' section also shows 'branch-xxx' updated 5 minutes ago by Fab993, with a progress bar and a 'New pull request' button. The interface includes a search bar, navigation links (Pull requests, Issues, Marketplace, Explore), and a footer with copyright information and links to GitHub's resources.

Taking Stocks

- ▶ You are now able to create and maintain **your own repositories**.
- ▶ Please **make sure to only push changes to your own branch during this course**. You are encouraged to collaborate and see what other people is up to, but **never** commit changes directly to other people's branches.
- ▶ Good news is that you can always revert changes back if you do so by mistake. This is why GitHub is so useful!

Introduction
○○○○○

First Steps With GitHub Desktop
○○○○○○○○○○○○○○○○○○○○

Commit and Push
●○○○○

Browse Through History
○○○○○○

More Functionalities
○○

Commit and Push

Commit and Push

- ▶ **Committing** and **pushing** are the main two words you have to familiarize with.
- ▶ *Committing changes* to a branch, means that you are creating a new version of your file.
- ▶ *Pushing changes* means, instead, that you are publishing them online on GitHub. Think of the pushing action as a way of creating different stable releases of your code.
- ▶ Only push changes online if you have made a stable change.

Commit and Local Saving

- ▶ Notice that **committing to GitHub** and **saving your file locally** are very different things.
- ▶ If you save locally, you only change the file on your device. If you commit, you add a “node” to the chain of your versions.
- ▶ With this respect, we recommend to commit changes regularly, but also to use **standards**.

Commit Standards

- ▶ We encourage you to adopt the following standards to commit tidily.
- ▶ “Summary” should be either **Minor Change**, **Major Change** or **Bug Fixes**.
The first should indicate small changes in syntax or general improvements.
The second to major modifications (e.g. add new section or function), while
the third is to notify that you have fixed some bug.
- ▶ **Description** should briefly explain what the summary refers to.

Commit Standards

- ▶ Suppose you **create a new function for data cleaning in your code**. When pushing this change to GitHub, you want to give **Major Change** as summary and "added function for data cleaning" as description.
- ▶ A tidy commit activities will create a full history of changes in GitHub that you can scroll through to check different versions of your code.
- ▶ Finally, it will also help other people to understand your work.

Introduction
○○○○○

First Steps With GitHub Desktop
○○○○○○○○○○○○○○○○○○○○

Commit and Push
○○○○○

Browse Through History
●○○○○○

More Functionalities
○○

Browse Through History

Browse Through History

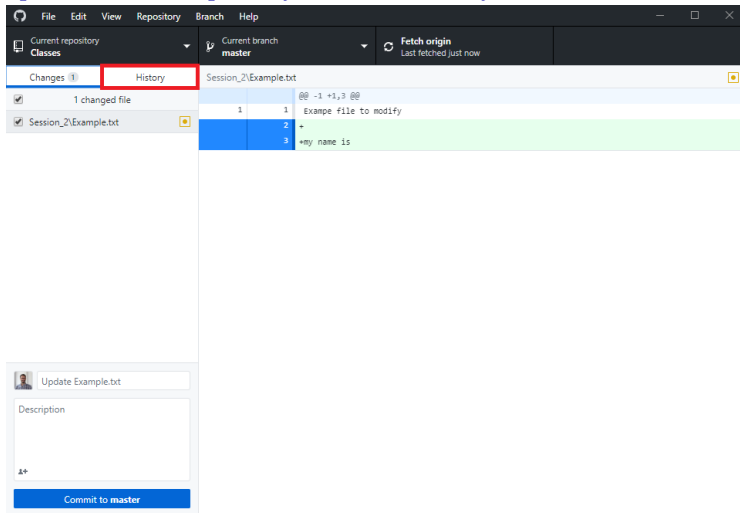
With GitHub Desktop, you can also browse through the entire history of your commits. This is very helpful to

- ▶ **Revert back changes.** Imagine you update a code but, at some point, you realise that one of the previous releases was better.
- ▶ **Compare different versions.** From time to time, you may want to look back at previous versions of your work (slide, paper, code,...).

A tidy committing activity will help you to easily browse through meaningful versions of your work. If you commit too often, you have to search a lot (versions only differ marginally from one another). If you commit too infrequently, you may lose information.

Step1. Revert Back Changes

Open GitHub Desktop at the current repository. Then hit "History".



Step2. Revert Back Changes

You can scroll through the whole history of changes you have made up to that moment.

The screenshot displays the GitHub Desktop application window. The top bar includes menus: File, Edit, View, Repository, Branch, and Help. Below this, the 'Current repository' is set to 'Classes', the 'Current branch' is 'master', and the 'Fetch origin' button is visible, indicating the last fetch was 28 minutes ago.

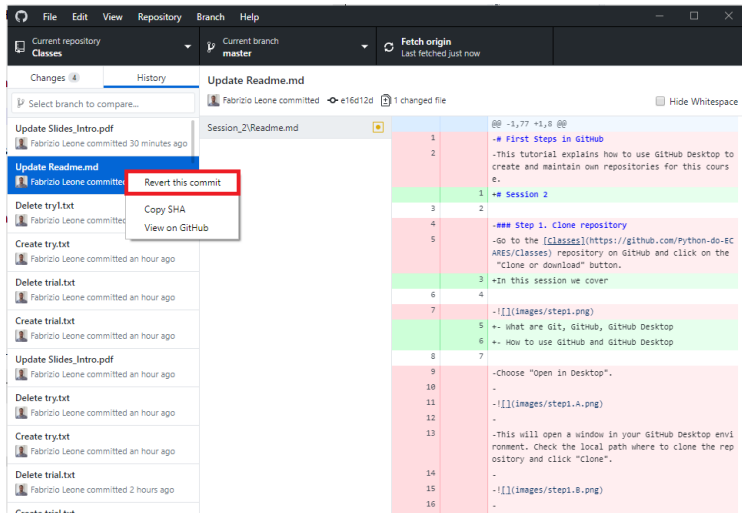
The main interface is divided into three panes. The left pane shows a list of changes, with 'Update Readme.md' (committed 31 minutes ago) highlighted in blue. The middle pane shows the selected file 'Session_2\Readme.md' with a diff view. The right pane shows the diff details for the selected file, indicating 1 changed file.

The diff view shows the following changes:

- Line 1: `-- First Steps in GitHub`
- Line 2: `-This tutorial explains how to use GitHub Desktop to create and maintain own repositories for this course.`
- Line 3: `+## Session 2`
- Line 4: `--### Step 1. Clone repository`
- Line 5: `-Go to the [Classes](https://github.com/Python-do-ECA/Classes) repository on GitHub and click on the "Clone or download" button.`
- Line 6: `+In this session we cover`
- Line 7: `--`
- Line 8: `+-. What are Git, GitHub, GitHub Desktop`
- Line 9: `+-. How to use GitHub and GitHub Desktop`
- Line 10: `-Choose "Open in Desktop".`
- Line 11: `--`
- Line 12: `-`
- Line 13: `-This will open a window in your GitHub Desktop environment. Check the local path where to clone the repository and click "Clone".`
- Line 14: `-`
- Line 15: `--`
- Line 16: `-`

Step3. Revert Back Changes

Click on "Revert this commit" to go back to that version.



More Details

- ▶ You can switch back and forth your versions as many times as you want.
Your local directories will change accordingly.
- ▶ If instead of "Revert this commit" you hit "View on GitHub", you can see all the differences between two versions of your file online.

Introduction
○○○○○

First Steps With GitHub Desktop
○○○○○○○○○○○○○○○○○○○○

Commit and Push
○○○○○

Browse Through History
○○○○○○

More Functionalities
●○

More Functionalities

More Functionalities

- ▶ Communicate with other people (your coauthors, other developers, etc.) by creating **issues**. Look at [here](#) for how to do it.
- ▶ You find a nice tutorial on how to do both [here](#).