Python Setup - Installation and Environment

1 Contact

If you have questions about this document you can find our team webpage with contact info at https://up-rs-esp.uni-potsdam.de/. Most questions can be addressed to Dr. Harald Schernthanner (hschernt@uni-potsdam.de).

2 Objectives

This document describes in detail the installation of a Python environment for courses in the Remote Sensing Master's. The document focuses on the course *Data Analysis and Statistics*, but is applicable to our other Python-based courses. We suggest using Ubuntu or some other Linux-based distribution (Mac is Linux-based) as these are the most flexible operating systems, but Windows OS will work as well. We also discuss the Jupyter Notebook environment that we use for lectures and homework.

A separate document titled *Geological Remote Sensing SSH Setup* walks through how to access and work remotely using the PC Pool at Potsdam with `ssh` logins, and another document titled *Geological Remote Sensing Additional Software* contains some additional software to install (e.g., QGIS).

3 Python Installation

3.1 Why We Choose Python

The main reasons that we choose Python in our classes are: (1) it's open source requiring no license or fees and (2) it's popular in the geoscience and remote sensing communities. Beyond this, Python is versatile, easy to read and learn, and has excellent support online. Most questions or issues can be solved with web searches and on sites like StackOverflow.

3.2 Basic Terminology

We will define more terms as we go through our courses, but these are the basics to get you started and following along:

Python is a ***programming language***, which allows a user to pass human-readable instructions ("add 2+2") through an interpreter that converts them to machine-readable code ("0010110000"). Other popular, open-source languages for data analysis include R, Java, and C++. Many ***packages*** exist for Python, which serve as toolboxes containing functions for specific tasks like linear algebra, machine learning, and plotting figures. These packages are developed by one or more users and can be installed to a given Python ***virtual environment***. Most operating systems include some native install of Python, but it is wise to keep separate Python environments for different tasks (e.g., data analysis). These virtual environments (and the various Python packages within them) are kept isolated from one another thus preventing any package or system conflicts.

Note: If you want a deeper understanding of Python you're encouraged to have a look through the official Python Tutorial.

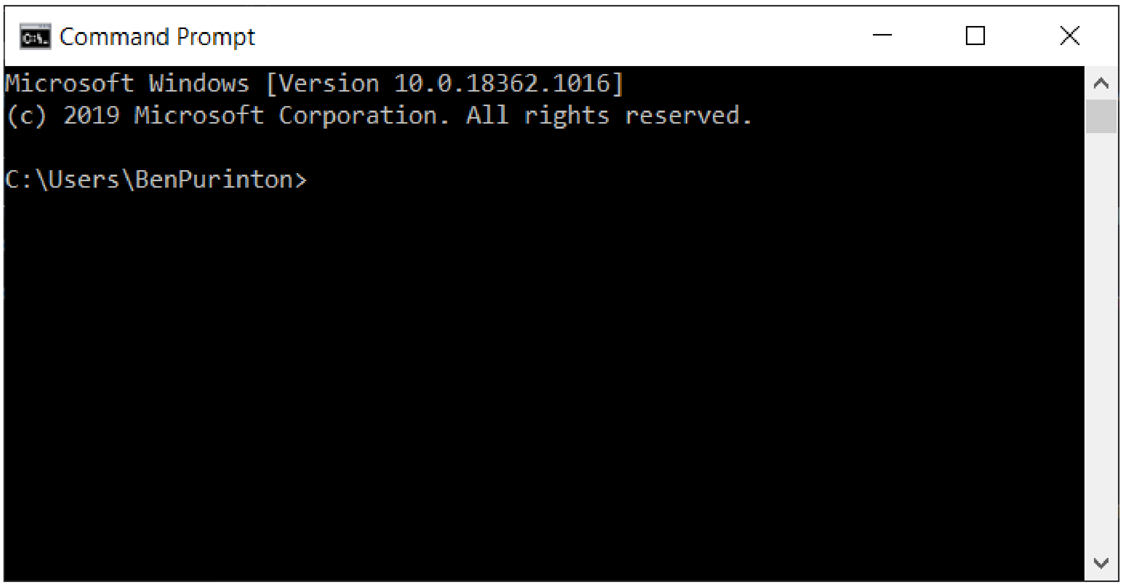
3.3 Environment Manager: Anaconda vs. Miniconda

Anaconda is a popular Python environment manager for data science. Downloading Anaconda provides many scientific packages automatically and allows users to install new environments and additional packages easily using either a desktop application or a built in Anaconda shell. In our courses we prefer to use the light-weight version of Anaconda, called Miniconda. Miniconda is a much smaller version of Anaconda that only contains the command-line tool `conda`, Python, and a small number of useful packages like `pip`. We'll discuss virtual environment and package installs using `conda` and `pip` below.

3.4 The Command Line

Before we move on, we need to familiarize ourselves with the computer's ***command line*** (sometimes called a ***shell*** or ***terminal***). The command line of any OS allows you to pass instructions to the computer in the form of lines of text. You can think of command-line instructions as an alternative to clicking in a graphical user interface (***GUI***). Among many things, you can: open programs; move through your file structures; create, rename, and delete folders and files; and, of course, manage your Python environments!

On Windows you can open a command-line prompt by running the program "cmd.exe" (Open the start menu > search for "cmd" or "command prompt" > click the icon). You should see something like this:

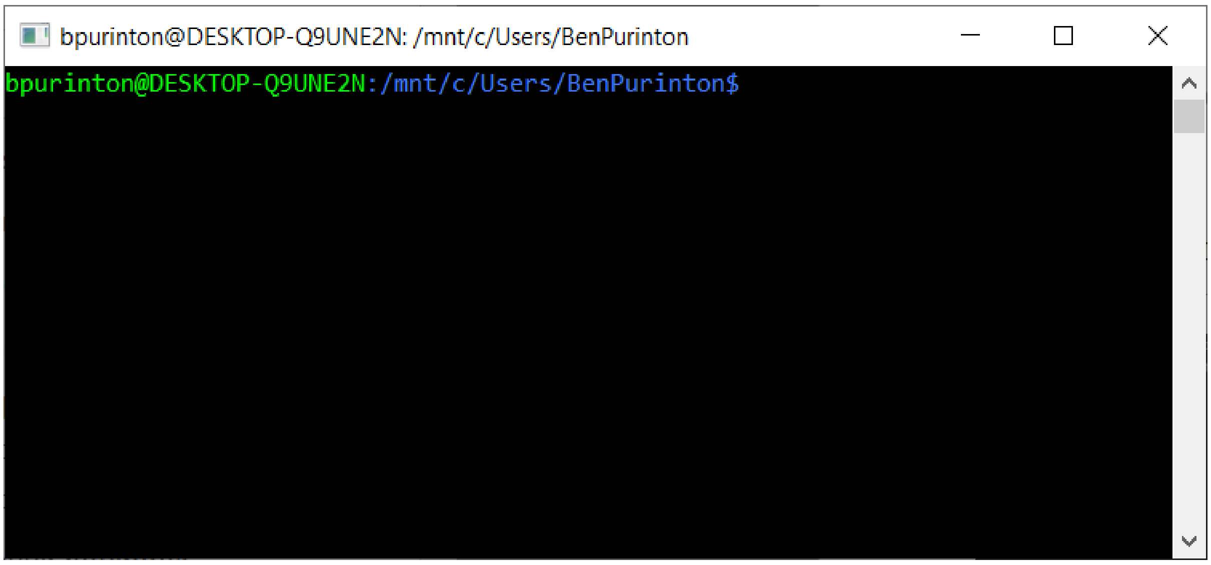


```
Microsoft Windows [Version 10.0.18362.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\BenPurinton>
```

Figure 1: Windows command line.

On Mac or Linux you can open the "terminal" program (Open the system search > search "terminal" > click the icon). You should see something like this:



```
bpurinton@DESKTOP-Q9UNE2N:/mnt/c/Users/BenPurinton$
```

Figure 2: MacOS / Linux command line.

3.4.1 Basic Command-Line Navigation

Rather than clicking around in Finder or Explorer, you can move around your computer's entire file-system structure (so navigating in and out of folders along ***file paths***), create new folders, move files, and list folder contents with a few commands worth memorizing. You can also find cheat sheets with these and other commands online for both Windows and MacOS / Linux. ***Note:*** *Windows uses back slashes whereas Mac and Linux use forward slashes for file paths.* ***Also Note:*** *A > means Windows prompt and $ refers to Mac / Linux.*

To navigate to different folders:

```
$ cd path/to/folder
```

or

```
> cd path\to\folder
```

You can navigate "up" to a parent folder using:

```
cd ..
```

Once in a directory you can list the contents using:

```
$ ls
```

```
> dir
```

Create a folder:

```
mkdir <foldername>
```

Copy a file or folder to a new location:

```
$ cp <file or foldername> new/location/
```

```
> copy <file or foldername> new\location\
```

Move or rename a file or folder:

```
$ mv <folder or filename> <new name or path to new location>
```

```
> move <folder or filename> <new name or path to new location>
```

Delete a file or folder:

```
$ rm -r <file or folder name>
```

```
> del <file or folder name>
```

Last Note: *Your command prompt should support something called "tab completion" meaning that you can just type the first few letters of the filename or path you are interested in and then press the tab button on your keyboard to complete the name. Very useful rather than typing out huge file / path names!*

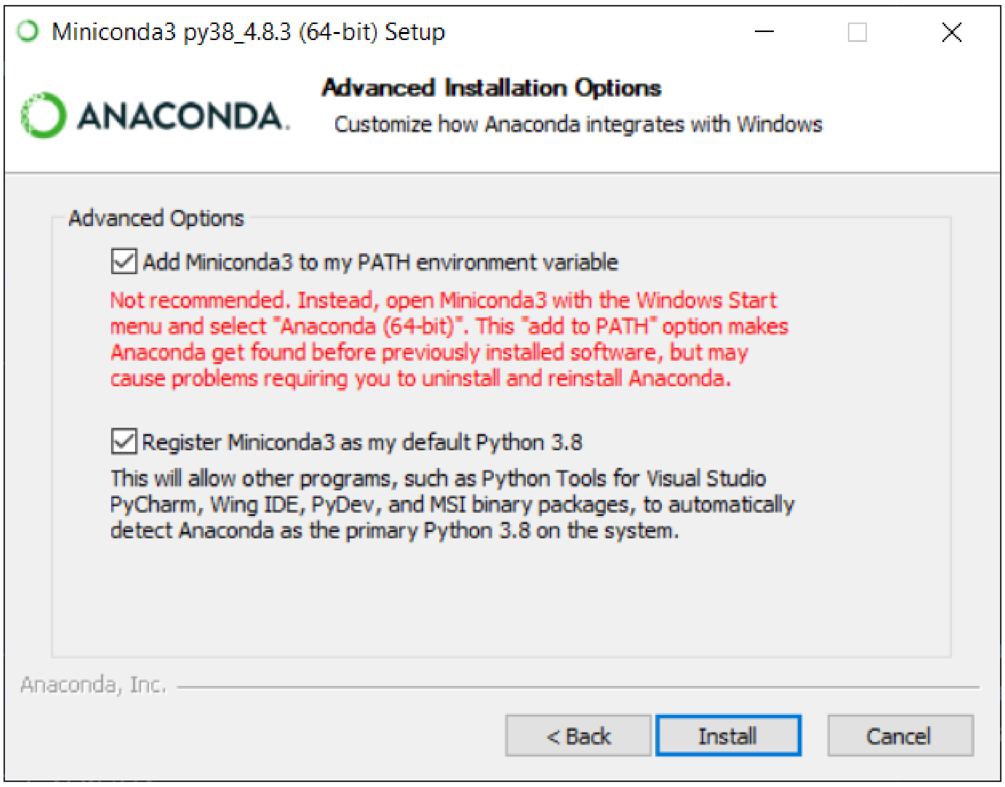
3.5 Install Miniconda

Now that we are familiar with some basic terms, the idea of Python environments, and our command line, it's time to install things. The first step is installing Miniconda from here: https://docs.conda.io/en/latest/miniconda.html. Download the installer corresponding to your operating system, the latest Python version, and your system architecture (most likely that is 64-bit for any computer purchased in the last ~10 years). As of August 19th, 2020 the file would be:

- Windows: "Python 3.8 Miniconda3 Windows 64-bit"
- Mac: "Python 3.8 Miniconda3 MacOSX 64-bit bash"
- Linux: "Python 3.8 Miniconda3 Linux 64-bit"

3.5.1 Windows

Open the downloaded file (e.g., "Miniconda3-latest-Windows-x86_64.exe") and follow the prompts. For the destination folder you can use the default (which should be something like `C:\Users\USERNAME\miniconda3`) or change it to another location. As shown in the below figure, be sure to add Miniconda to the system PATH (which tells the command line where the commands like `conda` are), you can ignore the warning it gives:


Miniconda3 py38_4.8.3 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

☑ Add Miniconda3 to my PATH environment variable

Not recommended. Instead, open Miniconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☑ Register Miniconda3 as my default Python 3.8

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.8 on the system.

Figure 3: Add Miniconda to the system path during Windows install.

3.5.2 Mac / Linux

For Mac and Linux you should have downloaded a `.sh` bash file. This is a command-line script that you can run in the terminal. Open your terminal and navigate to the Downloads folder (or wherever you put the Miniconda `.sh` file) using: `cd path/to/Downloads/`. When you enter the command `ls` in this folder you should see the Miniconda file listed. From here simply run:

```
$ bash <filename>.sh
```

Then follow the command line prompts asking for the install location (e.g., `/home/miniconda3/`) and saying “yes” to adding Miniconda to the system path when prompted.

3.6 conda: Create Python Environment for Course

To make sure `conda` is installed go to your command prompt and enter `conda list`. This should show a list of installed packages. If you get an error like “conda not found” then you may not have added `conda` to the system path. If you don’t get an error then we can create our virtual Python environment for the course *Data Analysis and Statistics*. You can find a good cheat sheet for `conda` commands here.

First we create the new environment using:

```
conda create -n DataAnalysis python=3.*
```

Say “y” when prompted to proceed. This creates a new environment within the Miniconda3 directory called `DataAnalysis`. We are also telling the environment to use the latest Python 3 version using the * wildcard character.

To install packages into this new environment we first need to activate it using:

```
> conda activate DataAnalysis
```

or, on Mac / Linux:

```
$ source activate DataAnalysis
```

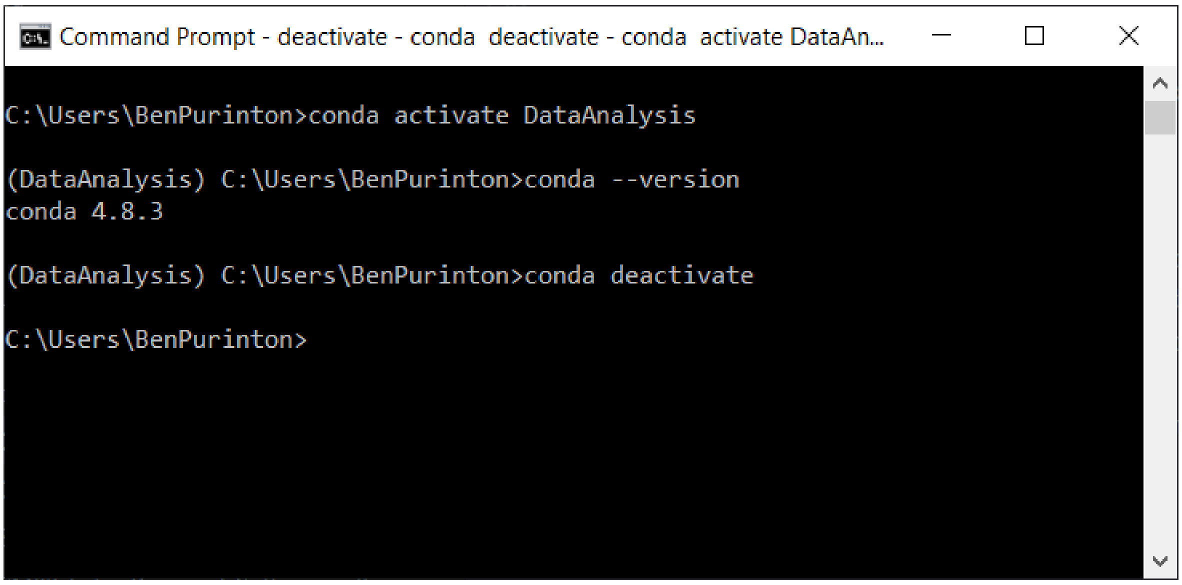
To deactivate the environment, simply run:

```
> conda deactivate
```

or, on Mac / Linux:

```
$ source deactivate
```

When the environment is active you will see `(DataAnalysis)` to the left of your command prompt as in this example from Windows:


Command Prompt - deactivate - conda deactivate - conda activate DataAn...

```
C:\Users\BenPurinton>conda activate DataAnalysis

(DataAnalysis) C:\Users\BenPurinton>conda --version
conda 4.8.3

(DataAnalysis) C:\Users\BenPurinton>conda deactivate

C:\Users\BenPurinton>
```

Figure 4: Activating and deactivating your new conda environment.

3.7 Install Packages in the Environment

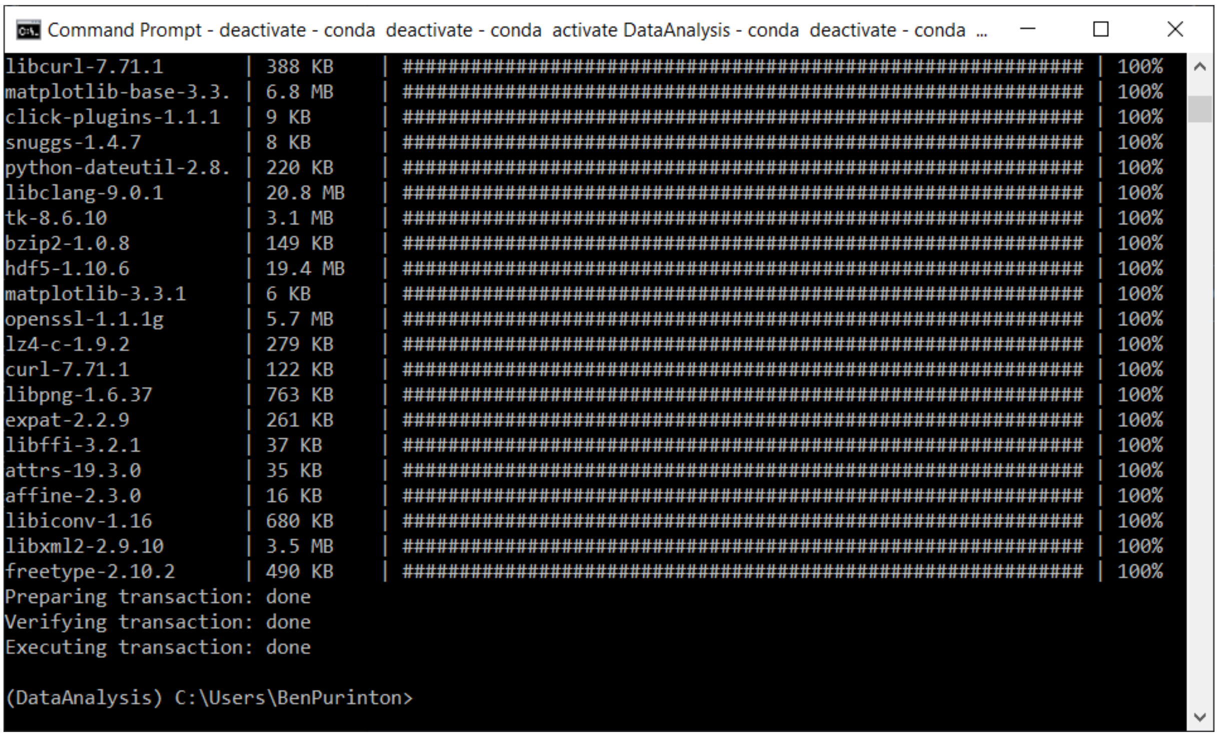
For this course (*Data Analysis and Statistics*) we will rely on some fundamental packages including:

- numpy for math
- matplotlib for plotting
- pandas for tabular data manipulation
- scipy for advanced math
- rasterio for geospatial array manipulation
- jupyter notebook for interactive scripting, we'll run the lecture materials and homeworks through the Jupyter Notebook interface

From the command line, activate the environment so that you see `(DataAnalysis)` next to the command prompt. Now enter the following commands (same for any OS):

```
conda config --prepend channels conda-forge
conda install --y numpy matplotlib pandas scipy rasterio notebook
```

In the end you should see something like this (without any failure messages):



Command Prompt - deactivate - conda deactivate - conda activate DataAnalysis - conda deactivate - conda ...

```
libcurl-7.71.1       | 388 KB    | ############################################################ | 100%
matplotlib-base-3.3. | 6.8 MB    | ############################################################ | 100%
click-plugins-1.1.1  | 9 KB      | ############################################################ | 100%
snuggs-1.4.7         | 8 KB      | ############################################################ | 100%
python-dateutil-2.8. | 220 KB    | ############################################################ | 100%
libclang-9.0.1       | 20.8 MB   | ############################################################ | 100%
tk-8.6.10            | 3.1 MB    | ############################################################ | 100%
bzip2-1.0.8          | 149 KB    | ############################################################ | 100%
hdf5-1.10.6          | 19.4 MB   | ############################################################ | 100%
matplotlib-3.3.1     | 6 KB      | ############################################################ | 100%
openssl-1.1.1g       | 5.7 MB    | ############################################################ | 100%
lz4-c-1.9.2          | 279 KB    | ############################################################ | 100%
curl-7.71.1          | 122 KB    | ############################################################ | 100%
libpng-1.6.37        | 763 KB    | ############################################################ | 100%
expat-2.2.9          | 261 KB    | ############################################################ | 100%
libffi-3.2.1         | 37 KB     | ############################################################ | 100%
attrs-19.3.0         | 35 KB     | ############################################################ | 100%
affine-2.3.0         | 16 KB     | ############################################################ | 100%
libiconv-1.16        | 680 KB    | ############################################################ | 100%
libxml2-2.9.10       | 3.5 MB    | ############################################################ | 100%
freetype-2.10.2      | 490 KB    | ############################################################ | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(DataAnalysis) C:\Users\BenPurinton>
```

Figure 5: Successful installation of packages using conda.

You can update any package (when new versions are released) using:

```
conda update <package name>
```

You can also install packages directly into your base conda environment, such that you do not need to create a new environment and activate / deactivate it. To do this you just run:

```
conda install <package name>
```

without the `conda create env` command. However, this isn't recommended as package conflicts can occasionally occur. You can always delete and remake virtual environments, so duplicate packages on your system are okay (the packages / environments tend to be relatively small in size). Conceptually, this is similar to the schematic in the following figure:

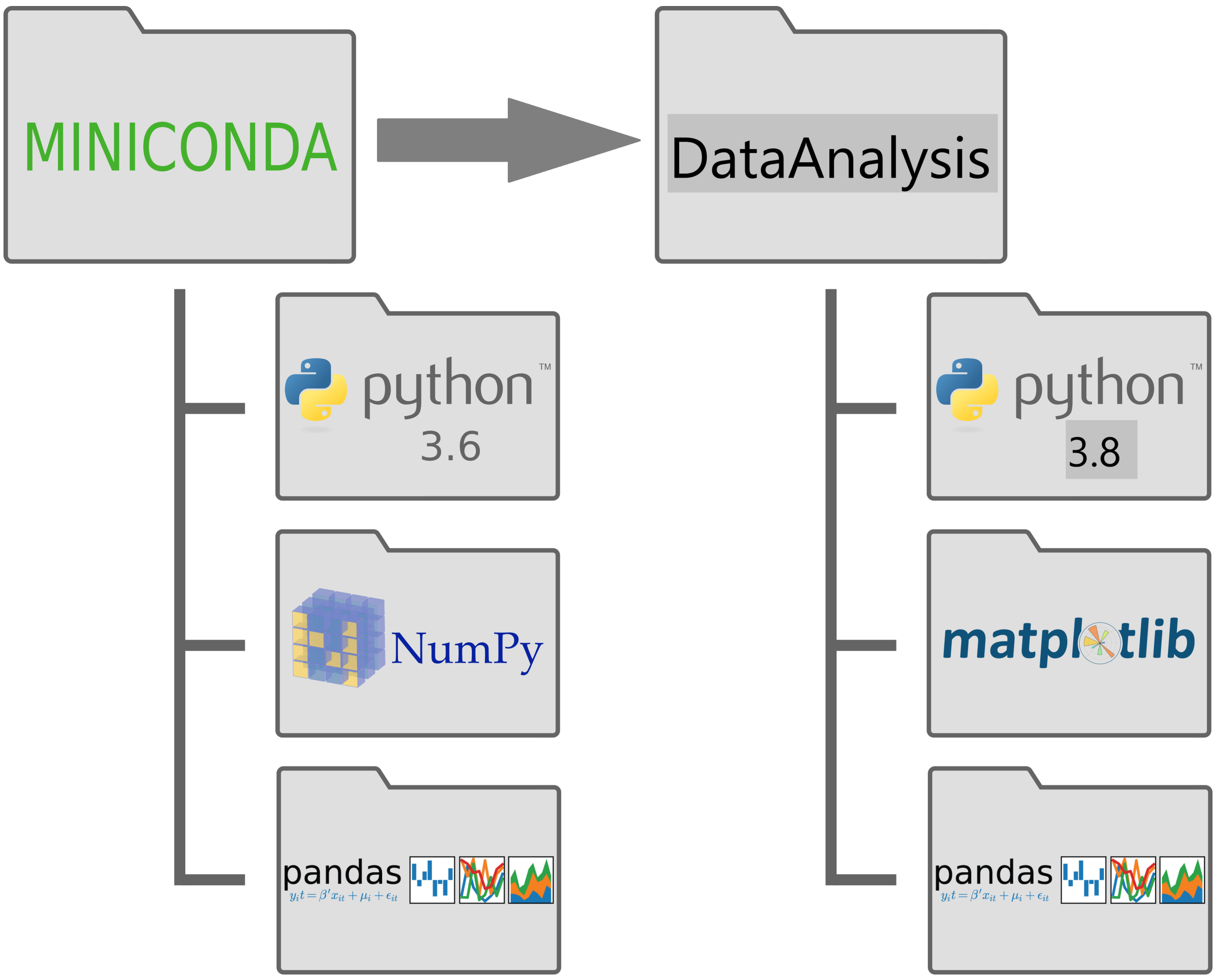


Figure 6: Visualization of core Python and virtual environment installed with Miniconda. On the left: Core Python 3.6 containing the packages NumPy and pandas. On the right: A Python 3.8 virtual environment called "DataAnalysis" installed in the Miniconda directory containing the packages matplotlib and pandas. These Python environments exist completely isolated from one another, hence the double install of pandas. Modified from source: https://python-for-scientists.readthedocs.io/en/latest/_pages/environments.html

3.7.1 Side Note: pip and conda

We installed packages using the `conda` command-line tool. The Miniconda install also included another Python package installer called `pip`. You may occasionally come across Python packages that need to be installed using `pip install <package name>` rather than `conda install <package \ name>`, and you can use `pip` from within the `conda` environment. You can read more about this here.

Note: Before installing any package to an environment you should search the internet for the installation instructions (e.g., using Google to search: "install pandas using conda on Windows 10" or "install openCV using pip on Ubuntu"). `conda` and `pip` don't always work well together so use caution when mixing them in an environment!

3.8 Testing the Install

Now let's just check that the Python environment for *Data Analysis and Statistics* is ready to go before we move on. Please run the following at your command line:

```
python -c "import matplotlib, numpy, pandas, scipy, rasterio"
```

This shouldn't produce any output (or errors).

4 Jupyter Notebook

Jupyter Notebooks offer a way to write rich text (in the Markdown format) and code together in one document, execute blocks of code in an underlying ***kernel***, and output results and figures below each code block. The kernel that Jupyter uses for Python is IPython!

Jupyter works through a web browser communicating locally with your computer ports, so you don't need an internet connection to run it. In this course we'll rely on the classic Jupyter Notebook environment, though you can also use the newer Jupyter Lab, which consists of many modular elements including a file browser, text editor, terminal window, interactive IPython console, and notebook viewer. **Note: Jupyter Notebook code + markdown files always end with the extension `.ipynb`.**

You should have already installed Jupyter Notebook via `conda install notebook`. To start a new session go to your `DataAnalysis` activated command line (navigated to the folder for this class) and enter:

```
jupyter notebook --generate-config
jupyter notebook password
```

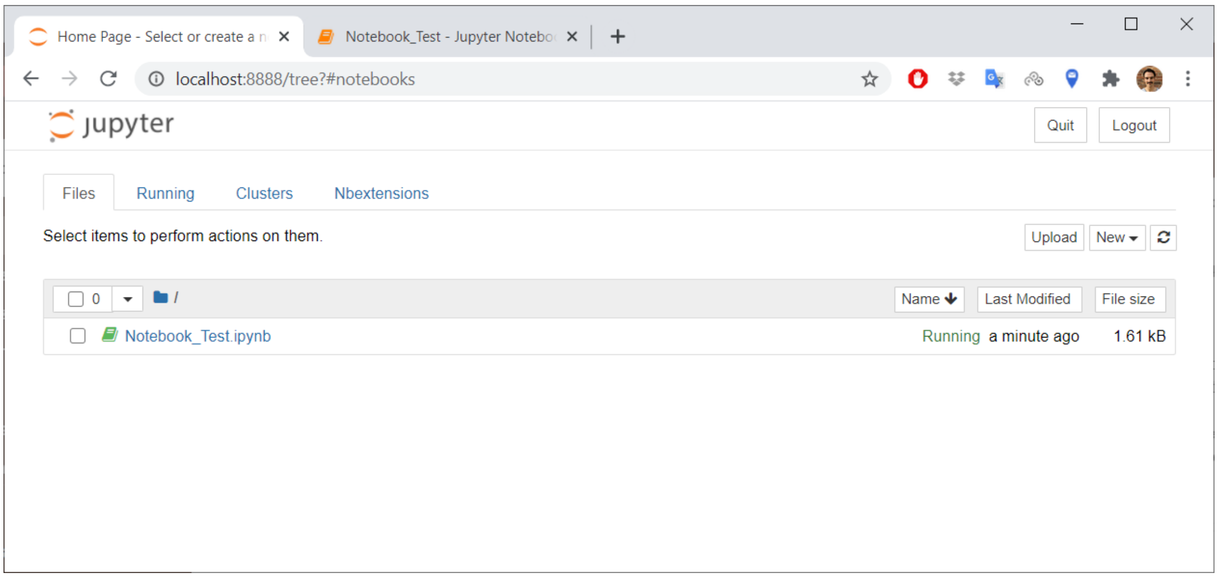
These steps generate a configuration file for Jupyter and allow you to create a password to access the Jupyter Notebook session. Now you can enter:

```
jupyter notebook --no-browser
```

Now open a new internet browser (**either Firefox, Google Chrome, or Safari; NOT e.g., Windows Internet Explorer**) and enter the url "http://localhost:8888/".

Note: Jupyter defaults to port 8888, you can specify a port of your choosing by adding `--port=XXXX` to the previous command (e.g., `jupyter notebook --no-browser --port=1111`), then change 8888 in the url to whatever XXXX port you entered.

You will have to enter the password. Then you will see a screen like this:

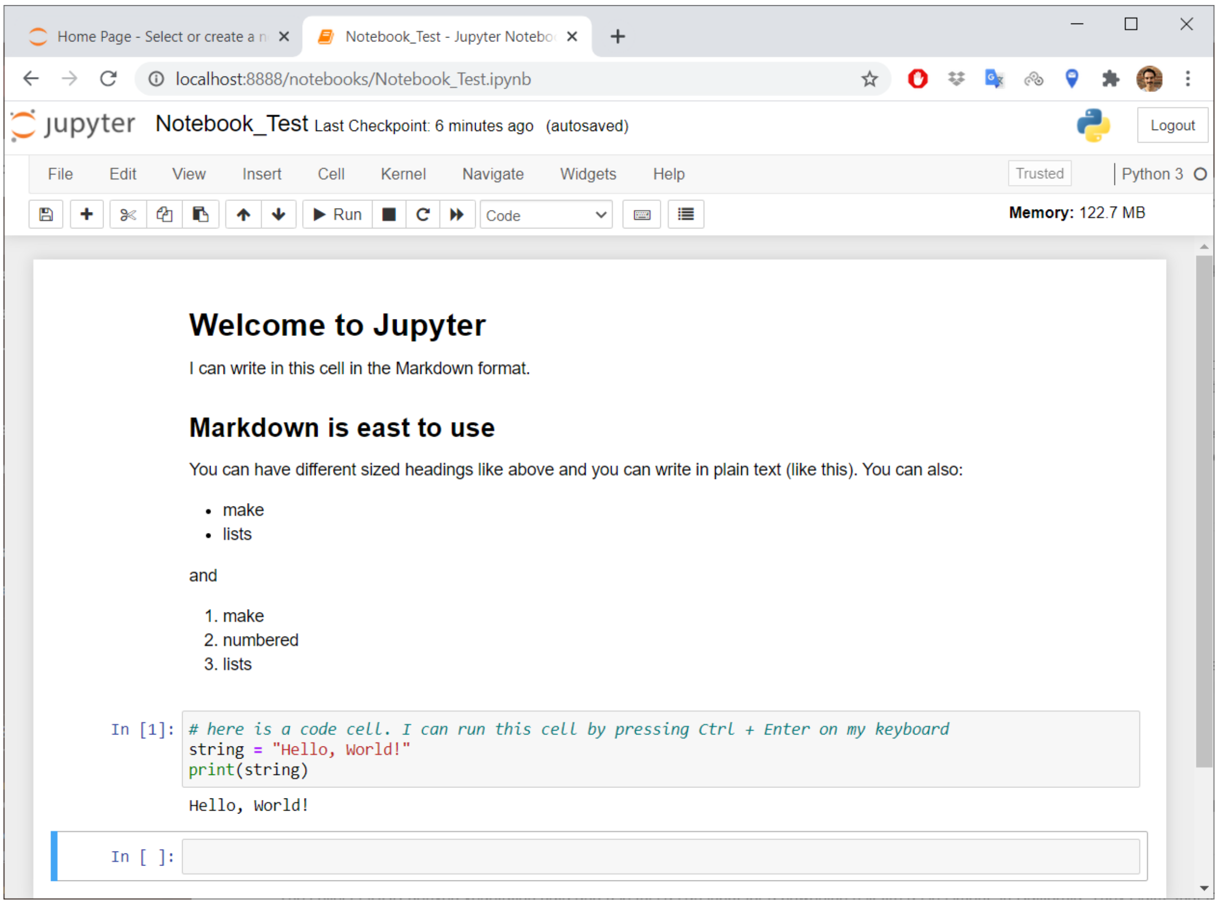

Home Page - Select or create a n
Notebook_Test - Jupyter Notebo
localhost:8888/tree?#notebooks
jupyter
Quit
Logout
Files
Running
Clusters
Nbextensions
Upload
New
Name
Last Modified
File size

Select items to perform actions on them.

0 /	Name	Last Modified	File size
Notebook_Test.ipynb	Running	a minute ago	1.61 kB

Figure 7: Jupyter Notebook showing the "Files" browser (whatever folder you ran the `jupyter notebook --no-browser` command from).

You can open the `.ipynb` files here and run the mixed Markdown and code cells:


Home Page - Select or create a n
Notebook_Test - Jupyter Notebo
localhost:8888/notebooks/Notebook_Test.ipynb
jupyter Notebook_Test Last Checkpoint: 6 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Navigate Widgets Help
Trusted
Python 3
Run
Code
Memory: 122.7 MB

Welcome to Jupyter

I can write in this cell in the Markdown format.

Markdown is east to use

You can have different sized headings like above and you can write in plain text (like this). You can also:

- make
- lists

and

1. make
2. numbered
3. lists

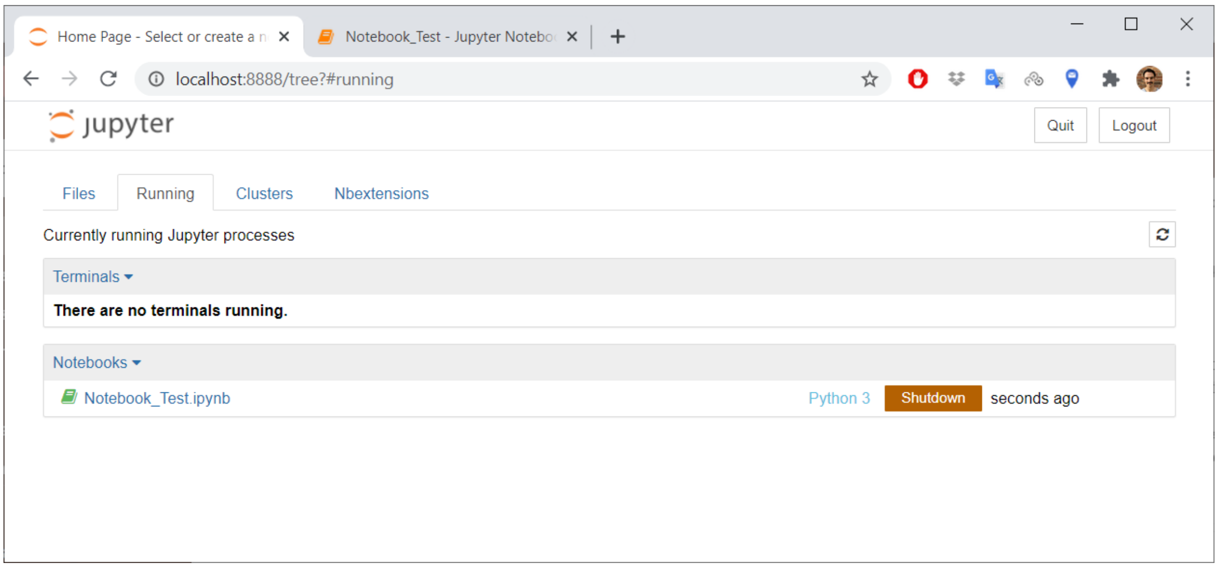
```
In [1]: # here is a code cell. I can run this cell by pressing Ctrl + Enter on my keyboard
        string = "Hello, World!"
        print(string)

        Hello, World!

In [ ]:
```

Figure 8: An open `.ipynb` in Jupyter showing a mix of Markdown and code cells.

Each open `.ipynb` file is attached to a new IPython kernel instance. In the "Running" tab you can see the open notebooks and their kernels (and you can manually shut them down here as well):


Home Page - Select or create a n
Notebook_Test - Jupyter Notebo
localhost:8888/tree?#running
jupyter
Quit
Logout
Files
Running
Clusters
Nbextensions

Currently running Jupyter processes

Terminals

There are no terminals running.

Notebooks

Notebook_Test.ipynb Python 3 Shutdown seconds ago

Figure 9: A list of currently opened `.ipynb` files and associated kernels.

5 Installing a Text Editor

For coding and taking small notes, a text editor is required. Your computer comes with one (e.g., "Notepad" on Windows), but you should install something with more coding functionality.

Install your favorite editor - for example Atom is a great choice and the one we recommend on any platform. You can also use Notepad++ on Windows.