Lecture 1: Setting our environment and parsing text files

COSC 526: Introduction to Data Mining

Spring 2020

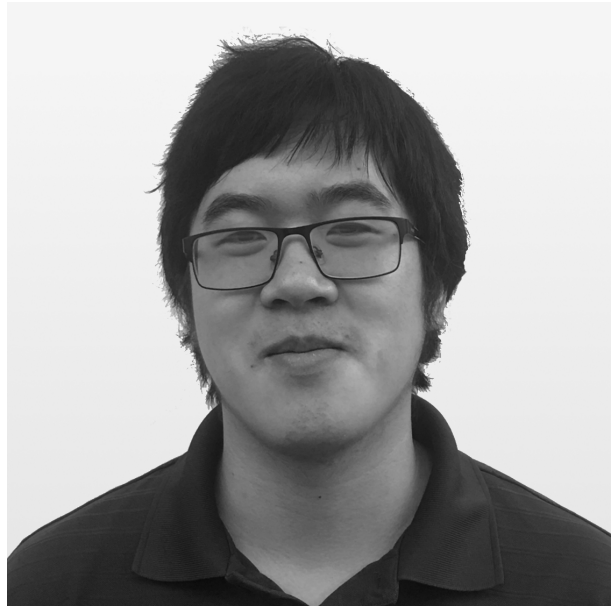
THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Instructors:

Michela Taufer

Danny Rorabaugh

GRA:



Nigel Tan

Set up our working environments

- We **use git & github** to distribute & collect assignments as well as other class materials (e.g., slides, code, and datasets)
  - Create a GitHub username (if you have not one)
  - Install git and github desktop on your laptop
- We use **Jupyter notebook** for our assignments and project
  - Follow the steps in the file StartHere to install Jupyter
- We use **XSEDE Jetstream** as our platform for assignments and the semester project
  - Introduce its use in the 2nd lecture
  - Create accounts and provide you with access to the cloud in the 4th lecture

Git and GitHub

GitHub and Git

- **GitHub Account:** web-based hosting service for **version control**
  - Use it to distribute and collect assignments, to share class materials (e.g., slides, codes, and datasets)
  - Provide us with your GitHub username
- **Git:** software used by GitHub
  - Install git on your laptop
- **Class GitHub Repository:**
  - Clone the course repository

    https://github.com/CISC879-BigData/**courses-UTK-COSC526-S20**

Git and GitHub

- Git: a free, open source distributed version control system
- GitHub: cloud hosted git repositories

Material built from:

http://rogerdudler.github.io/git-guide/

https://marklodato.github.io/visual-git-guide/index-en.html

https://kbroman.org/github_tutorial/pages/init.html

https://kbroman.org/github_tutorial/pages/routine.html

Install git on your laptop (if you have not yet)

- Windows or Mac
  - Download the GitHub Desktop application: https://desktop.github.com
  - Find GitHub's installation help here: https://help.github.com/desktop/guides/getting-started-with-github-desktop/
- Linux or Mac
  - Install the command-line interface (CLI) of git
  - Find help here: https://www.atlassian.com/git/tutorials/install-git
- Log into github

Connect new repos to github

- Go to github
- Log in to your account

You don’t have a github account?

Connect new repos to github

- Go to github
- Log in to your account

You don’t have a github account?

- Get a github account
  - Go to the GitHub Sign Up page https://github.com/join
  - Create a free account

Connect new repos to github

- Go to github
- Log in to your account

You don't have a github account?

- Get a github account
  - Go to the GitHub Sign Up page https://github.com/join
  - Create a free account
- Share your github account with us:
  - Complete form https://forms.gle/CKugke8Dzqjm9tQ89

Command-line interface (CLI)

CLI: Your first time with git and github (I)

After installing git and github desktop, if you are using CLI:

- Set up git with your user name and email

```
$ git config --global user.name "Your name here"
$ git config --global user.email "your_email@example.com"
```

- Set up ssh on your computer
  - Look to see if you have files ~/.ssh/id_rsa and ~/.ssh/id_rsa.pub.
  - If not, create such public/private keys:

```
$ ssh-keygen -t rsa -C "your_email@example.com"
```

From: https://kbroman.org/github_tutorial/pages/first_time.html

CLI: Your first time with git and github (II)

- Copy your public key (the contents of the newly created id_rsa.pub file) into your clipboard – e.g., on Mac

```
$ pbcopy < ~/.ssh/id_rsa.pub
```

- Paste your ssh public key into your github account settings
  - Go to your github Account Settings
  - Click "SSH Keys" on the left.
  - Click "Add SSH Key" on the right.
  - Add a label (like "My laptop") and paste the public key into the big text box.

From: https://kbroman.org/github_tutorial/pages/first_time.html

CLI: Your first time with git and github (III)

- In a terminal/shell, type the following to test it:

```
$ ssh -T git@github.com
```

- If it says something like the following, it worked:

```
Hi username! You've successfully authenticated, but Github does
not provide shell access.
```

From: https://kbroman.org/github_tutorial/pages/first_time.html

Submit your github username

- Complete form https://forms.gle/CKugke8Dzqjm9tQ89

Use git and github

- The routine use of git involves just a few commands:
  - init
  - add and commit
  - push and pull
  - status
  - diff
- You can deal with git and github via:
  - GitHub Desktop (GUI)
  - command line (CLI)

A new repo from scratch

- Create a directory to contain the project
  $ mkdir YourRepos
- Go into the new directory
- Type *git init*
  $ git init
- Create a new file – ReadMe.md is a good start
  $ echo "# YourRepos " >> ReadMe.md
- Type *git add* to add the file
  $ git add README.md
- Type *git commit* to commit the file
  $ git commit -m "first commit”

At this point, your repos is ONLY local

Keep the repo clean

- Create a **.gitignore** file to indicate all of the files you don't want to track in a (sub)directory

  $ git add .gitignore

- Create a **.gitignore_global** file to indicate all of the files you don't want to track for the entire directory. You have to tell git about the global .gitignore

  $ git add .gitignore_global

  $ git config --global core.excludesfile ~/.gitignore_global

.gitignore_global and .gitignore

*~

.*~

.DS_Store

.Rhistory

.RData

Connect new repo to github

- Go to github
- Log in to your account
- Click the new repository button in the top-right
- Click the “Create repository” button

- Back to your terminal:

$ git remote add origin https://github.com/YourAccount/YourRepos.git

$ git push -u origin master

Add and commit (I)

- Add completely new files as well as to “add” modifications to files that already exist in the repository (e.g., README.md)

  $ mkdir Lecture01

  $ touch Lecture01/README.md

  $ emacs README.md

  $ git add Lecture01 Lecture01/README.md README.md

- Use git commit to add the modifications to the repo

  $ git commit OR $ git commit -m “Message: 40 / 60 characters”

```
$ git commit -m "Fix such and such"
```

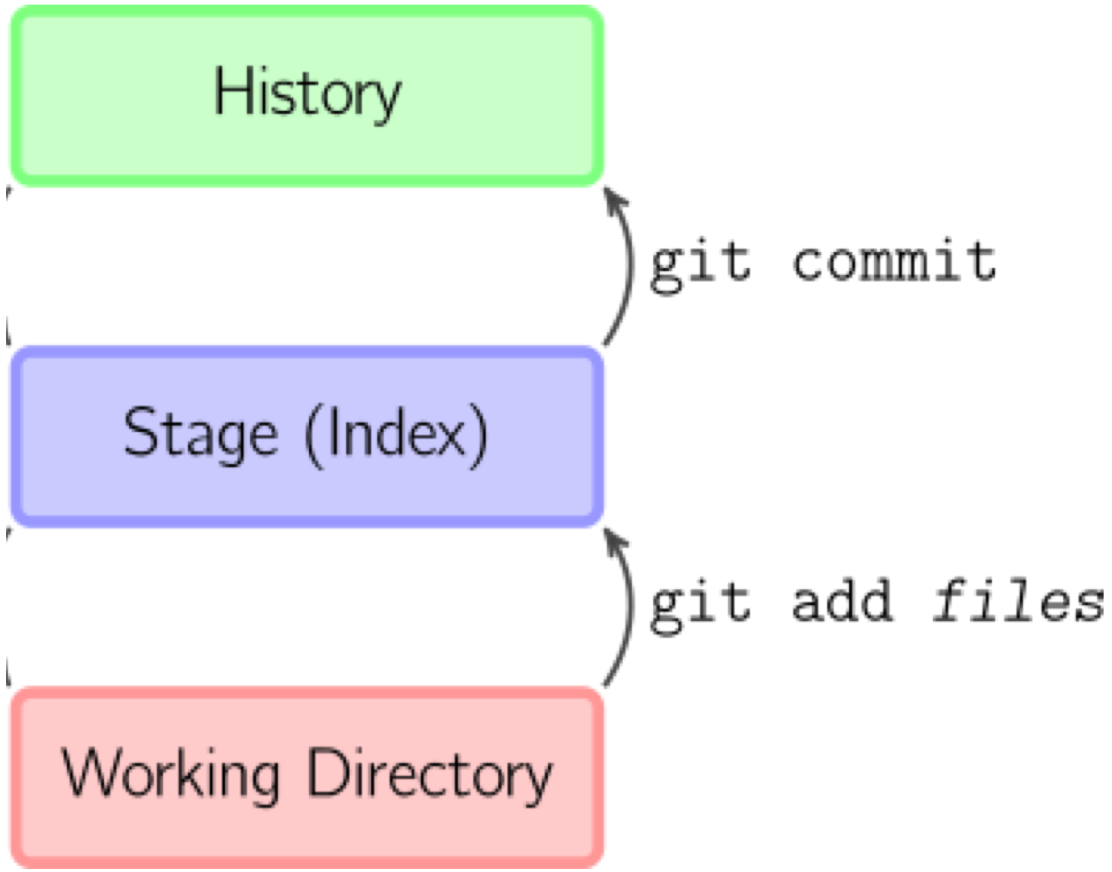
From: https://kbroman.org/github_tutorial/pages/routine.html

Add and commit (II)

$ git add <filename>

$ git add

$ git commit -m "Commit message”



Push to and pull from github

- After any *git add* and *git commit* commands, changes are in the **HEAD** of your local working copy
- To push committed changes to your remote repository - github, type:

  $ git push
- To pull committed changes from your remote repository - github, type:

  $ git pull

From: https://kbroman.org/github_tutorial/pages/routine.html

Differences github

- To extract differences and changes, type:

  $ git diff

- To extract changes in one specific file, type

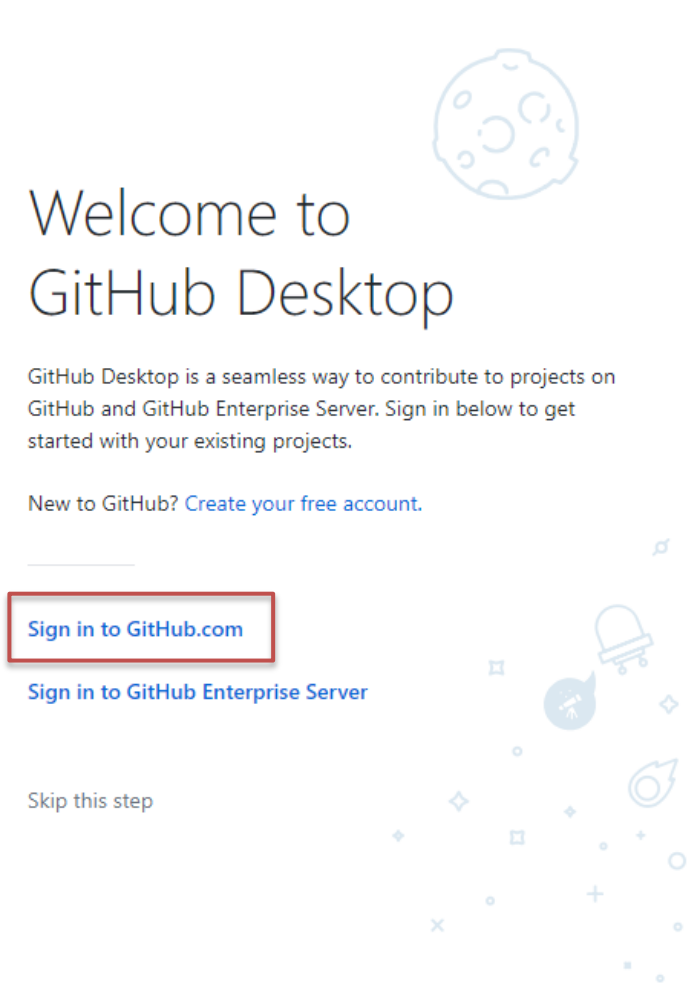
  $ git diff README.md

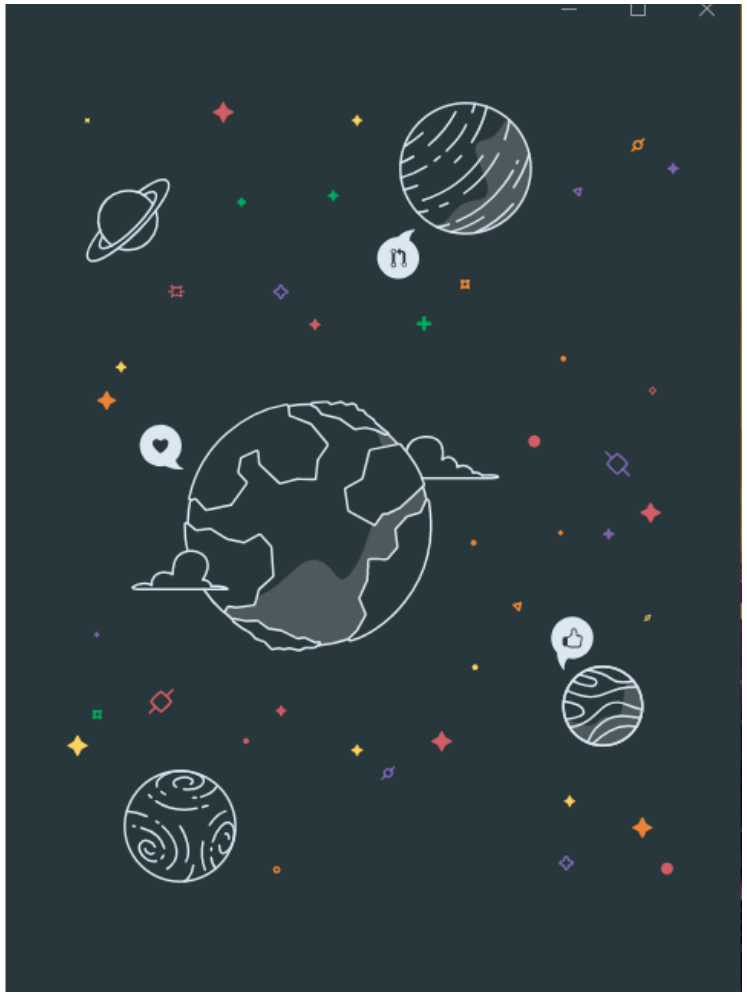
From: https://kbroman.org/github_tutorial/pages/routine.html

GitHub Desktop application

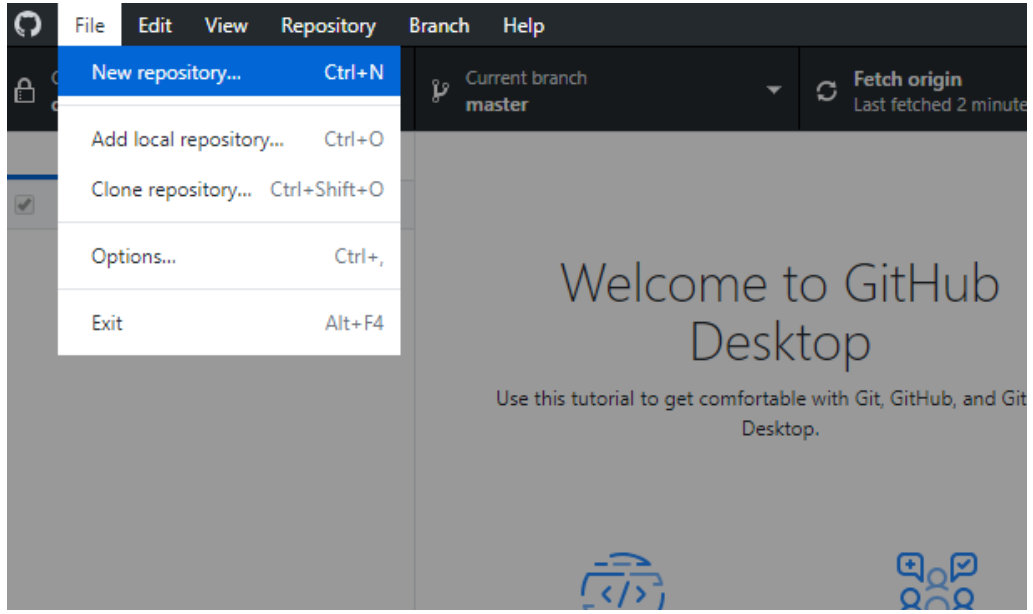
GitHub Desktop: sign-in

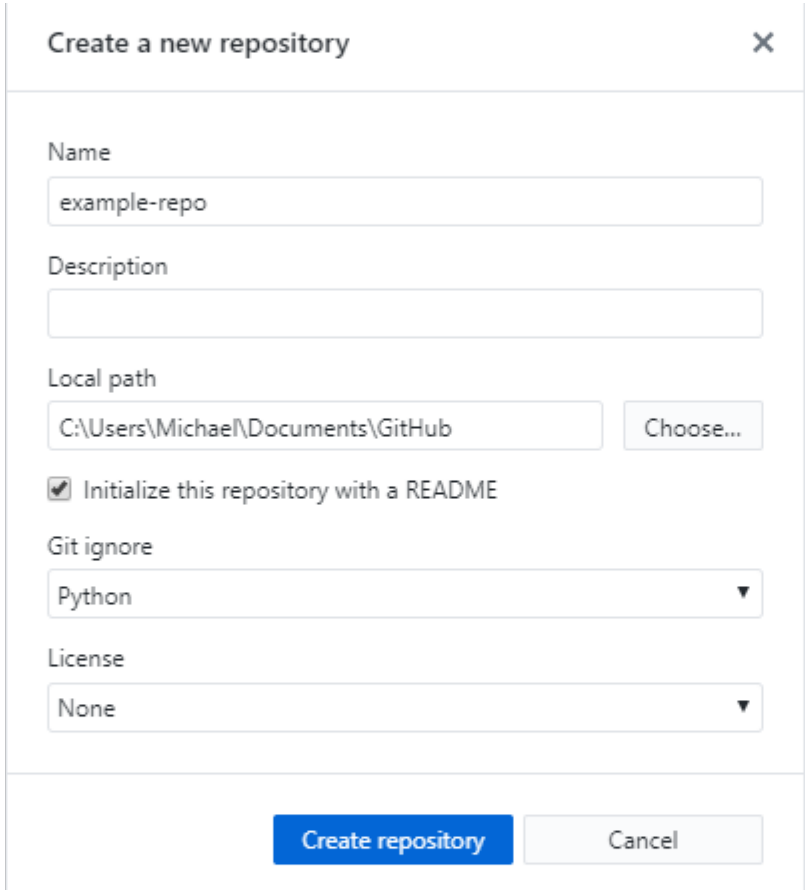
- Sign in with your Git username and password
- No need to setup an SSH key!
- If you want the CLI for Git on windows, check out: https://gitforwindows.org



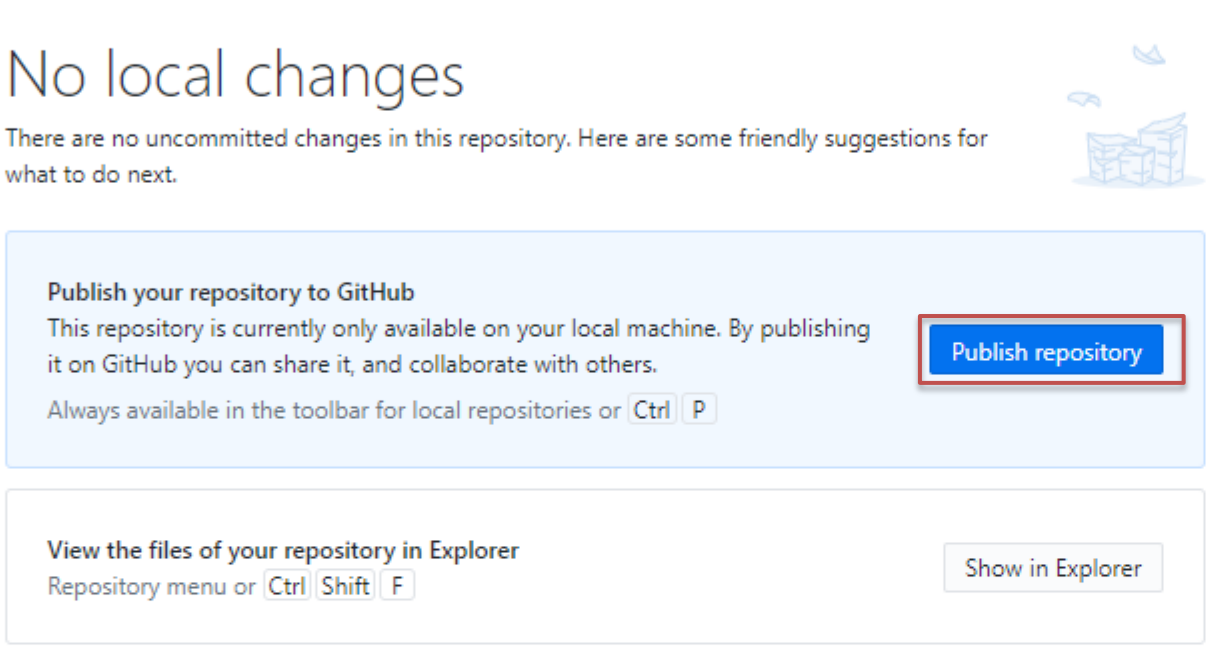


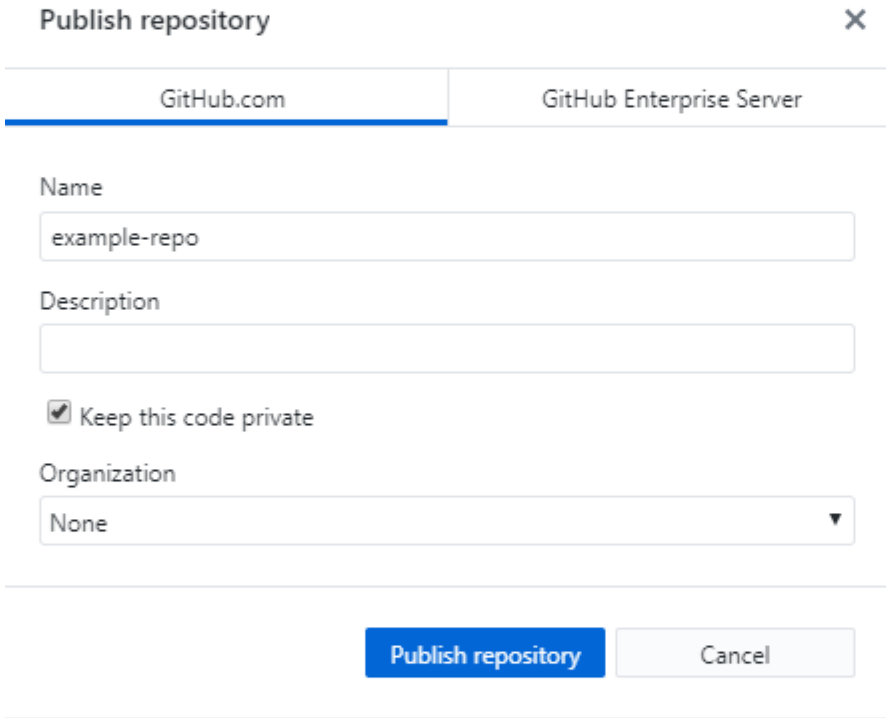
GitHub Desktop: create repo



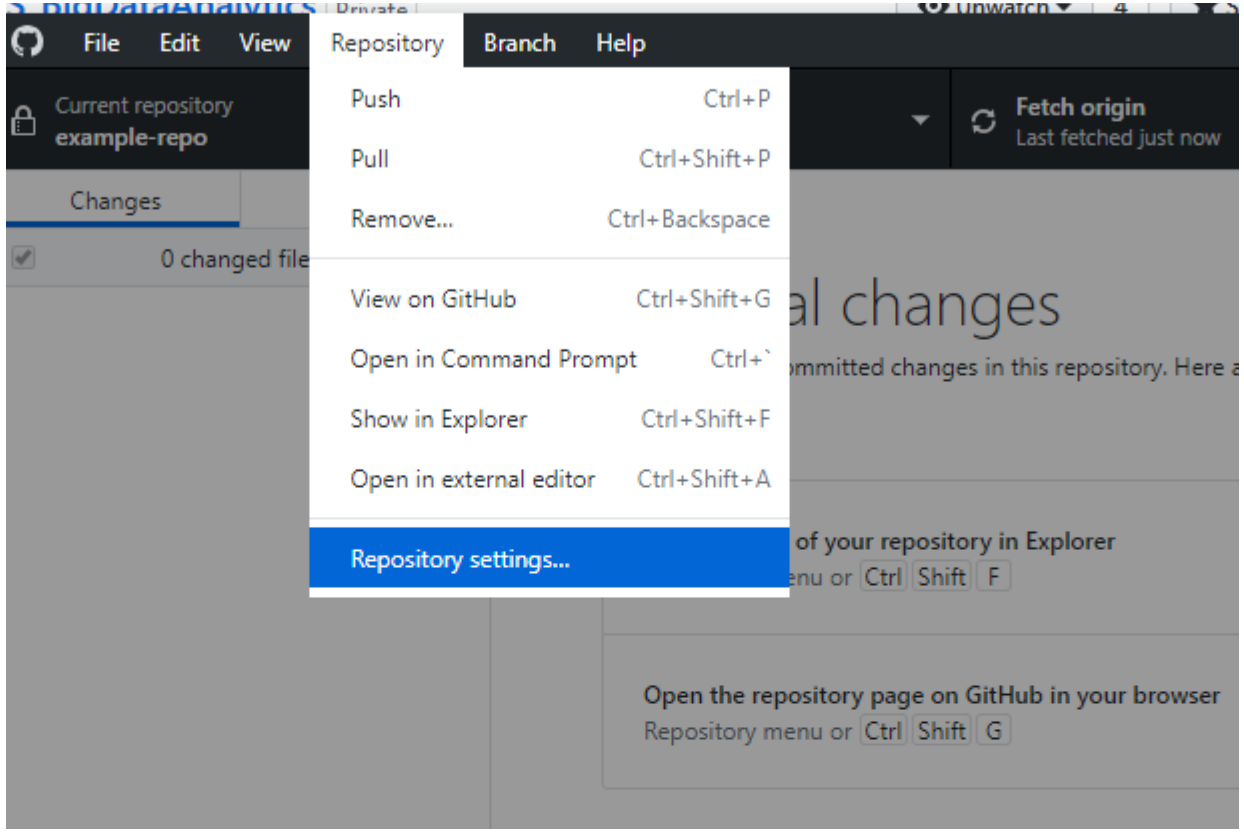


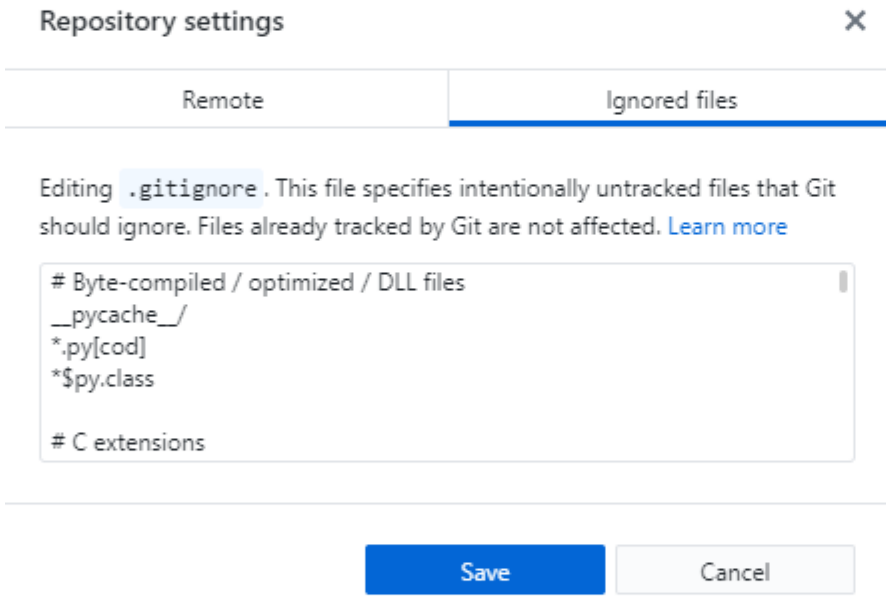
GitHub Desktop: push repo



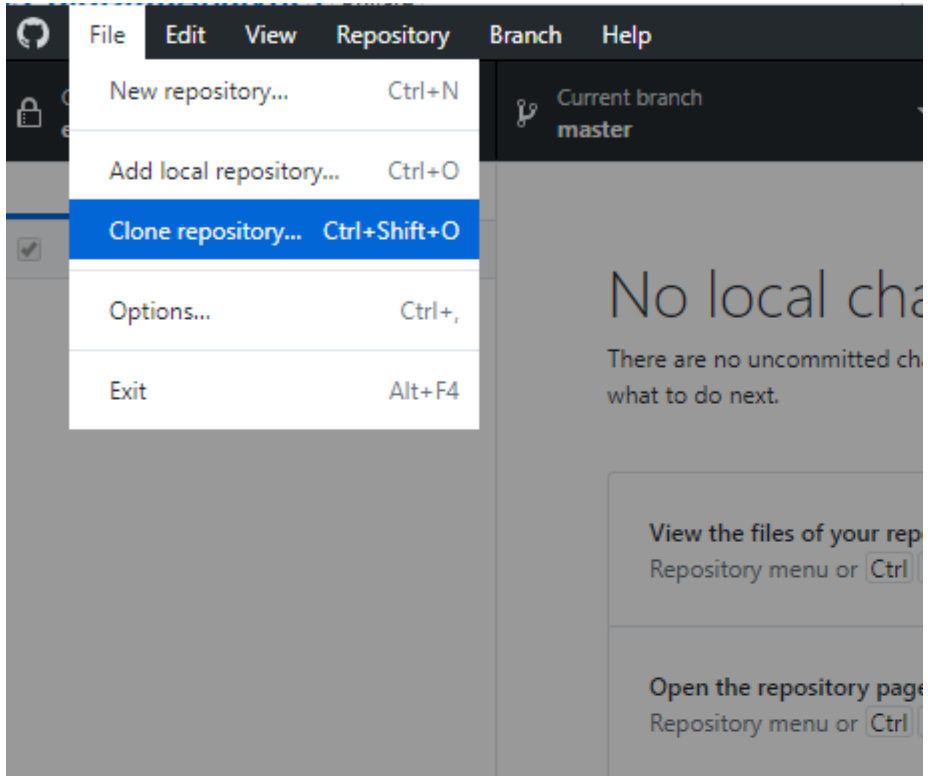


GitHub Desktop: git ignore

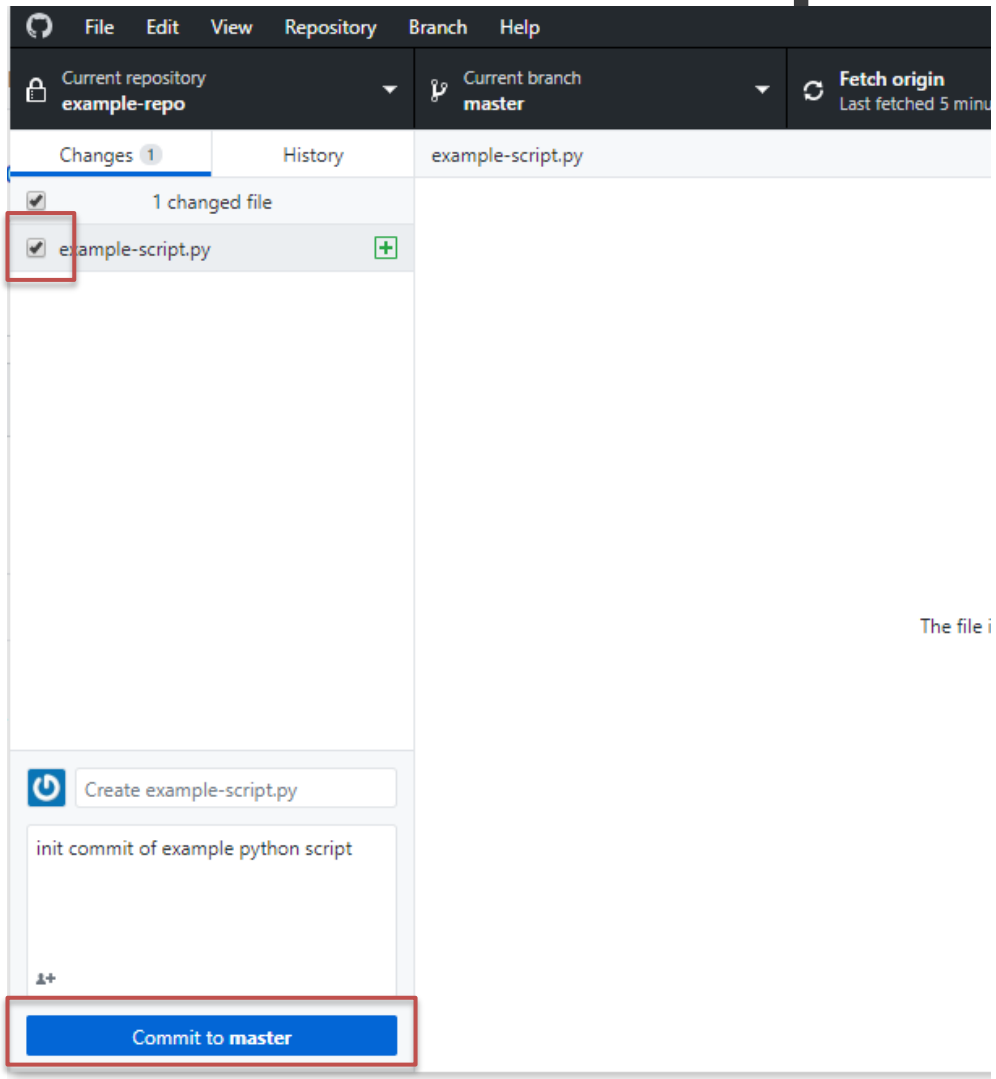


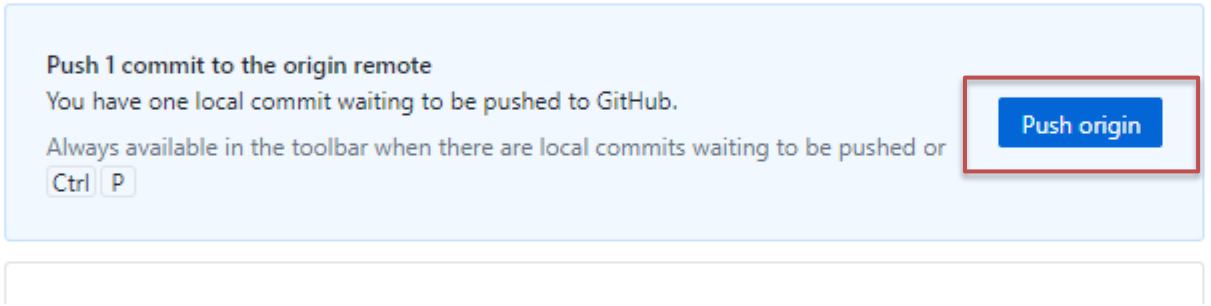


GitHub Desktop: clone from git



GitHub Desktop: adding and committing



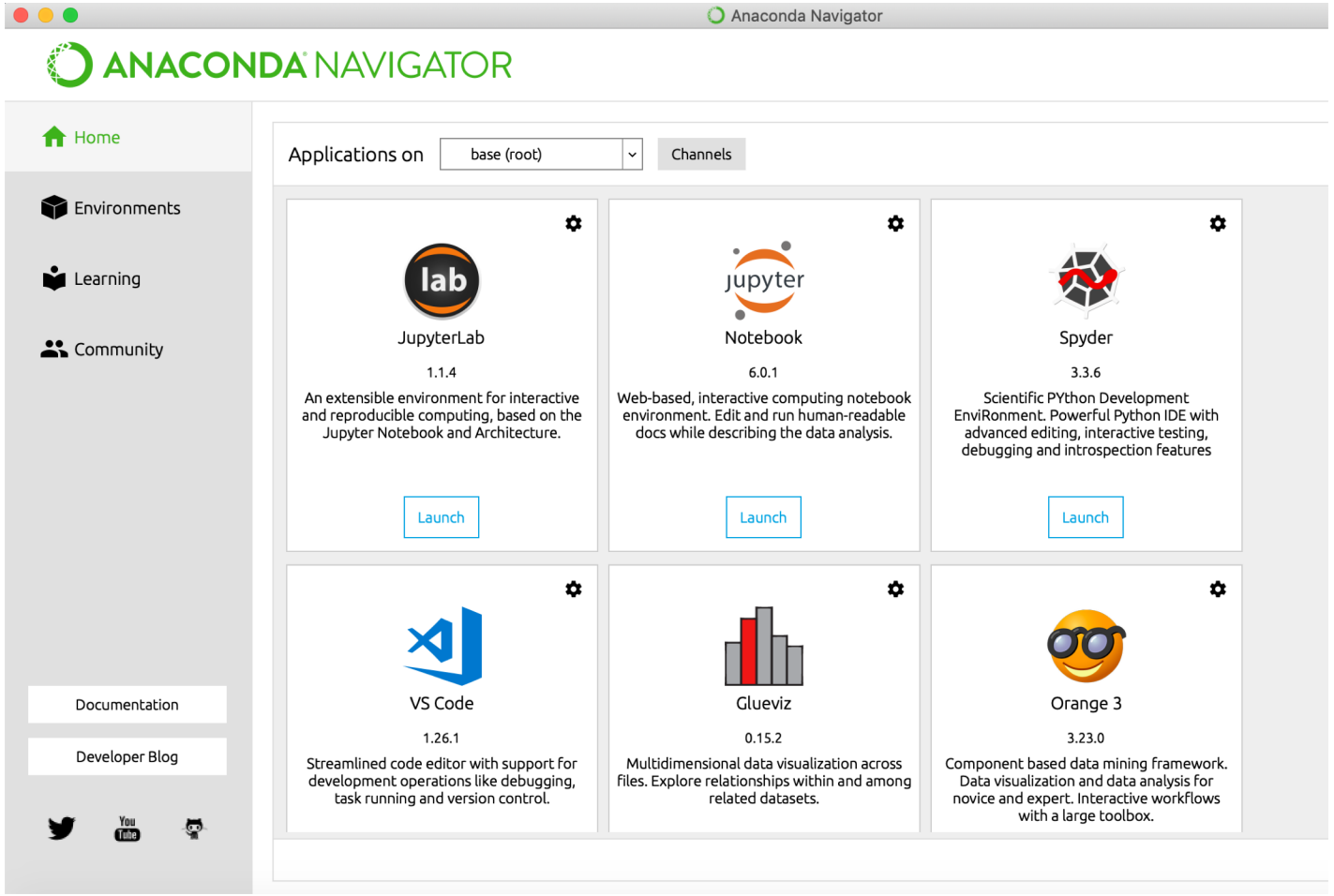


Python, Anaconda, and Jupyter Notebook

Python and Anaconda

- **Python:** It is Python 3.6!
- **Anaconda:** Python distribution
  - Include many popular packages by default, including Jupyter notebook
  - Make installing additional packages easy
- Your task:
  - Install Anaconda and launch the Jupyter notebook on your laptop by following the steps in the file StartHere.pdf file in course repos (Assignment01)

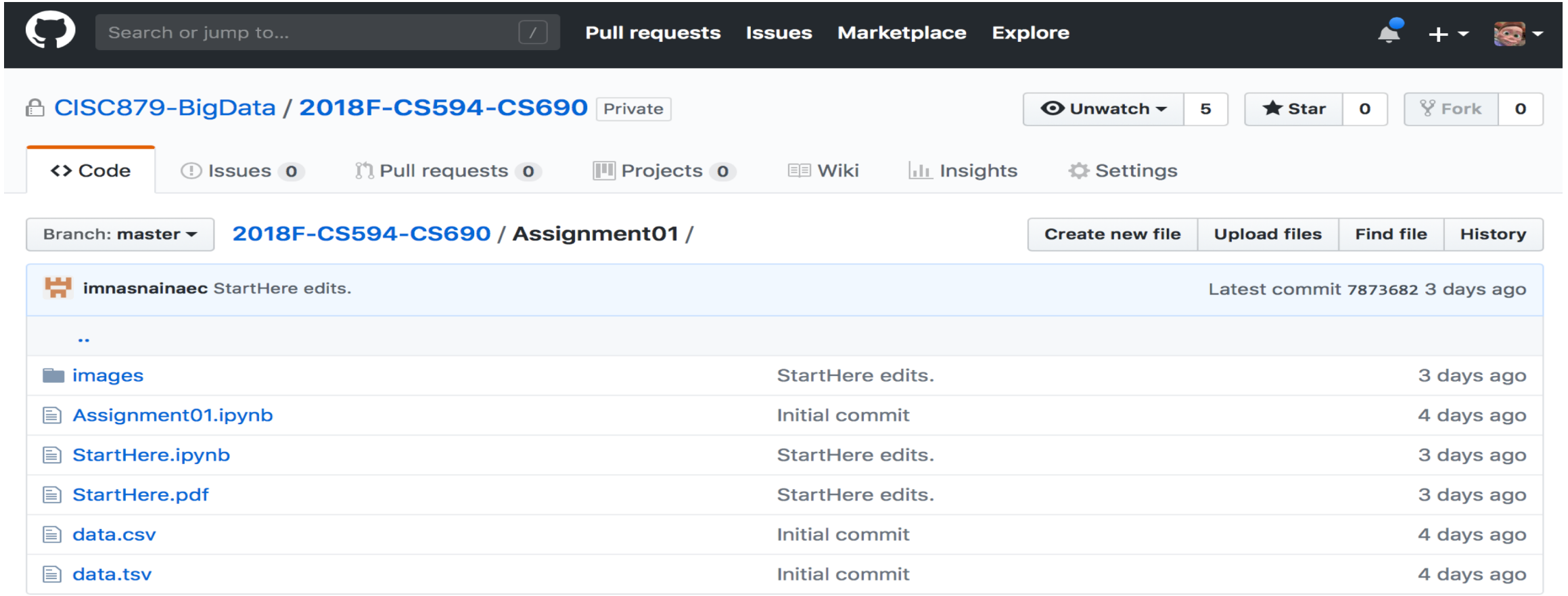
Anaconda navigator



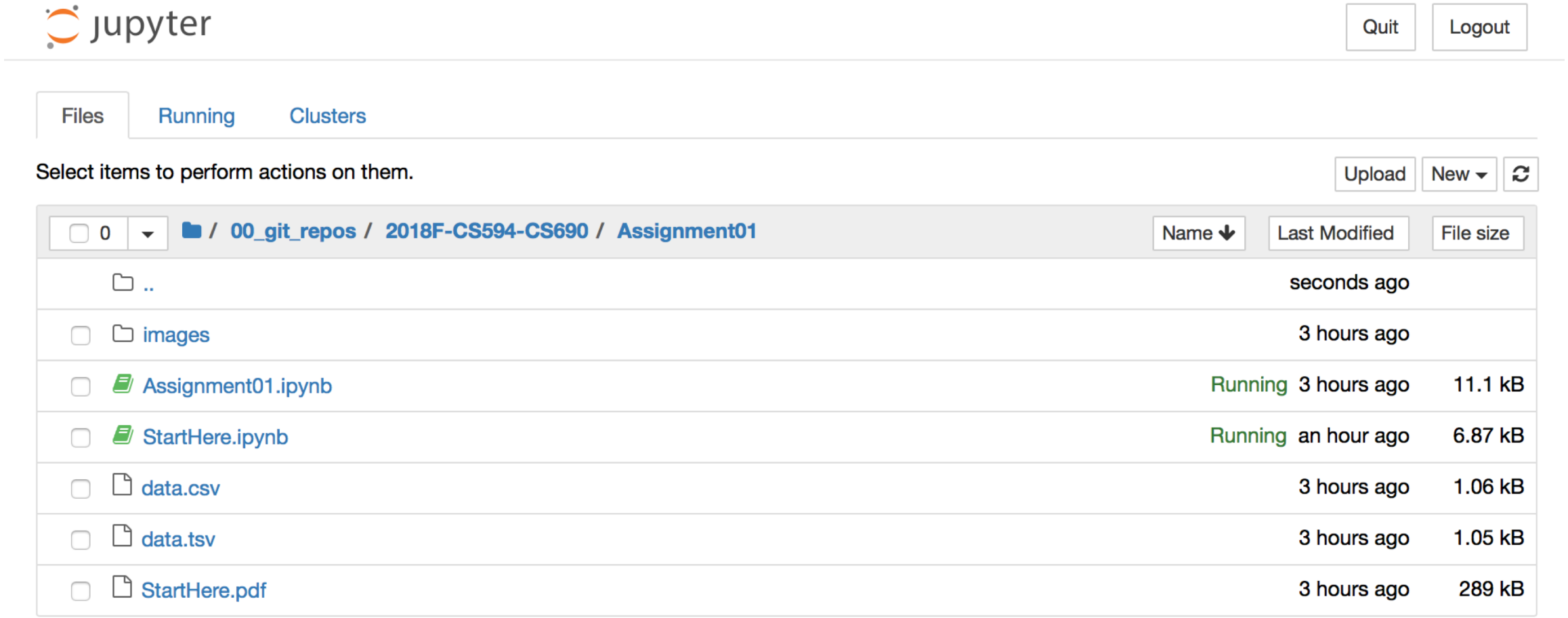
Jupyter

- **Jupyter:** Our notebook for data analytics
- Programming in a browser
  - Create code in a cell – code in edit mode
  - Run code in a cell – code in command mode
  - Write text before and after code cells – markdown
- Your task:
  - Launch Jupyter from Anaconda GUI (or command line)

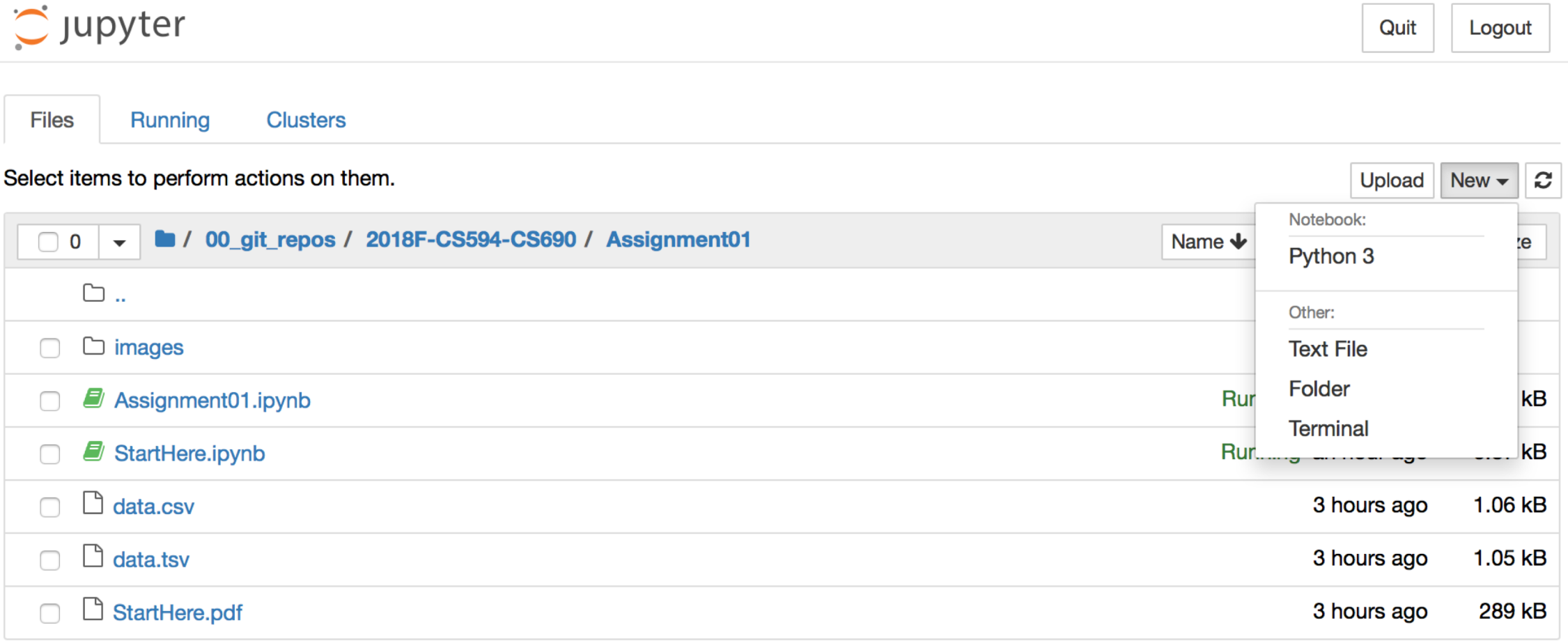
Open your assignment directory



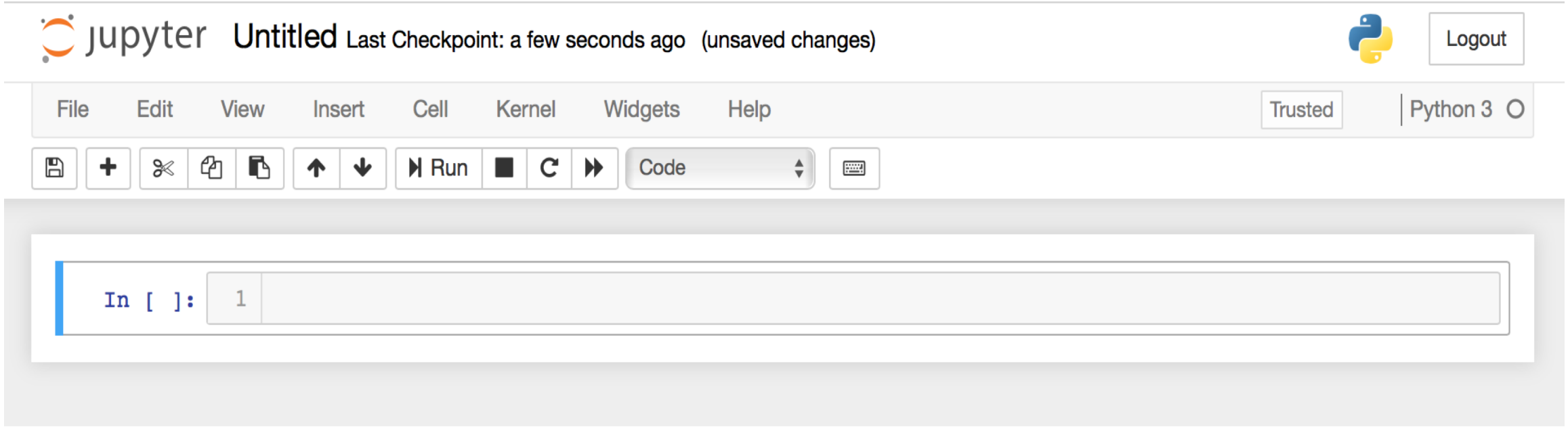
Open your assignment directory



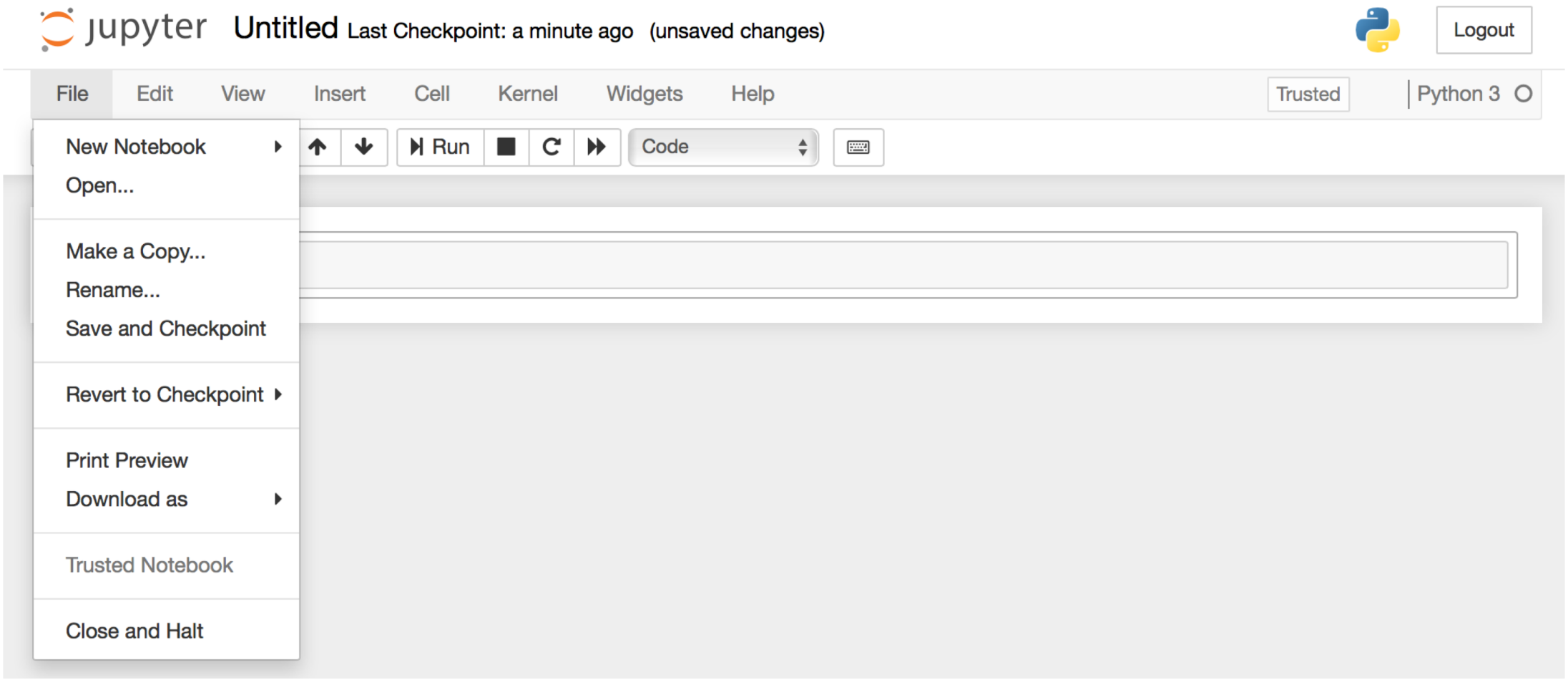
Create your own notebook



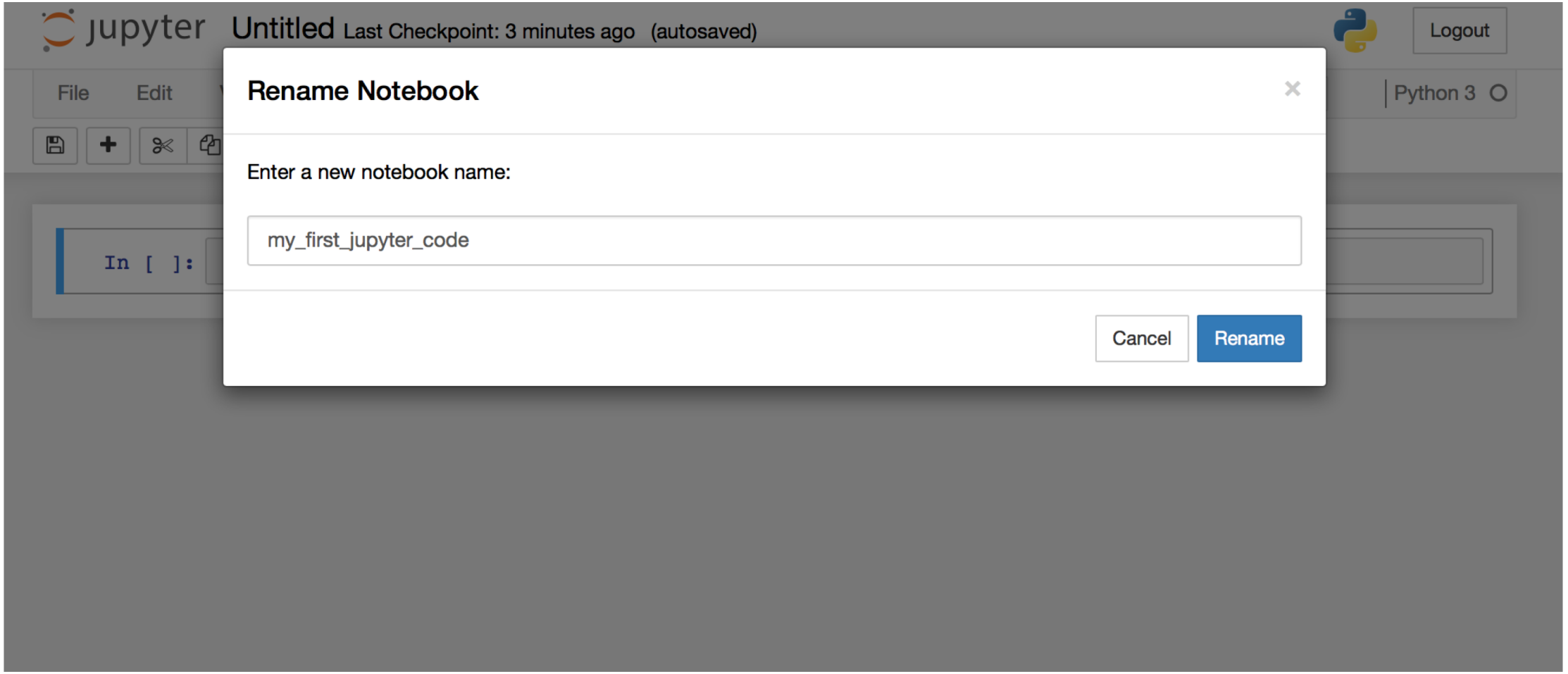
Create your own notebook



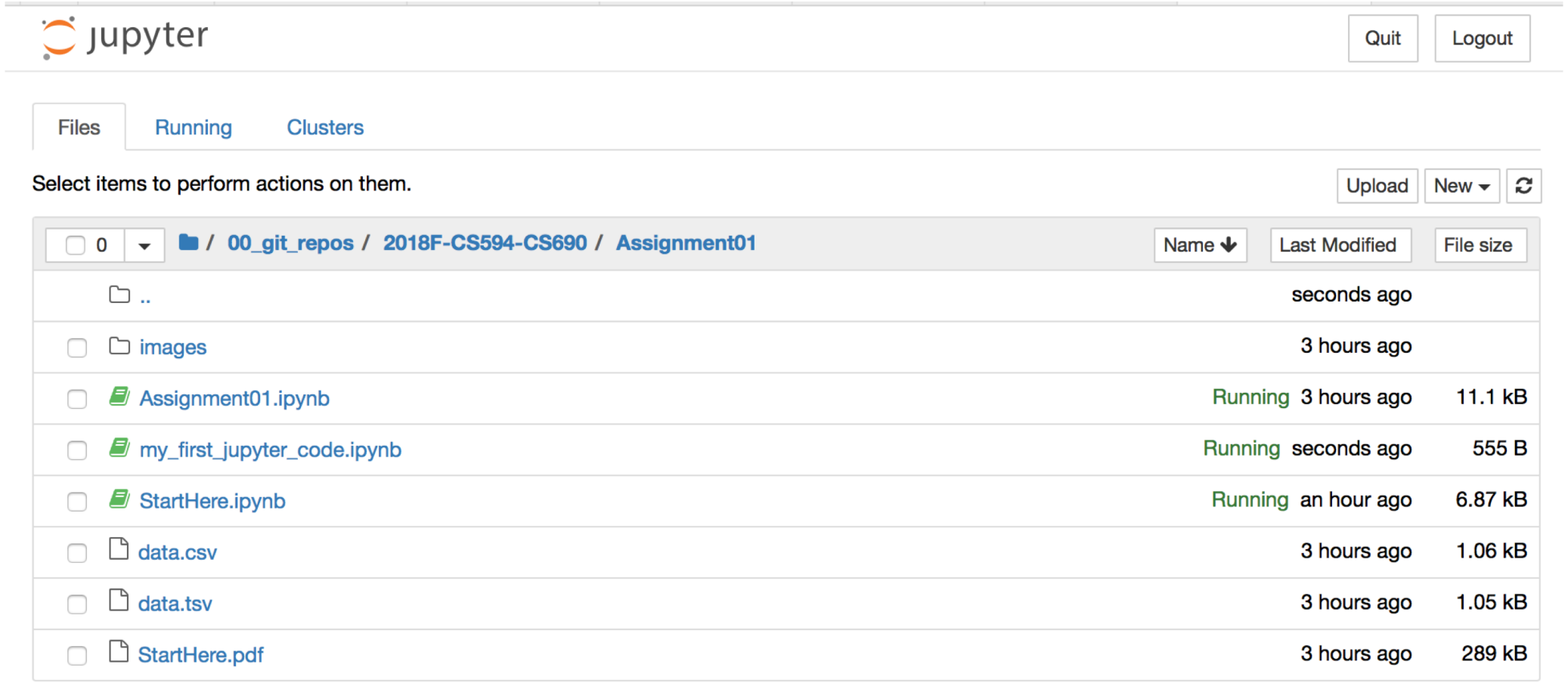
Create your own notebook



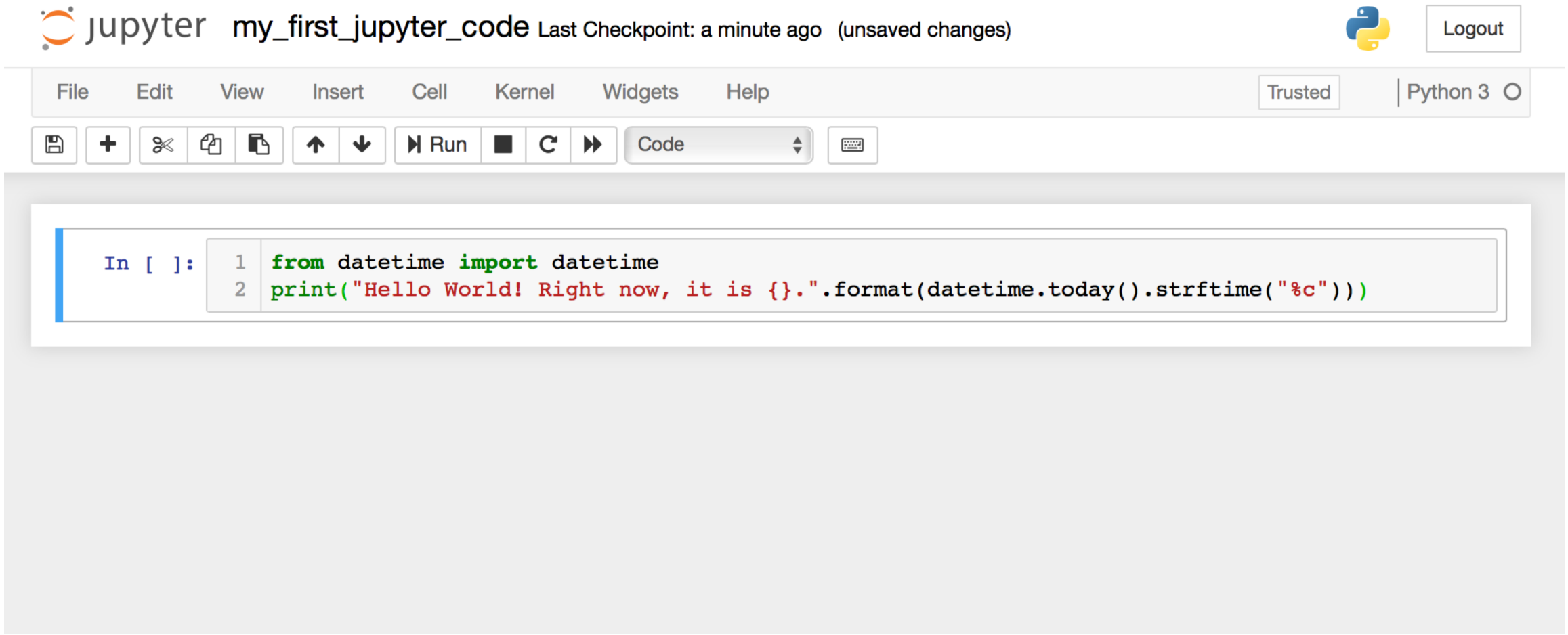
Rename the file



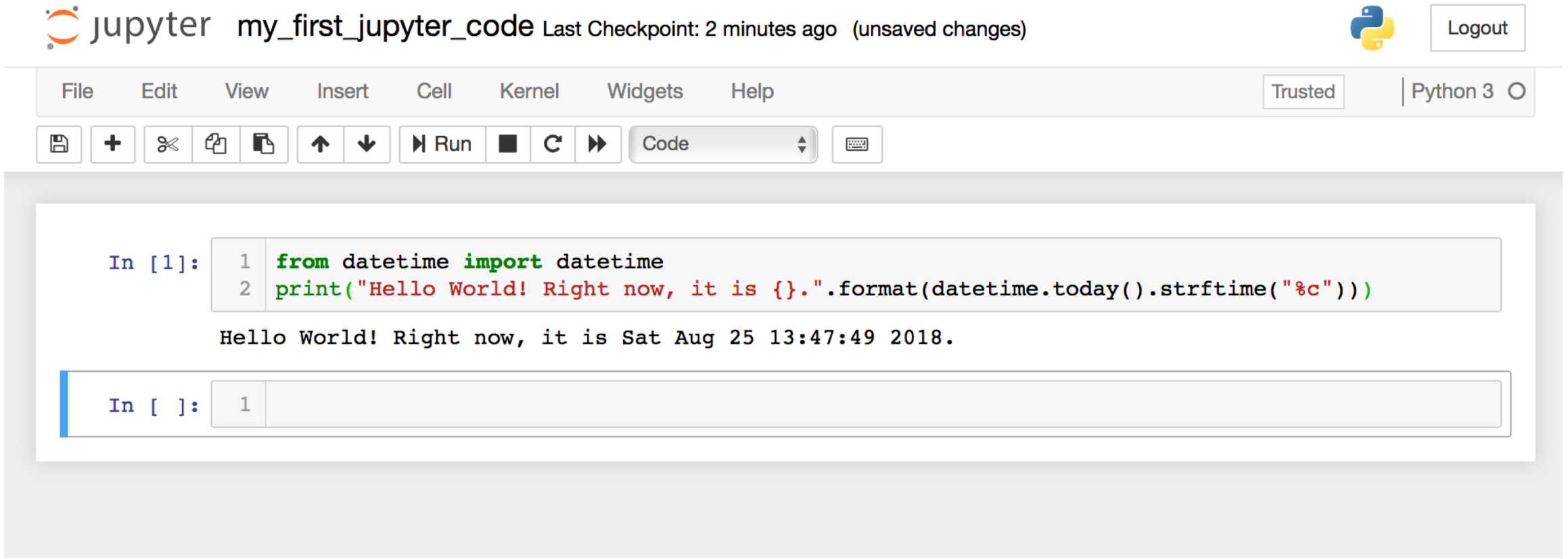
Create your own notebook



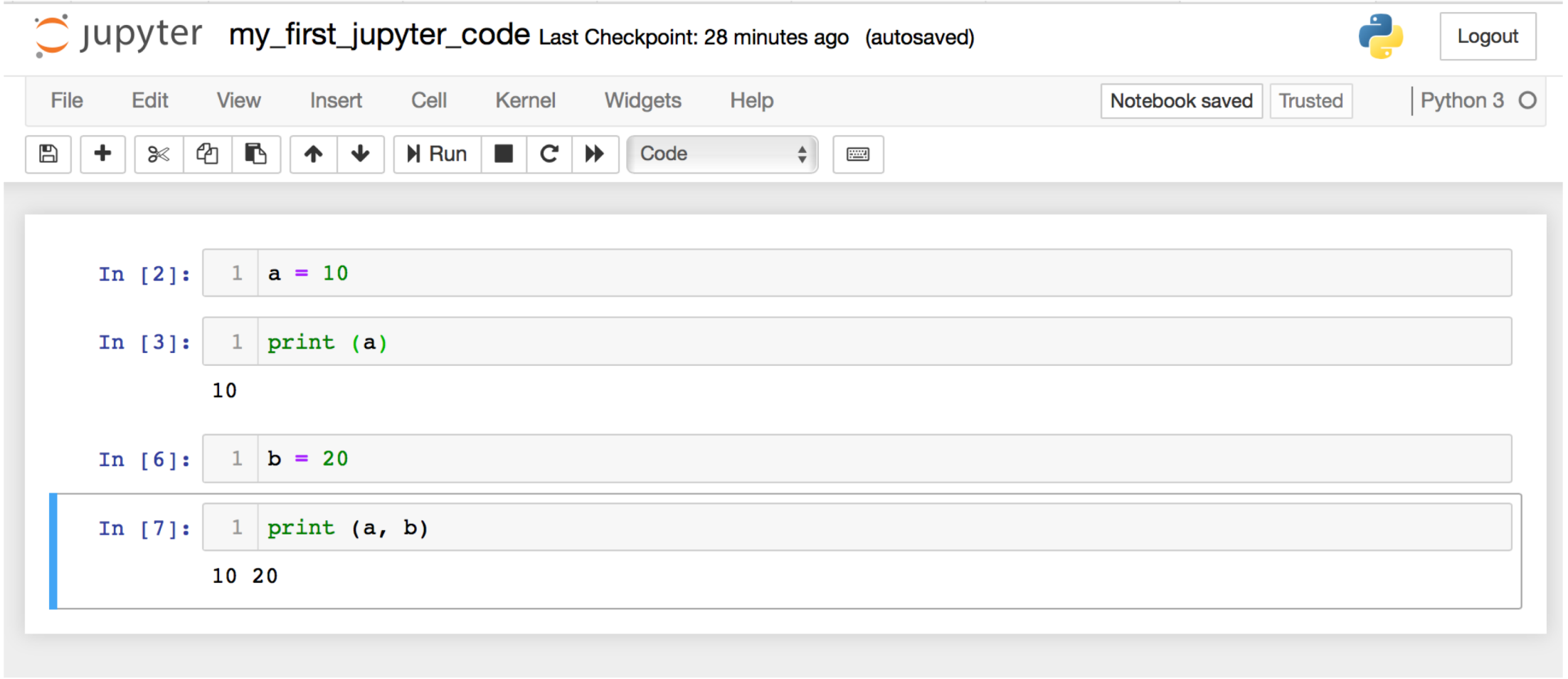
Create a code cell



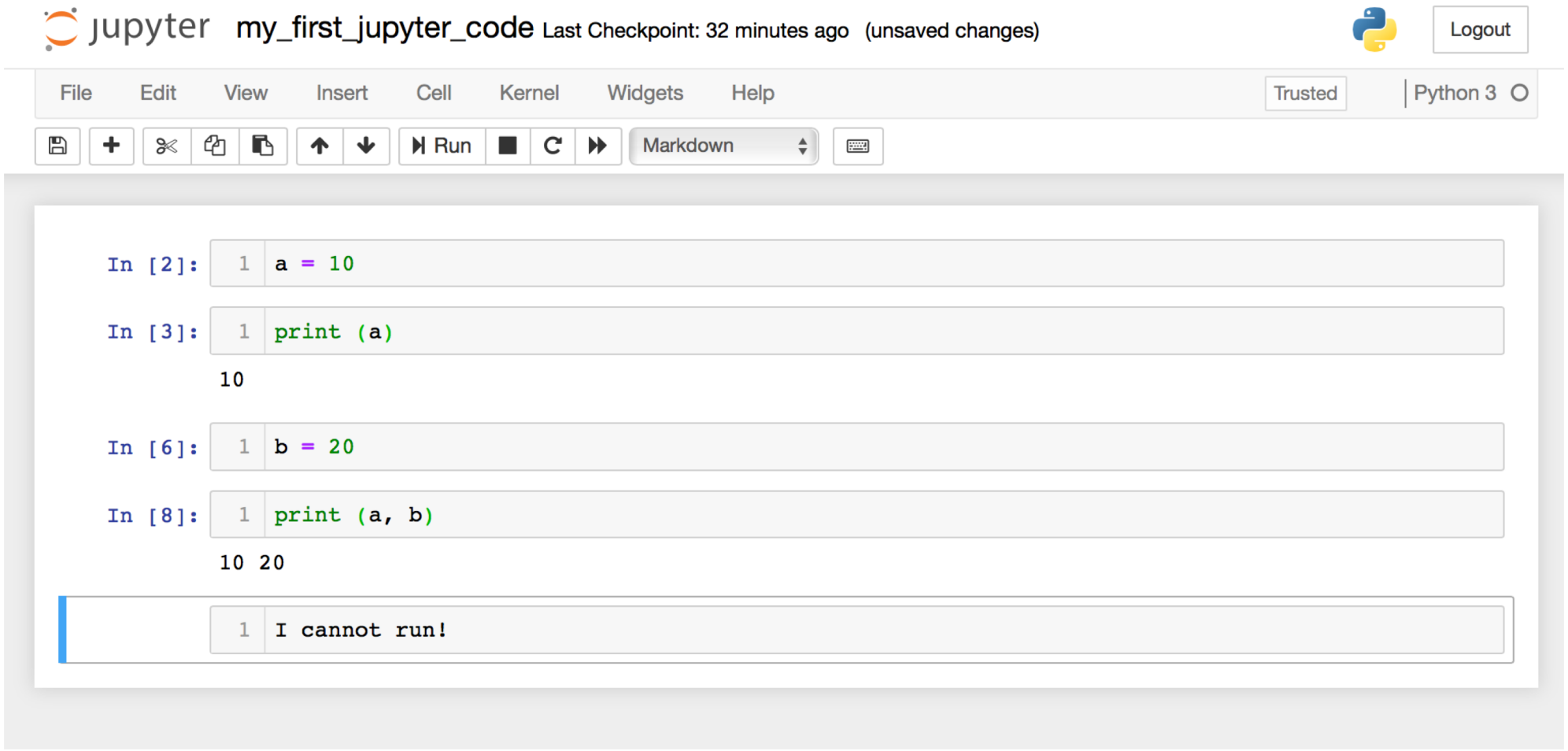
Run your code code



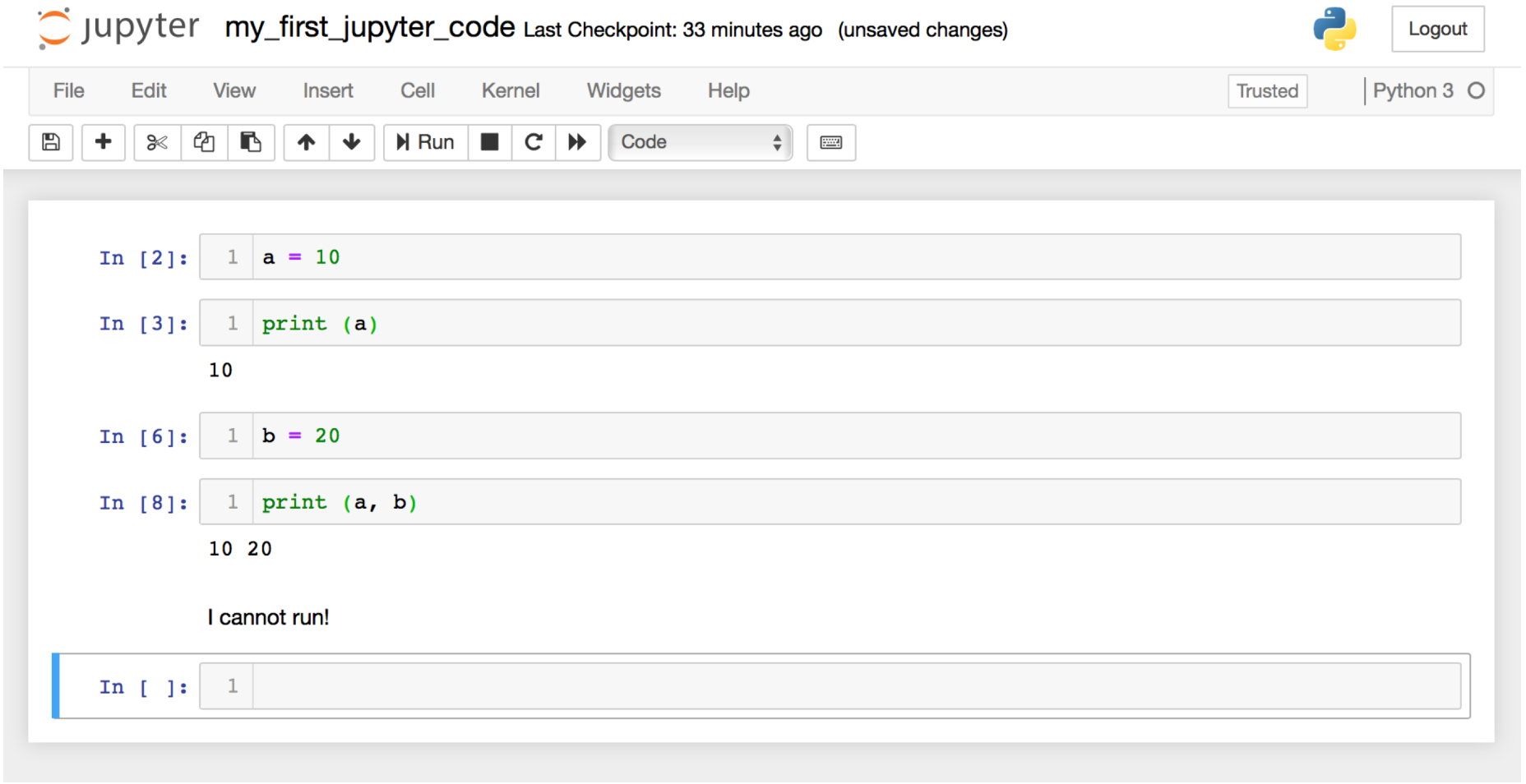
Propagations: from cell to cell



Add text to your notebook



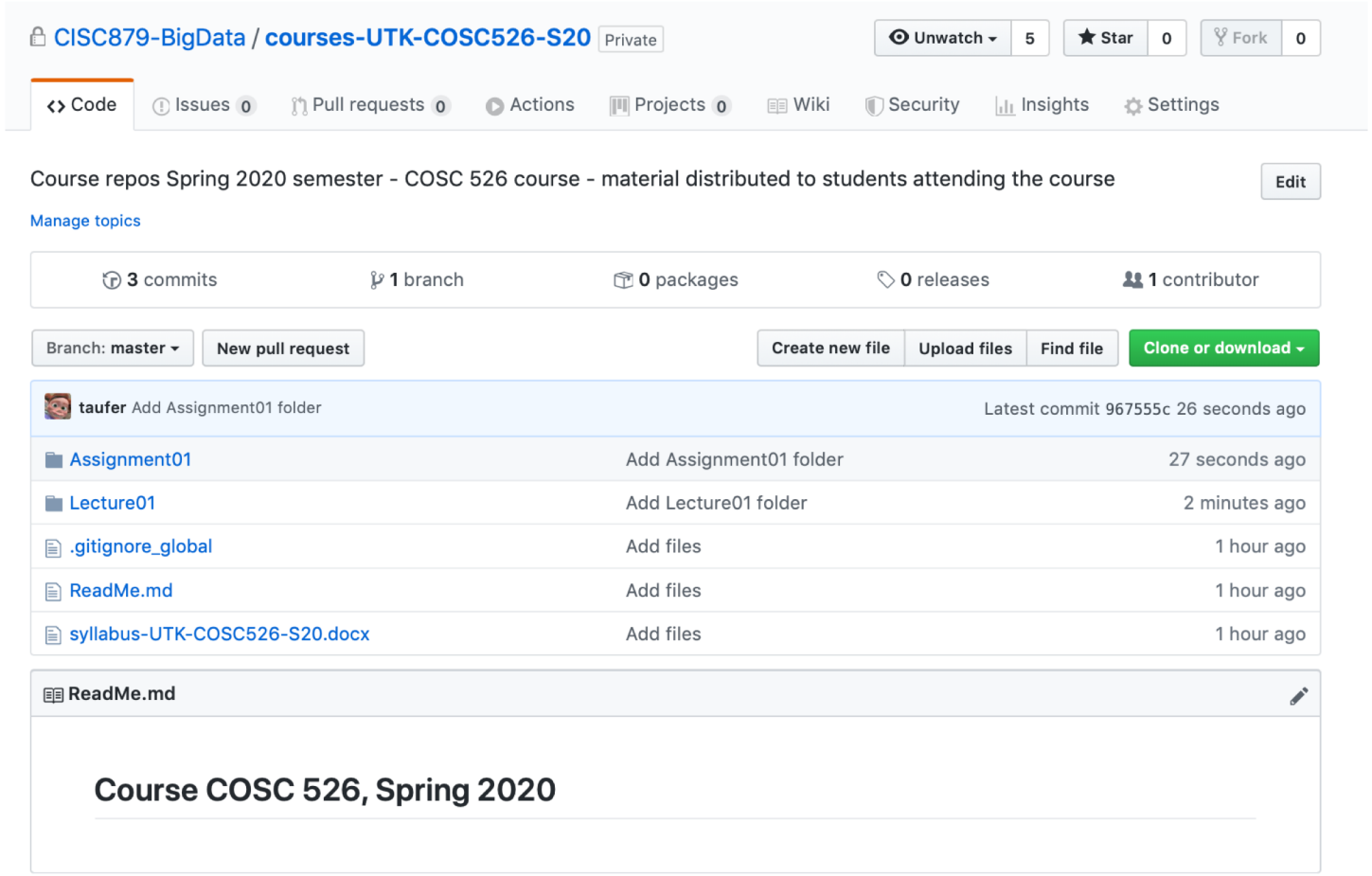
Add text to your notebook



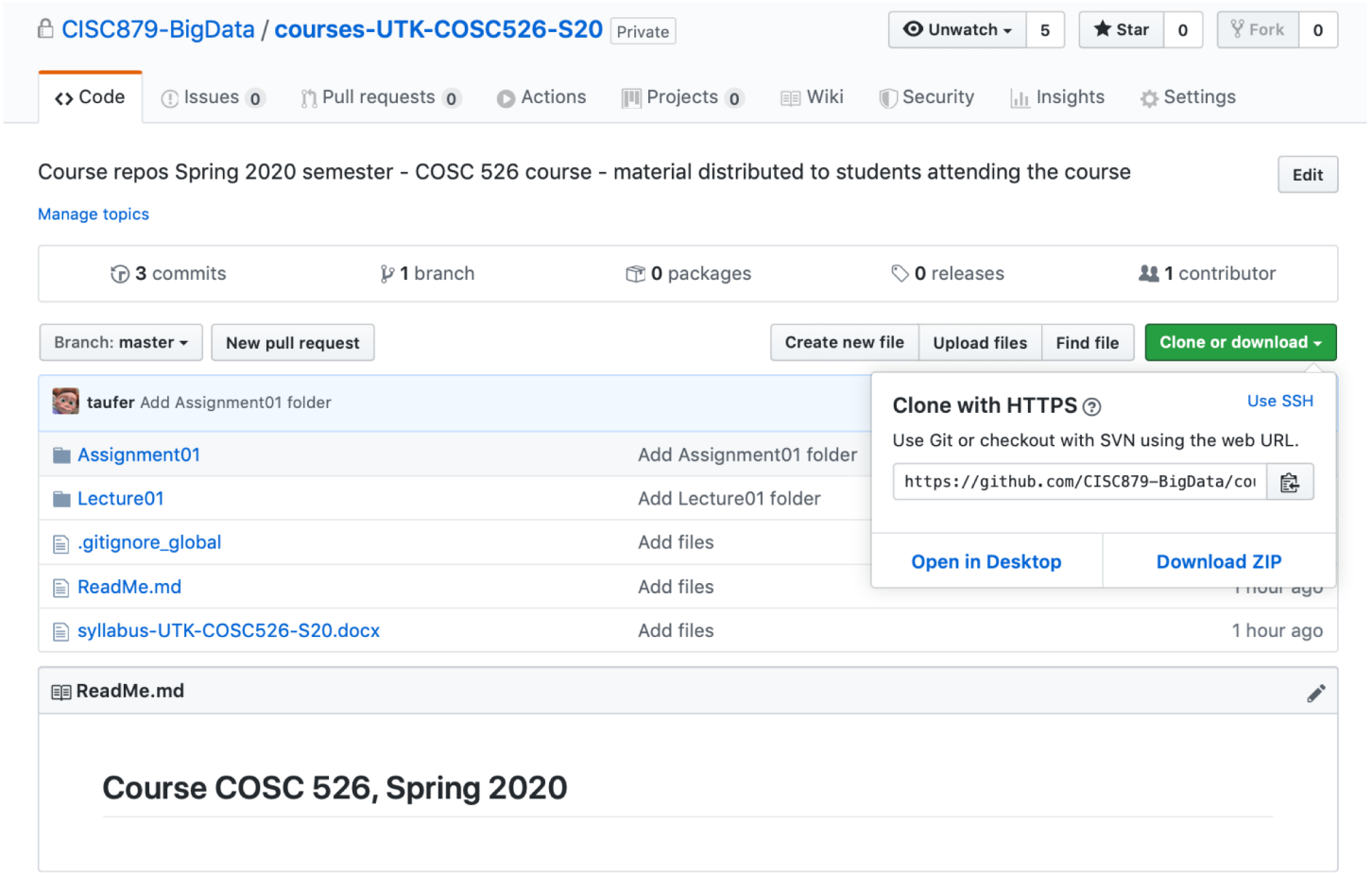
Course Repository

Clone the course repository

https://github.com/CISC879-BigData/courses-UTK-COSC526-S20



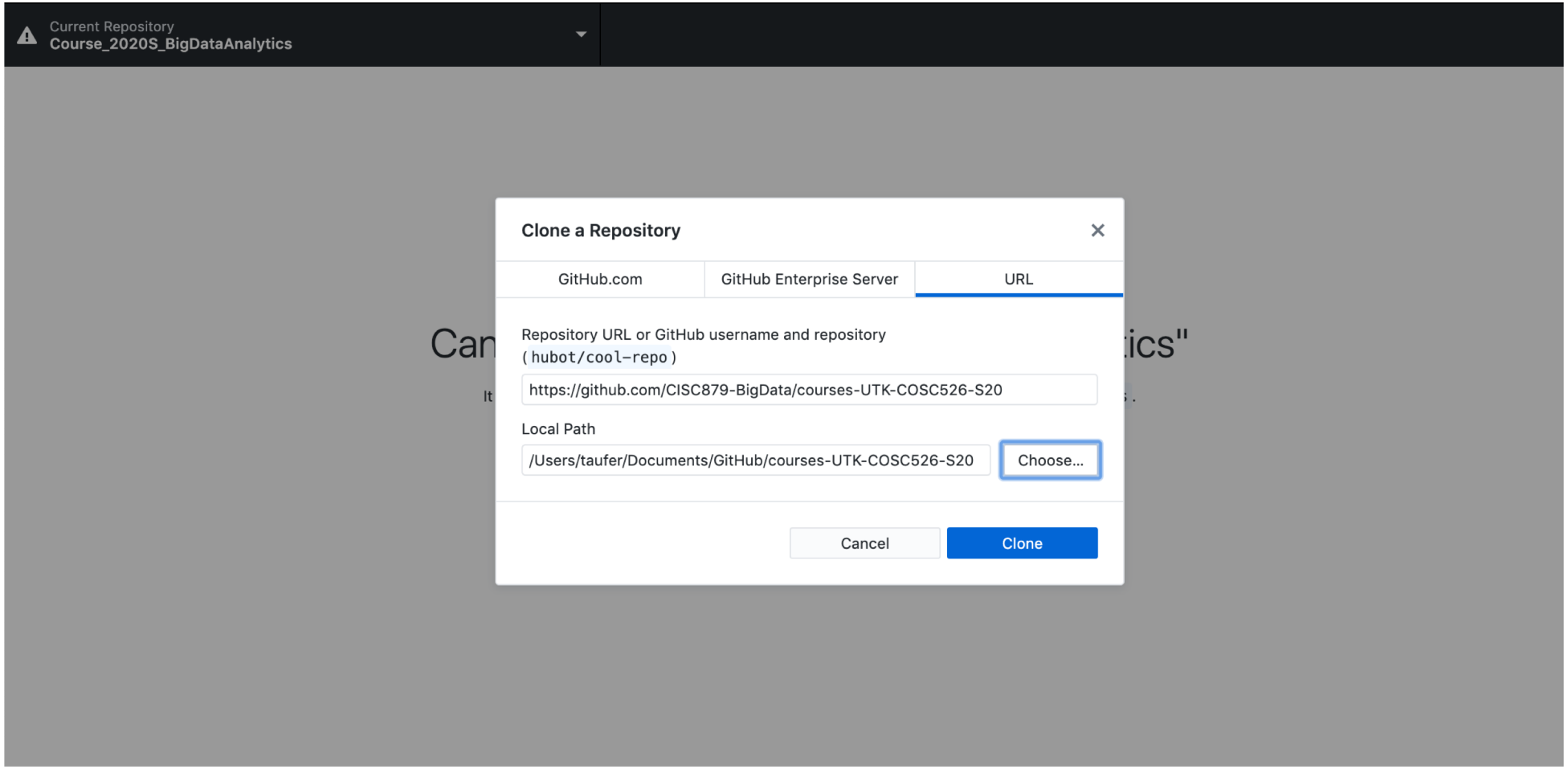
Clone the course repository



Open from CLI

$ git clone https://github.com/CISC879-BigData/courses-UTK-COSC526-S20.git

Open from GitHub desktop



Practical programs

Assignment 1: Dealing with text

- Reading in, parsing, and processing delimiter-separated values stored in files – **comma-separated values (csv)** and **tab-separated values** (**tsv**)
  - Count (and print) the number of rows of data (header is excluded) in the csv file
  - Count (and print) the number of columns of data in the csv file
  - Calculate (and print) the average of the values that are in the "age" column - You can assume each age in the file is an integer, but the average should be calculated as a float

Assignment 1: Dealing with different formats

- Converting the unicode-formatted names into ascii-formatted names
  - Use the provided tranliteration dictionary that maps several common unicode characters to their ascii transliteration to convert the unicode strings to ascii

Assignment 1: Practical tasks

- Create your own GitHub account
- Send your GitHub username to
  https://forms.gle/CKugke8Dzqjm9tQ89

  (NOTE THIS IS DUE ON FRIDAY JAN 17 before 8AM ET)
- Install Git on your laptop

Assignment 1: Answer free-form questions

Answer with codes:

- Your solutions for Problems 1 & 2 probably share a lot of code in common. You might even have copied-and-pasted from Problem 1 into Problem 2. Refactor *parse_delimited_file* to be useful in both problems.
- Are there any pre-built Python packages that could help you solve these problems? If yes, refactor your solutions to use those packages.

Answer with text:

- Describe the challenges you faced in addressing these tasks and how you overcame these challenges.
- Did you work with other students on this assignment? If yes, how did you help them? How did they help you?

For the next week ….

- Get your solution done before our next class
  - This week we are not pushing the solution into your own repos yet

- Next week ….
  - More about private GitHub repos
  - Pull your solution into your GitHub
  - More practice with Jupyter, Python, and problem solving