Contents

Installing Python

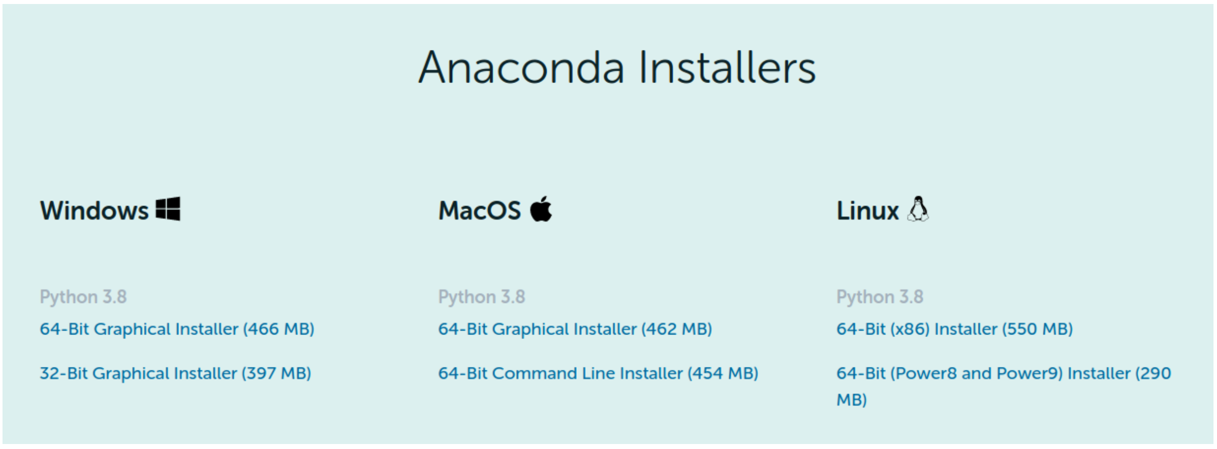
Python is one of the most popular programming languages used today. It is simple to start with, and powerful when mastered. Arguably, it is the most popular language in data science, and close to the top for scientific computing in general (if not at the top). Python owns most of its success to the large number of libraries available to quickly implement and use complex algorithms, classes, and data structures. In this course, we will make great use of two Python powerhouses: Numpy for fast array and matrix operations and Matplotlib for visualization. In other courses (e.g Data Preprocessing) you'll familizarize yourself with other more specialized libraries, such as Pandas (data wrangling) or Scikit-learn (machine learning).

There are several ways to start programming in Python, one of the simplest is to use one of the several available online IDEs. For this course, though, we recommend installing Python locally, since most assignments are more easily handled running a local Jupyter server (more on that later).

The purpose of this page is to help you to install Python and different Python packages into your own computer. Even though it is possible to install Python from their homepage, **we highly recommend using** Anaconda which is an open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. In short, it makes life much easier when installing new tools to your Python.

Install Python on Windows and macOS

There is a convenient graphical installer that can be used to install Anaconda for Windows or Mac. We recommend you install Anaconda by visiting the Anaconda downloads page, moving to the bottom of the page, and clicking on the button to install the latest Python 3 version of Anaconda, as shown below. For most modern computers (> 10 years), the 64 bit version should be preferred.


Anaconda Installers

Windows	MacOS	Linux
Python 3.8	Python 3.8	Python 3.8
64-Bit Graphical Installer (466 MB)	64-Bit Graphical Installer (462 MB)	64-Bit (x86) Installer (550 MB)
32-Bit Graphical Installer (397 MB)	64-Bit Command Line Installer (454 MB)	64-Bit (Power8 and Power9) Installer (290 MB)

Figure 1: Anaconda installation

Install Python on Linux

The following have been tested on Ubuntu 18.04. Run the following command (note the name of the installer is the most recent one as of the writing of this tutorial, you may need to update it):

```
# Download and install Anaconda
wget https://repo.anaconda.com/archive/Anaconda3-2020.07-Linux-x86_64.
    sh
bash Anaconda3-2020.07-Linux-x86_64.sh

# Add Anaconda installation permanently to PATH variable
nano ~/.bashrc

# Add following line at the end of the file and save (EDIT ACCORDING
    YOUR INSTALLATION PATH)
export PATH=$PATH:/PATH_TO_ANACONDA/anaconda3/bin
```

Checking everything works

On Linux/macOS, open a terminal, on Windows, open the Anaconda Prompt. Run conda `conda -V` to print the conda version. `cd` to your working directory, type `jupyter-lab`. A browser window should pop up with the Jupyter Lab interface. You are ready to go. To close Jupyter Lab, navigate to `File/Shut Down` in the upper left corner.

Note: It is important to `cd` into the desired working directory before running Jupyter Lab, otherwise Jupyter will open in the default directory and you will have a hard time accessing your file.

How to find out which conda -command to use when installing a package?

The easiest way

The first thing to try when installing a new module `X` is to run in a command prompt (as admin) following command (here we try to install a hypothetical module called X)

```
conda install X
```

In most cases this approach works but sometimes you get errors like (example when installing a module called shapely):

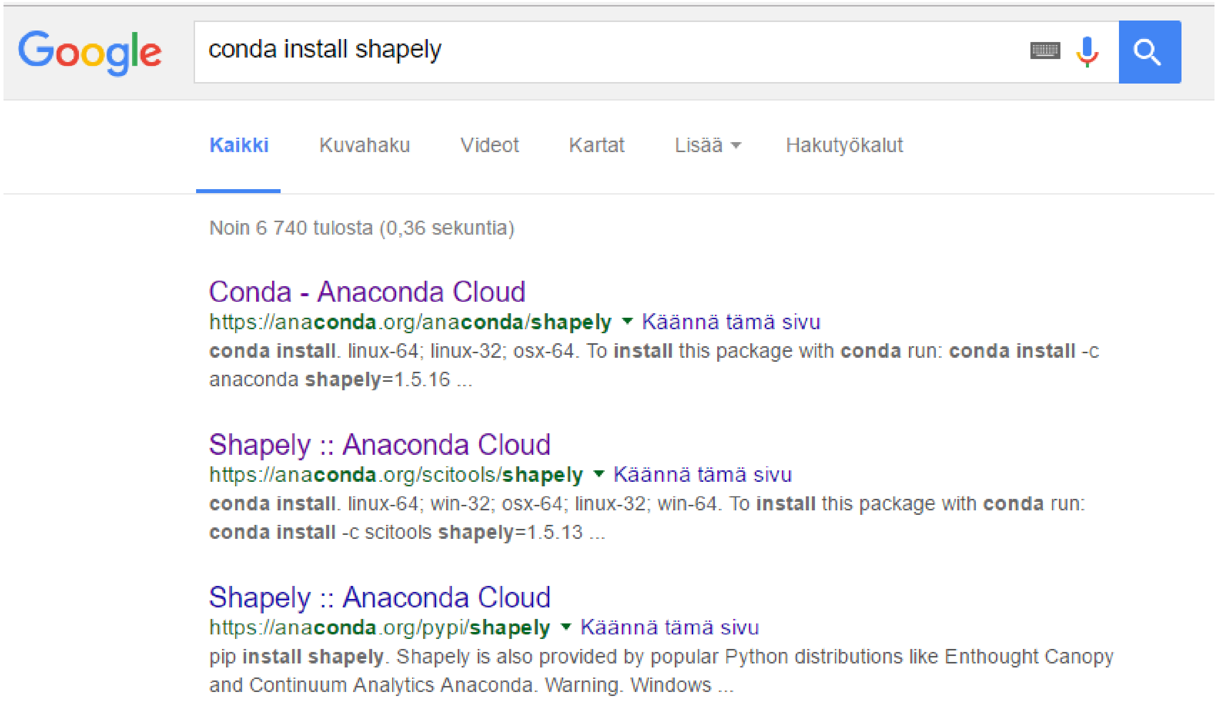
```
C:\WINDOWS\system32>conda install shapely
Using Anaconda API: https://api.anaconda.org
Fetching package metadata .........
Solving package specifications: .
Error: Package missing in current win-64 channels:
    - shapely

You can search for packages on anaconda.org with

    anaconda search -t conda shapely
```

In this case conda was not able to find the shapely module from the typical channel it uses for downloading the module.

Alternative way to install packages if typical doesn't work

If `conda install` command was not able to install the package you were interested in there is an alternative way to do it by taking advantage of different conda distribution channels that are maintained by programmers themselves. An easy way to find the right command to install a package from these alternative conda distribution channels is to Google it.

Let's find our way to install the Shapely module by typing following query to Google:



Noin 6 740 tulosta (0,36 sekuntia)

Conda - Anaconda Cloud
https://anaconda.org/anaconda/shapely ▾ Käännä tämä sivu
conda install. linux-64; linux-32; osx-64. To install this package with conda run: conda install -c anaconda shapely=1.5.16 ...

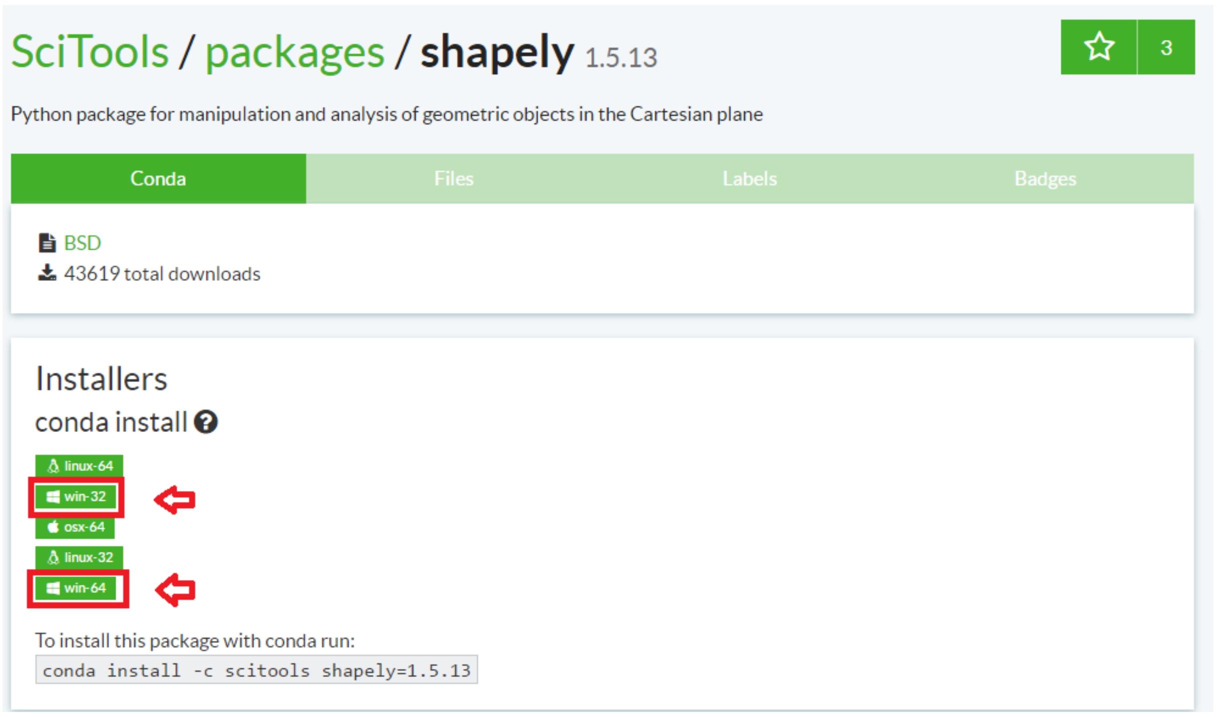
Shapely :: Anaconda Cloud
https://anaconda.org/scitools/shapely ▾ Käännä tämä sivu
conda install. linux-64; win-32; osx-64; linux-32; win-64. To install this package with conda run: conda install -c scitools shapely=1.5.13 ...

Shapely :: Anaconda Cloud
https://anaconda.org/pypi/shapely ▾ Käännä tämä sivu
pip install shapely. Shapely is also provided by popular Python distributions like Enthought Canopy and Continuum Analytics Anaconda. Warning. Windows ...

Here, we can see that we have different pages showing how to install `Shapely` using conda package manager.

Which one of them is the correct one to use?

We need to check the operating system banners and if you find a logo of the operating system of your computer, that is the one to use! Thus, in our case the first page that Google gives does not work in Windows but the second one does, as it has Windows logo on it:

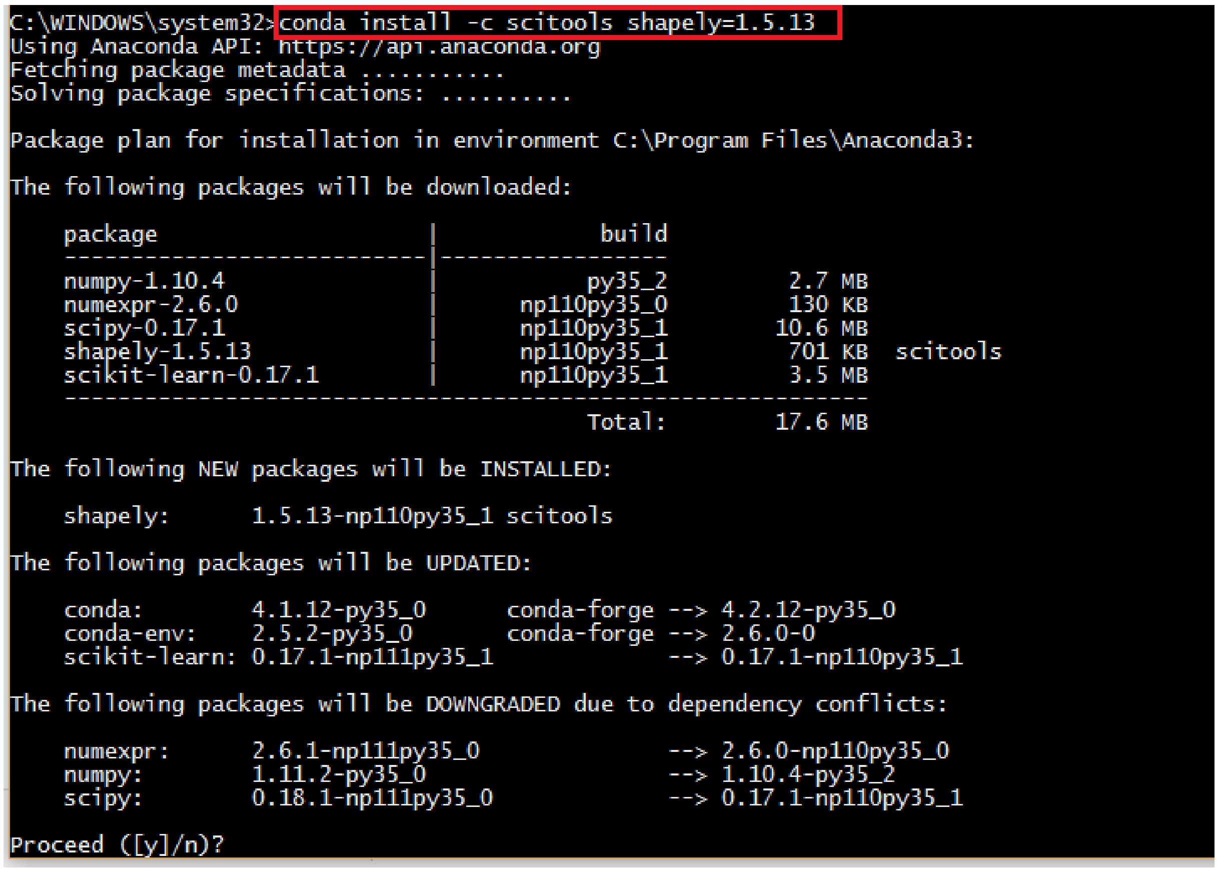


Python package for manipulation and analysis of geometric objects in the Cartesian plane

To install this package with conda run:

```
conda install -c scitools shapely=1.5.13
```

From here we can get the correct installation command for conda and it works!



```
C:\WINDOWS\system32>conda install -c scitools shapely=1.5.13
Using Anaconda API: https://api.anaconda.org
Fetching package metadata ...........
Solving package specifications: ..........

Package plan for installation in environment C:\Program Files\Anaconda3:

The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    numpy-1.10.4               |             py35_2         2.7 MB
    numexpr-2.6.0              |       np110py35_0         130 KB
    scipy-0.17.1               |       np110py35_1        10.6 MB
    shapely-1.5.13             |       np110py35_1         701 KB  scitools
    scikit-learn-0.17.1        |       np110py35_1         3.5 MB
    ------------------------------------------------------------
                                           Total:        17.6 MB

The following NEW packages will be INSTALLED:

    shapely:      1.5.13-np110py35_1 scitools

The following packages will be UPDATED:

    conda:        4.1.12-py35_0      conda-forge --> 4.2.12-py35_0
    conda-env:    2.5.2-py35_0       conda-forge --> 2.6.0-0
    scikit-learn: 0.17.1-np111py35_1             --> 0.17.1-np110py35_1

The following packages will be DOWNGRADED due to dependency conflicts:

    numexpr:      2.6.1-np111py35_0              --> 2.6.0-np110py35_0
    numpy:        1.11.2-py35_0                  --> 1.10.4-py35_2
    scipy:        0.18.1-np111py35_0             --> 0.17.1-np110py35_1

Proceed ([y]/n)?
```

You can follow these steps similarly for all of the other Python modules that you are interested to install.

Git and version control

To deliver and submit assignments we will use **GitHub** and **git**.

Git is a distributed version control system. A version control system (or source code management system) is a tool for storing the source files of a project, distributing them to developers working on the project, and coordinating changes made during development. A version control system stores not just the current state of the project, but all past states as well. This makes it an important tool both for collaboration and for historical investigation. (Often finding when a bug was introduced makes it much easier to diagnose and fix it.)

A distributed version control system does all this without requiring a central master server. Instead, each copy of the project contains the full change history and copies can be synchronized with each other to share updates.

Git is one of several available distributed version control systems; it was originally developed by Linus Torvalds for development of the Linux kernel. For information on how to use git, see A Guide to Git.

Installing git

Linux

Most Linux distributions provide git packages right from the repositories. In Ubuntu, for example, you only need to run

```
sudo apt install git
```

Windows

For Windows, we recommend using Git Bash, which provides a BASH emulation used to run Git from the command line. The BASH emulation behaves just like the "git" command in LINUX and UNIX environments. You can download Git Bash from its home page, at https://gitforwindows.org/.

After installing, running the Git Bash program will open a terminal window with all the available git commands.

macOS

There are several options for installing Git on macOS. Note that any non-source distributions are provided by third parties, and may not be up to date with the latest source release. For several options, see https://git-scm.com/download/mac.

One time git configuration

To start using git, you need to set up your name and mail, run the following, adjusting the parameters for your case:

```
git config --global user.name "FIRST_NAME LAST_NAME"
git config --global user.email "ID@upy.edu.mx"
```

GitHub Classroom

Disclaimer: This section is excerpted from http://katrompas.accprofessors.com/github-classroom/.

GitHub Education and GitHub Classroom is a product and service offered by GitHub for students and educators to learn and teach version control, and also to facilitate training in professional software development. For this course. students will be required to use GitHub Classroom and version control. Students will maintain private student repositories and submit all code via GitHub.

Prepare

The first thing you'll need to do is to get a GitHub education account . You will fall into one of the following situations. Choose your case from the following list to get yourself set up with GitHub Education.

- **You already have a GitHub account with your school email and you already have access to GitHub Education**: You're all set.
- **You have a GitHub account with your student email, but do not have access to GitHub Education**: Go to GitHub Education for Students and sign up for student benefits (you can skip the extra Student Developer Pack but it has more benefits you might find useful).
- **You have a GitHub account but not with your student email.** You have two choices to proceed:
  - Add your student email to your current account.
  - Make a new account with your student email.

  Once your student email is registered and verified, Go to GitHub Education for Students and sign up for student benefits (you can skip the extra Student Developer Pack but it has more benefits you might find useful).
- **You do not have a GitHub account**: Make a new account with your student email. Once your student email is registered and verified, Go to GitHub Education for Students and sign up for student benefits (you can skip the extra Student Developer Pack but it has more benefits you might find useful).

IMPORTANT: Once you have an account, you should add an extra email to your account in case you lose access to your school email.

General Tutorial

Once you are set up with GitHub, you can read and complete the tutorial, which can be found in the group files. The tutorial covers more things than those we'll need for this course, but it is a good introduction to basic version control operations and concepts.

This tutorial assumes you start local and push remote. The assignments in class will be the reverse, they start remote and pull local. You need to know how to do both to be a developer. The specifics of how we will use GitHub for assignments in class is detailed next.

General GitHub Process for Assignments

The **general** process for working with GitHub Classroom to access and complete assignments is as follows:

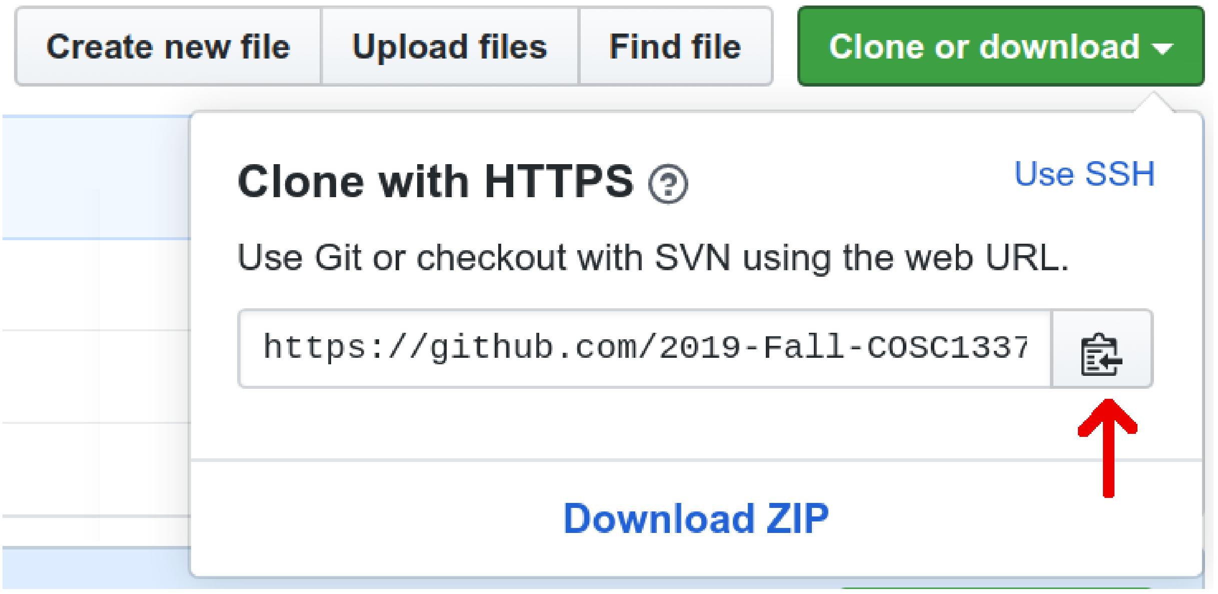
- You will have a classroom area in GitHub for your class. You will be given that link at the start of the semester.
- For each assignment you will be given an invitation link through a Teams assignment.
- Accepting the invitation will create a private remote repository with assignment specifications and optional starter code.
- You will create a local area on your computer to work.
- In that local area you will initialize git, connect your repo, and you will pull code back to your local development area from your remote repo.
- You will work in your local area, frequently committing code, and using version control and your repo normally. **While working on assignments, you must commit regularly and intelligently.** This means you commit with every small, logical, well-test block of code. For example, when you have completed one function or method, and tested it very well, commit it. Another example, you fix one bug, you test the fix very well, commit it. Your commit messages need to be informative and explain fully what you just added/fixed. For example, *"Fixed bug in function get$_{array}$(). There was a one-off error causing the function to overrun the array. Fixed it by changing the <= to < in the for loop."* Do not make commit messages like, "done" or "commiting code" or "fixed bug" or "final commit." Commit messages should tell a "story" about what you did, how you did it, your thought process, and why it was necessary to perform each step.
- When you are ready for grading, you will push all your code to the remote repo and submit the assignment on Teams. Submitting the assignment in Teams is simply your signal to me that you are ready for grading.
- While working, you can push as much or as little as you like, however for grading purposes you must push all the code you want graded. All code is graded on GitHub. **If it's not there, it will not be graded.**
- If you make changes after you submit in Teams but before I grade it, that's fine, but you cannot be sure when I will grade so try to leave your repo stable until it's graded.

- Once your assignment is graded you cannot re-submit. That is why we will dedicate regular class hours to go over your work if you need help. Also, most assignments will let you know if you are getting the expected answer, **don't ignore these self-checks.**

Assignment Specific Instructions

The following are the specific instructions which will accomplish the process explained above. Follow these instructions for each assignment. **Note: Failure to commit regularly and intelligently may result in your assignment being rejected and being assigned a grade of zero. This is a required part of every assignment.**

- Set-up version control and a course directory **prior** to writing **any** code. This directory will hold each assignment's repository. Try to keep things ordered.
- Your invitation to the assignment is provided with the Teams assignment.
- Click that link in the assignment. A repo with starter files will be created for you.
- Clone your repository to your local directory. Get your specific url from your repo by clicking the "clone or download" button in your repo and then using the copy button.



Use Git or checkout with SVN using the web URL.

```
https://github.com/2019-Fall-COSC1337
```

- In a terminal, cd to the course directory and run:

```
git clone https://<your-repo>.git
```

A new directory containing the assignment code will be created for you to work with.

- After completing some part of the assignment, or fixing a bug, it is good idea to commit your work. You'll need to commit at least once, before submitting the assignment. Committing adds all your changes to the repository's shared history. To commit run:

  ```
  git commit -a -m "SOME USEFUL MESSAGE"
  ```

- Finally, when done with the assignment, you need to *push* your changes to the GitHub repository for grading. This will replicate all your work in GitHub, and sync your local repo to the GitHub repo. To push do:

  ```
  git push origin master
  ```

  When you are ready for grading, do your final commits, do a final push, and submit the corresponding Teams assignment.

Basic vocabulary of Version Control

Few basic terms that are used often when discussing about version control (not exhaustive).

- **Repository** = a location where all the files for a particular project are stored, usually abbreviated to "repo." Each project will have its own repo, which is usually located on a server and can be accessed by a unique URL (a link to GitHub page for example).
- **Commit** = To commit is to write or merge the changes made in the working copy back to the repository. Whe you commit, you are basically taking a "snapshot" of your repository at that point in time, giving you a checkpoint to which you can reevaluate or restore your project to any previous state. The terms 'commit' or 'checkin' can also be used as nouns to describe the new revision that is created as a result of committing.
- **Revision / version** = A revision or a version is any change in made in any form to a document(s).
- **Clone** = Cloning means creating a repository containing the revisions from another repository. This is equivalent to pushing or pulling into an empty (newly initialized) repository. As a noun, two repositories can be said to be clones if they are kept synchronized, and contain the same revisions.
- **Pull / push** = Copy revisions from one repository into another. Pull is initiated by the receiving repository, while push is initiated by the source. Fetch is sometimes used as a synonym for pull, or to mean a pull followed by an update.
- **Merge** = A merge or integration is an operation in which two sets of changes are applied to a file or set of files.

Resources

In addition to our course, there are countless of excellent books, tutorials and examples related to programming in Python. Here we list some good places to look for further information.

Recommended readings

- Array programming with NumPy, Nature (2020)

Python Books

- Recommended textbooks on Python focusing on data applications:
  - VanderPlas, J. (2016) Python Data Science Handbook. O'Reilly Media
  - Scopatz and Huff (2015) Effective Computation in Physics. O'Reilly Media
  - McKinney, W. (2017) Python for Data Analysis. O'Reilly Media.

Python tutorials and documentation

- Whirlwind Tour of Python
- NumPy Documentation, including the recommended Quickstart tutorial
- SciPy reference guide, including the SciPy Tutorial and the API Reference
- SciPy Cookbook
- SciPy Lecture Notes
- Matplotlib user guide

Git + Github tutorials

- Online "Try-Git" tutorial (learn Git in your browser)
- Git simple guide ("no deep shit") tutorial
- Software Carpentry's Git novice tutorial
- Git official documentation
- Screencast series in Youtube for learning GitHub
- Tutorial on few extra features of GitHub not (most probably) covered in this course (e.g. branch, pull-request, merge)
- A TechCrunch article about 'What is GitHub Anyway?'
- A list of resources for learning Git and GitHub

Sources and references

- Geo-Python course at https://geo-python.github.io/site/
- http://katrompas.accprofessors.com/github-classroom/