PYTHON TUTORIAL

Created by: Megs Seeley

**Python 101 – Basic Introduction**

This tutorial requires no coding experience or familiarity with Python. This tutorial is a general introduction to using python in Jupyter notebooks.

**Section 0 – SETTING UP YOUR ENVIRONMENT**

If Anaconda is not already installed on your computer, install the latest version here: <https://www.anaconda.com/>

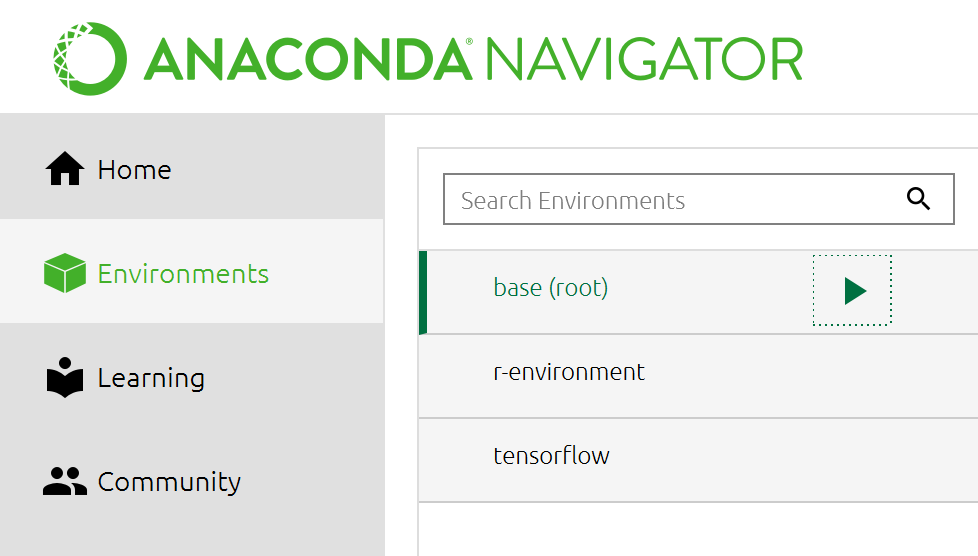
Anaconda is your one-stop-shop for all things python. It installs a variety of python applications, but for this tutorial we will use Jupyter Notebook. I encourage you to explore the different python platforms available.

**Installing Packages using Pip**

Let’s learn how to install a package!

We will install a package on Github that uses an API to import Landsat data: https://github.com/yannforget/landsatxplore/blob/master/README.rst

Open your anaconda navigator and switch to the ‘Environments’ tab. We will add packages in the base (root) environment.



Click on the right-facing arrow next to the base (root) environment then choose ‘Open Terminal’

We can use pip to install new packages in this base environment.

pip install rasterio

**Installing Packages using Conda**

Accessing packages from Jupyter can be a bit messy since package installation is dependent on a number of factors. I won’t go over the whys because you can read them here: <http://jakevdp.github.io/blog/2017/12/05/installing-python-packages-from-jupyter/index.html>

Sometimes the packages you install using pip will return an error when importing in Jupyter Notebook. There are a few ways to get around this.

*Installing Package in Anaconda Navigator using Conda*

You can reinstall the package using conda. In the same Anaconda terminal you accessed above, run:

conda install rasterio

*Install Package in Working Directory using Conda*

Through your Anaconda Prompt, navigate to your working directory

cd Documents/DEVELOP/geoTutorial

Then install your package here

conda install rasterio

*Install Package in Jupyter Notebook*

In your notebook cell, run:

import sys

!conda install --yes --prefix {sys.prefix} rasterio

import rasterio

I would recommend trying the above fixes in the order listed. If these options do not work, check your system path as outlined in the link above.

**Open Jupyter Notebook**

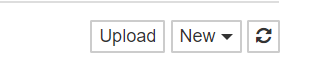
We will open jupyter notebook via the command line interface (CLI).

1. Type "CMD" into your computer's search bar.
2. Open the Anaconda Prompt application

On the prompt, you will see a path to your home directory, which will start with "C:"  
You can navigate your path like you would your file explorer folders.

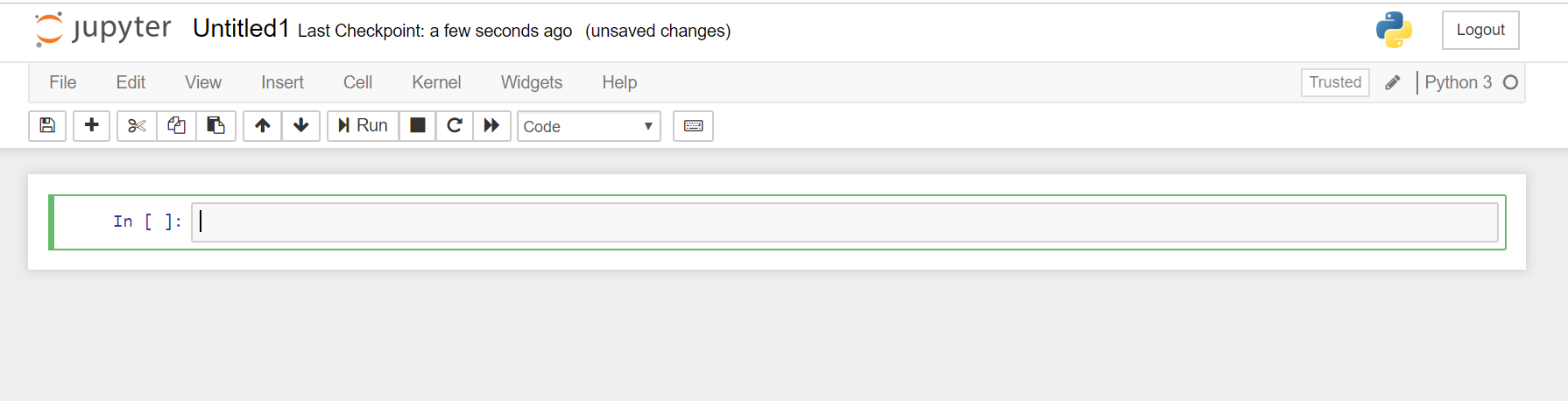
1. Navigate to your Documents folder with "cd"
2. Open jupyter notebook

cd Documents/DEVELOP/geoTutorial

<< Click ‘New’ in the upper righthand corner. Then choose ‘Python 3’

Awesome! Now we have a new jupyter notebook script!

**Section 1 – THE NOTEBOOK**



**2**

**1**

1. Editor Panel

Writing and editing. Add snippets of code to each cell. You can add cells and run them independently

2. Quick Navigation

From left to right:

a. Save

b. Insert new cell below

c. Cut cell

d. Copy selected cell

e. Paste cell below

f-g. Navigate up/down cell

h. Run cell

i. Stop running script

j-k. Restart and rerun notebook

l. Toggle cell from code to markdown language

3. Main Navigation

Important options for editing script

**Section 2 – NUMPY**

*Note: If you are waiting for packages to install, you can follow along with the tutorial by accessing the notebook at:* https://mybinder.org/v2/gh/MegsSeeley/pythonTutorialEnvir/master

*Note: Use # to comment*

# This is a comment

There are a handful of packages you will likely encounter if you continue to work with spatial data in python: NumPy, Pandas, Rasterio, and Matplotlib. We will briefly go over the first three in this tutorial. Tune in to the Data Visualization tutorial to learn about Matplotlib.

First, let’s import numpy!

import numpy as np

*Vocab!*

**tuple:** Items are iterable and the same  
**ndarray:** NumPy array class (also known as numpy.array)

*Best use:*

Multidimensional homogeneous arrays

**Arrays**

Manually Create an Array

arr = np.array([ [1,2,3], [4,5,6] ])

arr2 = np.array([ [7,8,9], [10,11,12] ])

Create an array based on a range

arrRange = np.arange(start, end, step)

arrRange = np.arange (0, 2, 0.3)

**Bonus Arrays:**

Array of zeros

np.zeros (2, 2, 3)Array of ones

Array of ones

np.ones((2,3), dtype = np.int16) # dtype specifies that values should be integers

**Operating on Arrays**

We can perform basic operations on arrays

subtractArr = arr – arr2

multiplyArr = arrange\*10

conditionalArr = arr<3

For more information on NumPy, please visit: <https://docs.scipy.org/doc/numpy/user/quickstart.html>

**Section 3 – PANDAS**

NumPy’s more elegant sibling. Pandas is your go-to for data manipulation. Here we will focus on a few simple methods of data manipulation, but keep this package in mind for time series analyses, working with csv data, data cleaning, and more.

First, let’s import pandas!

import pandas as pd

**Import data**

testDat = pd.read\_csv(‘data/testTimeSeries.csv’)

print(testDat)

**Calculate basic statistics based on data classification**

classes = testDat.groupby(‘class’)

classes.mean()

**Calculate basic statistics based on a column**

testDat[[‘A’]].min()

Hopefully this gives you a glimpse of the capabilities of pandas. To learn more, visit the pandas homepage: <http://pandas.pydata.org/>

**Section 4 – RASTERIO**

Rasterio allows us to manipulate geoTIFFs.

import rasterio as rio

from rasterio.plot import show

Next, we will read in our geoTiffs. The geoTiffs we will use have already been extracted from both the .gz and .tz to .tiff. While we will not go over the process here, there are python packages available that will allow you to work with both these zipped data types.

When opening rasters with rasterio, we have the option to open them as numpy arrays. This allows us to perform raster math on the images.

with rio.open('data/LC08\_L1TP\_038030\_20180911\_20180911\_01\_RT\_B4.tif') as red\_temp:

red = red\_temp.read(1, masked = True)

with rio.open('data/LC08\_L1TP\_038030\_20180911\_20180911\_01\_RT\_B5.tif') as nir\_temp:

nir = nir\_temp.read(1, masked = True)

Let’s break that down.

The first line -- with rio.open(‘file’) as tempName – tells the interpreter to open our file using rasterio with our temporary name.

The second line – imageName = tempName.read(band#, masked = True) – assigns our numpy array to a variable name. We can choose the band. In both these cases, the rasters have only one band. masked = True ensures that no data values appear as NA rather than as a number.

Next, we will calculate NDVI. To prevent errors in our calculation, we first need to make sure our data are of type float. This allows our output to have decimals.

nir = nir.astype(float)

red = red.astype(float)

Now we can calculate the NDVI

ndvi = (nir-red)/(nir+red)

Lastly, let’s plot the our results!

rio.plot.show(ndvi,

title = "NDVI")

**Bonus Section – IMPORTING IMAGES USING API**

**Import Landsat using API**

Import landsatxplore api

import landsatxplore.api as lx

from landsatxplore.earthexplorer import EarthExplorer

Sign in to Earth Engine Explorer

api = lx.API(“username”, “password”)

Search for scenes and create a list of IDs

scenes = api.search(

dataset = 'LANDSAT\_8\_C1',

latitude = 40.047875,

longitude = -6.884606,

start\_date = '2018-01-01',

end\_date = '2018-11-01',

max\_cloud\_cover = 20

)

print('{} scenes founds.'.format(len(scenes)))

# Create list of scene IDS

l8\_idList = []

for scene in l8\_scenes:

l8\_idList.append(scene['displayId'])

print(l8\_idList)

# Don’t forget to log out!

api.logout()

Download your landsat scenes

ee = EarthExplorer(“username”, “password”)

for scene in l8\_idList:

ee.download(scene\_id=scene, output\_dir='./data')

# Don’t forget to log out!

ee.logout()