Using Anaconda modules from the ESRI python environment

What is the Anaconda distribution?

Anaconda is an open-source Python distribution that makes is possible to easily install and manage many pre-packaged third party Python modules.

It has some big advantages over using the Python Esri provides with ArcGIS:

- Very useful tools not in Esri's standard distribution are available (IPython, Jupyter notebook, Spyder)
- You can install and update the Python stack as a regular (non-pr) user
- You can install and remove, and update third party packages easily, with Anaconda checking for compatibility for you
- Manage multiple environments (for example, one compatible with ArcMap, another compatible with ArcGIS Pro)

Workflow

The general workflow to make this happen is to:

- Install Anaconda without fouling the Windows environment (paths, registry) to break Esri's python stack
- Configure Anaconda with the particular add-ons you want, and
- Configure ArcGIS's Python so that it is aware of the Anaconda environment (so you can import modules not in Esri's stack)

1) Download and Install Anaconda

The main Anaconda distribution is pretty large (> 3 GB). *MiniConda* is a smaller version that just includes the Python standard library to start.

You can download and install either one, depending on whether you want to spend the time and disk space. Miniconda most efficient, as you can always download down anything in the conda distibution you want later.

- MiniConda for Python 2 32-bit (to work with ArcMap): https://repo.continuum.io/miniconda/Miniconda2-latest-Windows-x86.exe
- MiniConda for Python 3 64-bit (to work with ArcGIS Pro): https://repo.continuum.io/miniconda/Miniconda3-latest-Windows-x86_64.exe
- Other options: https://www.continuum.io/downloads

Run the .exe installers

Select install for a single user (Not "All Users")

1. Install to a folder where there is going to be plenty of space (recommend the D drive, not the C drive)
2. **IMPORTANT:** To avoid breaking ArcGIS, uncheck the checkboxes (a) make Anaconda the default Python and (b) add Anaconda's Python to the PATH.
3. Here are **screenshots** to help guide you through the install process.
4. Windows 10: after installing, rename the shortcut Anaconda Prompt to "Anaconda Prompt 32-bit" or "64-bit"

Anaconda first run

Open the Anaconda prompt window (Search in Windows and start the shortcut "Anaconda Prompt") and enter "python" in the command window:

First Anaconda Session

```
(D:\Users\jwpowell\Miniconda2) C:\Users\jwpowell>python
Python 2.7.13 |Continuum Analytics, Inc.| (default, May 11 2017, 14:07:41)
[MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>>
```

At this point, you have a nice Anaconda Python setup (~ 225 MB) that's spiffy and new and totally useless with ArcMap.

2) Set up an SSL certificate for Conda

The Department of Interior is now requiring SSL encryption. This means you cannot download Anaconda packages without an SSL certificate in place.

1. Download the DOI certificate: http://sslhelp.doi.net/docs/DOIRootCA2.cer to your Downloads folder (details on SSL setup)
2. Open "Anaconda Prompt" window
3. Set up the certificate to use it with Conda

Windows Command Prompt

```
mkdir %USERPROFILE%\.certificates
copy %USERPROFILE%\Downloads\/DOIRootCA2.cer %USERPROFILE%\.certificates
conda config --set ssl_verify %USERPROFILE%\.certificates\DOIRootCA2.cer
```

Set up SSL certificate for PIP (optional)

If you plan to use the pip utility to download and install packages not included in the Conda distribution, set up a certificate for pip:

Windows Command Prompt

```
mkdir %APPDATA%\pip
(echo [global] & echo cert=%USERPROFILE%\.certificates\DOIRootCA2.cer) >
%APPDATA%\pip\pip.ini
```

3) Configure Anaconda To Work with ArcGIS

The following workflow will demonstrate how set up a custom Python environment within Anaconda that has the same modules installed as ArcGIS Python, and then add a compatible version of Spyder and Jupyter.

The following example is for ArcGIS 10.4.1, assuming you have successfully installed MiniConda 2 32-bit (as above).

A. Find the versions of numpy and matplotlib for your version of ArcGIS Desktop

The critical modules for ArcGIS compatibility can be determined from your Desktop or Pro Python command line.

(Or you can just look at the list below under B.)

Python script for ArcGIS Python window

```
# pyversions.py - report ArcGIS Python modue versions
# Example output:
# C:\ArcGIS\Pro\bin\ArcGISPro.exe
# Python 3.5.2 |Continuum Analytics, Inc.| (default, Jul  5 2016, 11:41:13)
[MSC v.1900 64 bit (AMD64)]
# matplotlib 1.5.3
# numpy 1.11.2
# scipy 0.18.1
import sys
import os
ff = "{} {}"
try:
    print(sys.executable)
    print(ff.format("Python", sys.version))
    import matplotlib
    print(ff.format("matplotlib", matplotlib.__version__))
    import numpy
    print(ff.format("numpy", numpy.__version__))
    import scipy
    print(ff.format("scipy", scipy.__version__))
except:
    pass
```

B. Create an Anaconda environment for use with ArcGIS

We'll also include a few other modules that we know are shipped in the ArcGIS Python stack.

1. Open the Anaconda prompt (32 or 64, depending on the ArcGIS environment you are integrating with)
2. Create a compatible environment
    a. 32-bit (ArcMap, ArcCatalog)
    ArcGIS 10.2.2: *conda create -n arc1022 python=2.7.5 numpy=1.7.1 matplotlib=1.3.0 pyparsing xlrd xlwt console_shortcut*
    ArcGIS 10.3.1: *conda create -n arc1031 python=2.7.8 numpy=1.7.1 matplotlib=1.3.0 pyparsing xlrd xlwt console_shortcut*
    ArcGIS 10.4.1: *conda create -n arc1041 python=2.7.10 numpy=1.9.2 matplotlib=1.4.3 scipy=0.16.0 pandas pyparsing xlrd xlwt c onsole_shortcut *NOTE* Esri shipped 0.15.0, but I needed to use 0.16.0 to have conda work*
    ArcGIS 10.5: *conda create -n arc105 python=2.7.12 numpy=1.9.2 matplotlib=1.4.3 scipy=0.17.0 pandas pyparsing xlrd* xlwt *cons ole_shortcut*
    ArcGIS 10.5.1: *conda create -n arc1051 python=2.7.13 numpy=1.9.3 matplotlib=1.5.2 scipy=0.17.0 pandas pyparsing xlrd* xlwt *c onsole_shortcut*
    b. 64-bit (Background Geoprocessing (x64), ArcGIS Pro
    ArcGIS 10.x x64 background processing: same as above, from Anaconda 64-bit prompt, for example:
    conda create -n arc105x64 python=2.7.12 numpy=1.9.2 matplotlib=1.4.3 scipy=0.17.0 pandas pyparsing xlrd xlwt console_sh ortcut
    ArcGIS Pro 1.2: *conda create -n arcpro12 python=3.4.3 numpy=1.9.3 matplotlib=1.4.3 scipy=0.16.0 pandas pyparsing xlrd xlwt c onsole_shortcut*
    ArcGIS Pro 1.3: *conda create -n arcpro13 python=3.4.4 numpy=1.10 matplotlib=1.4.3 scipy=0.16.1 pandas pyparsing xlrd xlwt co nsole_shortcut*
    ArcGIS Pro 2.0: *conda create -n arcpro20 python=3.5.2 numpy=1.11.2 matplotlib=1.5.3 scipy=0.18.1 pandas pyparsing xlrd xlwt console_shortcut*
3. The conda tool will:

    - Determine that the specified package versions are compatible with each other.
    - Find any packages on which these depend and determine most recent versions that are compatible with what you've asked for, if not completely specified.
    - Show you what it plans to do and prompts you to continue.
    - Download (only) the packages you need to "overlay" onto the base environment to get the environment you have specified.
    - Set up a environment subdirectory, (D:\Users\jwpowell\Miniconda2\envs\arc1041 in our example) installing the downloaded packages into it. This adds up to about 1.5 GB in our ArcGIS 10.4.1 example.

- The *console_shortcut* conda package will adds a shortcut to your Windows start menu that directly starts up your environment. (Note, if you remove the environment, you will have to delete it yourself.)
- *Windows 10 note:*

  Windows 10 won't show a start menu item if the name of an item/shortcut within the new group matches a shortcut name in another group and has the same target (e.g. both 'Anaconda2 (32-bit)' and 'Anaconda2 (64-bit)' have a shortcut named 'Anaconda Prompt' with a target of cmd.exe which confuses Windows 10). So, when Miniconda2 32-bit (and/or 64-bit) is installed edit the 'Properties' of 'Anaconda Prompt' -> General tab -> rename to 'Anaconda Prompt 32-bit'. Also, rename each shortcut that is created for environments for 32-bit or 64-bit similarly for consistency. For example, if you created a new virtual environment in Anaconda 32-bit named arc1041 then you would rename the shortcut from 'Anaconda Prompt (arc1041)' to 'Anaconda Prompt 32-bit (arc1041).

C. Test the virtual environment

Open an Anaconda command window and load the virtual environment.

Anaconda command prompt

```
C:\Users\jwpowell> conda info --envs
# conda environments:
#
arc1041                  D:\Users\jwpowell\Miniconda2\envs\arc1041
root                  *  D:\Users\jwpowell\Miniconda2
D:\Users\jwpowell>activate arc1041
Activating environment "arc1041"...
[arc1022] D:\Users\jwpowell> conda list
# packages in environment at D:\Users\jwpowell\Miniconda2\envs\arc1041:
#
dateutil                  2.4.1                     py27_0
matplotlib                1.3.0                 np17py27_0
numpy                     1.7.1                     py27_3
...
```

D. Add more packages

You can add more packages using **conda install,** but make sure you specify version numbers for these that won't change the Python modules required to stay compatible with ArcGIS's Python stack.

Let's add the Jupyter notebook and the Spyder IDE which are both popular additions to your toolbox.

Specifying the list of module versions (ARCLIST) here ensures the environment will still work with ArcGIS 10.4.1.

Anaconda command prompt

```
set ARCLIST=python=2.7.10 numpy=1.9.2 matplotlib=1.4.3 scipy=0.16.0 pandas
pyparsing xlrd xlwt
conda install -n arc1041 %ARCLIST% jupyter spyder
...
The following NEW packages will be INSTALLED:
...
```

Pinning dependencies

Conda does provide a method for pinning dependencies to an environment so you don't have to specify them each time (as: ARCLIST above).

You need to list these packages in a filed called *pinned* in the conda-meta folder.

Anaconda command prompt

```
C:\Users\jwpowell> activate arc1041
(arc1041) C:\Users\jwpowell>(echo python ==2.7.10 & echo numpy ==1.9.2 &
echo matplotlib ==1.4.3 & echo scipy ==0.16.0) >
%CONDA_PREFIX%\conda-meta\pinned
(arc1041) C:\Users\jwpowell>type %CONDA_PREFIX%\conda-meta\pinned
python ==2.7.10
numpy ==1.9.2
matplotlib ==1.4.3
scipy ==0.16.0
```

Now that this has been set up, *conda install -n arc1041 scipy-0.18.1* will generate an error, and *conda install six* will update *six* to the highest version compatible with the pinned modules.

Finding more packages

You can search for more packages available in conda with **conda search.** There's a nice list on the Anaconda website that describes them all.

You're not limited to adding only packages to which conda has access to your new environment. Here's an example on how to install a commonly desired GIS-related package (**shapely**) into a conda virtual environment using the python **pip** utility:

http://deparkes.co.uk/2015/01/29/install-shapely-on-anaconda/

4) Configure ArcGIS python to see Anaconda environment and vice versa

This can most easily be done (personal opinion) with a Python usercustomize.py startup script

- Install Anaconda, setup environment to match your ArcGIS version
- Download (and rename from .py.txt to .py) this script: (usercustomize.py.txt) -- and edit it to match your setup:

Edit block in usercustomize.txt

```
############################################
# Edit here match your setup
# These paths must match your Anaconda setup exactly.

# Anaconda home folders
conda_arcmap_home = r"D:\Users\jwpowell\Miniconda2"
conda_arcmap64_home =  r"D:\Users\jwpowell\Miniconda3x64"
conda_arcpro_home = r"D:\Users\jwpowell\Miniconda3x64"

# anaconda environments set up to match Desktop and Pro
conda_arcmap_env = "arc1041"
conda_arcmap64_env = "arc1041x64"
conda_arcpro_env = "arcpro13"

# ArcGIS Pro install folder
default_pro_path = r"C:\ArcGIS\Pro"

# change to false after testing done
debug = True

# do not edit below this line
############################################
```

- IMPORTANT - Test and debug this script by running it from python.exe for each of your environments. (See examples in the script header.)

Create a user-specific site packages folder and place the script in the folder with the name *usercustomize.py*.
The user site-packages folder path can be found with: `python -m site --user-site`
Usually: `C:\Users\`*`username`*`\AppData\Roaming\Python\Python27\site-packages (ArcGIS Desktop)`
`C:\Users\`*`username`*`\AppData\Roaming\Python\Python34\site-packages (ArcGIS Pro, Python35 for Pro 2.0)`

Command prompt

```
:: Python 2.7
mkdir %APPDATA%\Python\Python27\site-packages
copy usercustomize.py %APPDATA%\Python\Python27\site-packages
:: Python 3.4
mkdir %APPDATA%\Python\Python34\site-packages
copy usercustomize.py %APPDATA%\Python\Python34\site-packages
```

Testing and Troubleshooting

ArcGIS

1. Start Desktop or Pro, open the Python window
2. `print ("\n".join(sys.path))  -- you should see the Anaconda site-packages near the end of the list`

Anaconda

1. Start Anaconda 32-bit or 64-bit command prompt
2. `activate arc1041  (or whichever environment in the list)`
3. `python`
4. `import arcpy`
5. `print ("\n".join(sys.path)) -- you should see the ArcGIS site-packages near the end of the list`

References

https://www.continuum.io/

How to do a separate Python installation with ArcGIS? (GIS Stack Exchange)

Using ArcPy with Anaconda (PyMorton)

Acknowledgements

Many thanks to Parker Norton for his reviews and excellent additions to this document!