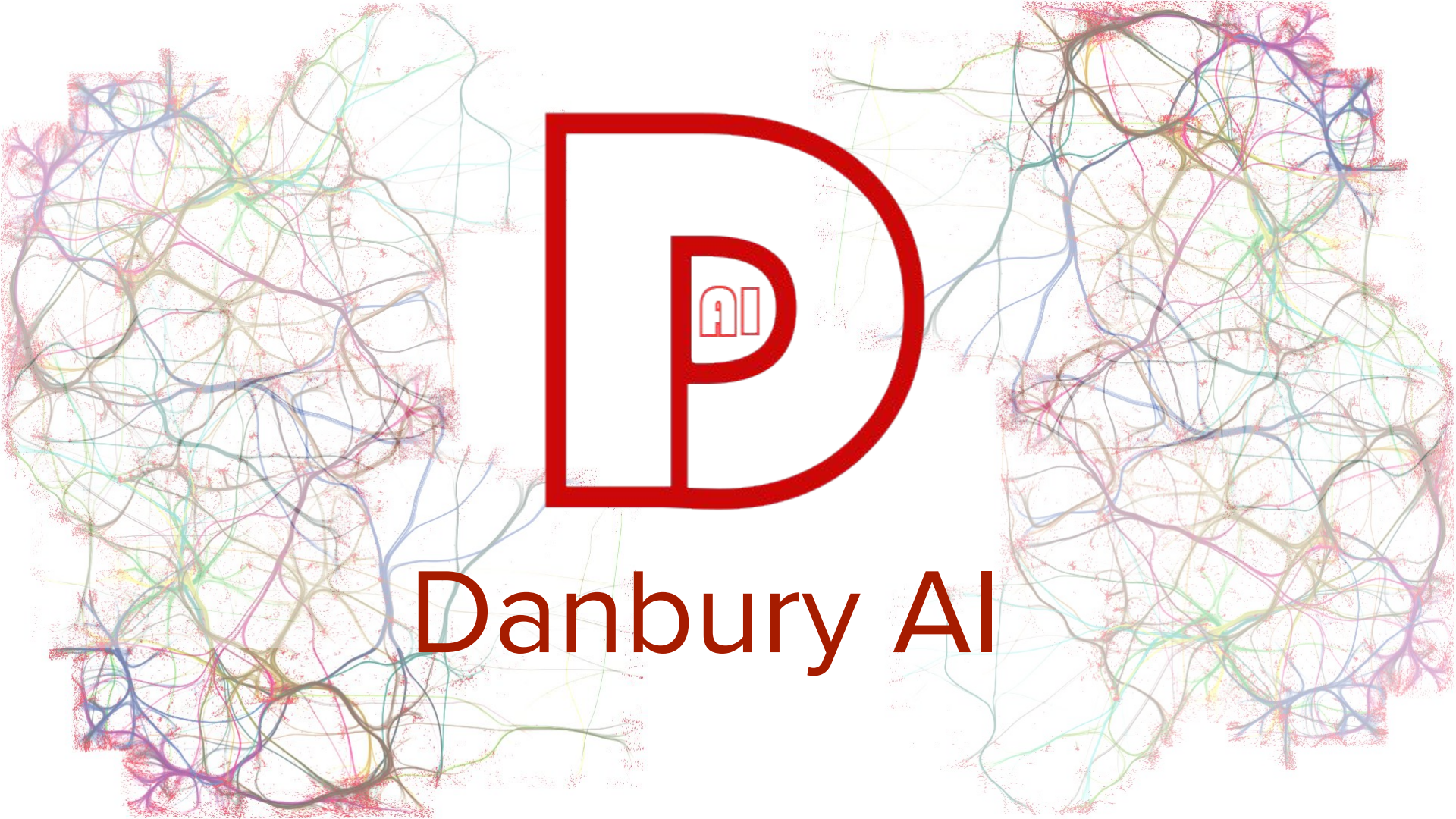


Danbury AI

WORKSHOP

Scientific Computing in Python

Python is one of the most popular open source languages in history. There are more than 100,000 open source packages published on the official package index PiPy alone and many more projects in general. Under the banner of SciPy, there is a mature ecosystem of python packages for doing far reaching scientific analysis in python. In this workshop we cover a good number of the core packages and show you the door for further study. This workshop is accompanied by several interactive Jupyter Notebooks which illustrate different aspects of the SciPy ecosystem.

December 2017

By: Andrew Ribeiro of KEXP.IO

Introductions

Welcome to Danbury AI! Let's introduce ourselves. Here are some things we'd like to know:

- Name?
- Areas of interest?
- What do you know about AI?
- Current projects?
- What are your hobbies? Give us at least one!
- What would you like to get out of this meetup?

Outline

- Introduction
  - Mathematical Software
  - Why Python?
  - Why ~Python?
  - SciPy
  - Additional Libraries that Play Nice with SciPy
  - Alternative Tools and Environments
- Setup
  - Environment Setup
- Workshop
  - Notebooks
  - Poke Pandas

KEXP

Introduction

KEXP

Mathematical Software

- There are many types of mathematical software used in research and development. Among which are:
  - Visualization (Graphing, Interactive Plots)
  - Symbolic Manipulation (CAS, Proof Assistants, Logical Languages)
  - Numerical Computation (Matrix Computations, High Precision Calculations, Statistics)
  - Simulation (Monte Carlo, Physics, Reinforcement Learning Environments)
- In modern mathematical and scientific studies, computations become extraordinarily large and complex. Tools have thus arisen to employ computers for managing this complexity.
- Mathematical software allows us to explore parameters, calculate the results of our assumptions, implement algorithms, and explore data.

KEXP

Why Python?

- Like learning a natural language, a huge motivator is the community of people or work you'd like to access with the language. Python is arguably the largest open source language to date with 123,482 packages on PiPy and hundreds of thousands of open source projects. There are many mature projects for doing mathematics and scientific analysis with python as well as many other diverse applications (Ex. Blender.).
- Python has a blend of modern language features that make it highly expressive and easy to learn.
- Python interops with C++ libraries easily, so one can implement high performance pieces of applications in the compiled language, then use them from python.

KEXP

Why ~Python?

- It is a dynamically typed language, so we do not get the security of a strongly typed language. Thus, when compile time checks are vital, Python is not recommended. Languages that provide these securities are sometimes called system languages. Among which are C/C++ and Rust.
- Python and python packages are primarily open source projects. This often comes with an abundance of headaches.
  - Will I be able to get support if something breaks?
  - Will the packages my project relies on be maintained?
  - Which packages do I use? There seems to be 10 for every problem.
  - Are the packages I need truly platform independent? Or do they have some dependencies that break them on different platforms.
- Sticking with an established packages backed by reputable companies with a commitment to their software generally abates many open source issues.If you use a package built by Danny the grad student, don’t expect reliability.

KEXP

SciPy

- SciPy is a far reaching ecosystem of open source packages for doing scientific computing in Python.
- The Language: Python. Dynamically typed and modern.
- Core Libraries
  - **NumPy**-Numerical computing package with high performance C++ backend.
  - **SciPy Library**-A suite of tools for common and domain specific numerical algorithms.
  - **Matplotlib**-Robust 2D Plotting and basic 3D plotting. As well as interactive and animation tools as well.
- Data and Computation
  - **Pandas**-Dataset utilities and common data structures like data frames.
  - **SymPy**-Full featured computer algebra system for symbolic mathematics .
  - **Scikit-Image**-Tools for image processing.
  - **Scikit-Learn**- A suite of tools for machine learning.
  - **H5py and PyTables** -Access to large HDF5 formatted datasets.

SciPy

- Productivity and High-Performance Computing
  - **IPython** - Interactive computing with python.
  - **Jupyter Notebooks** - Uses ipython in the browser and provides many kernels for different languages. A general web-based platform for interactive computing.
  - **Cython** - Easily build C/C++ implementations that can be used from python programs for when high performance is needed.
- Documentation and Testing
  - **Pytest**-A unit testing suite for quality assurance.
  - **Numpydoc** - An extension of sphinx for documenting numerical libraries.

Additional Libraries that Play Nice with SciPy

- Package Managers and Environments
  - **Anaconda Navigator** - Helps you configure different native environments and includes the package manager conda.
  - **Docker**- Used to produce containerized environments.
- Data Flow Graphs
  - **TensorFlow** - Multi-platform and mobile support. Heavy focus on cluster computing. Generally harder to use due to lots of idiosyncrasies and static computational graphs. Automatically generates optimal graphs and gradients. Includes TensorBoard for visualizing results and ongoing computations.
  - **PyTorch-** Centered on dynamic computational graphs and is generally easier to use.
- Visualization
  - **Plotly** - Great utilities for 3d plotting. Feels like a modern version of Matplotlib.
  - **Bqplot** - A newer widget for making interactive visualizations in Jupyter notebooks.
- Testing and Documentation
  - **PyDoc** - The common tool for producing HTML documentation from docstrings in your library.

There are many Pyhon more libraries that work well with the SciPy workflow!

KEXP

Alternative Scientific Computing Tools

- Open Source
  - **R** - Prefered by many statisticians. CRAN, the package manager for R, is very well documented and is more rigorous than Python's pypi. R studio is phenomenal.
  - **SageMath** - Pythonic and similar to Jupyter in respect to notebooks, but contains many more prepackaged libraries and functionality. Jupyter is more general and is meant to support interactive computing with many different kernels, not just mathematics in python.
  - **Octave** - An unashamed clone of Matlab.
  - **GeoGebra** - Open source, but not free for commercial applications. Excellent interactive tools and utilities for exploring geometry.
  - **Apache Spark -** Cluster computing for massive datasets.
- Closed Source
  - **Mathematica -** More focused on symbolic computing, but is general purpose. Was one of the first tools to introduce the computational notebook style.
  - **Maple** - Seen as a kindred spirit to Mathematica.
  - **Matlab -** More focused on numerical computing, but is general purpose.

There are many more! Especially for specific areas of mathematics!

KEXP

Setup

KEXP

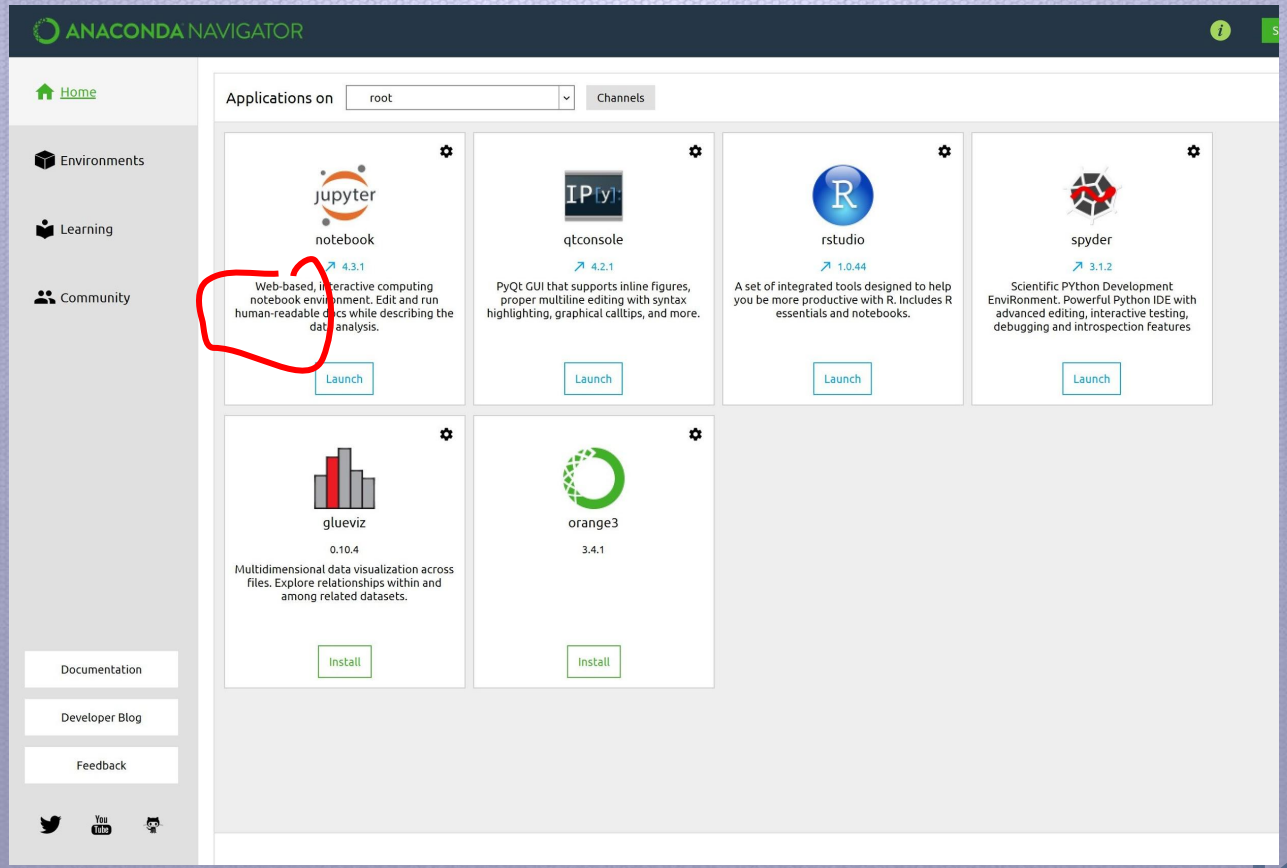
Environment Setup

1. Download this Workshop's Repo as a .zip file: http://bit.ly/2A6dTYp
2. Unzip the workshop .zip in a place you can remember. (Try Documents)
3. Download Anaconda Navigator (AN)

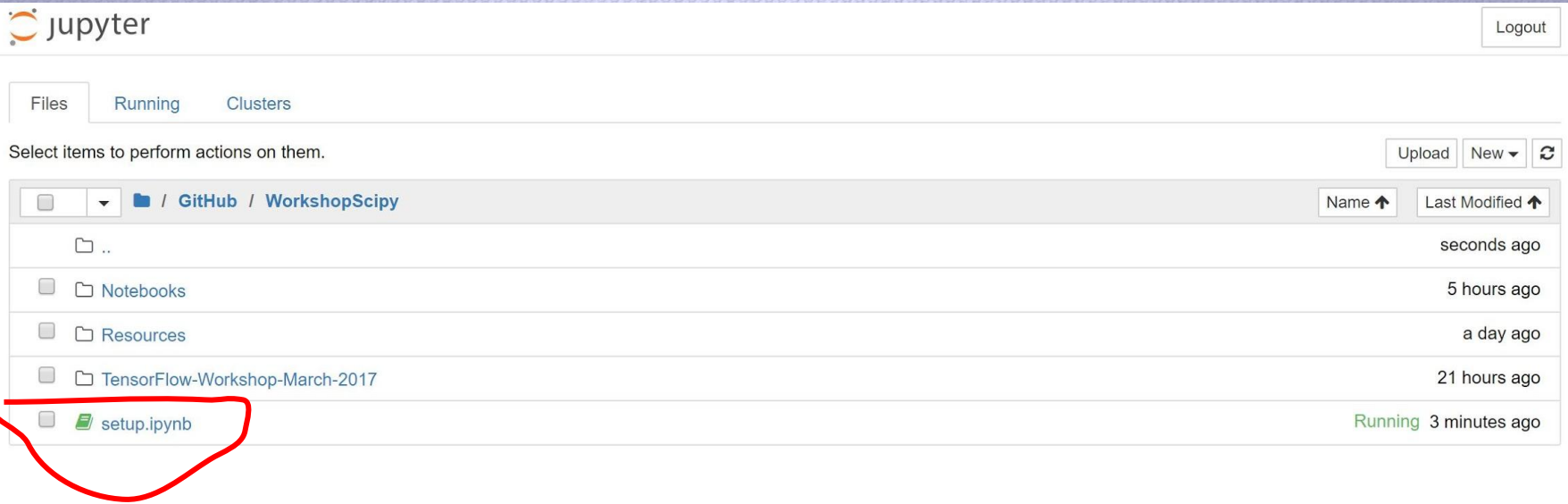
   PYTHON 3.6 Version https://www.continuum.io/downloads
4. Launch the root environment Jupyter Notebook server from the home tab.
5. A browser should have opened up upon launching the Jupyter Notebook server. In that browser, navigate to the workshop folder you unziped.
6. Click on **setup.ipynb** and follow the instructions.

KEXP

Anaconda Navigator : Home



In the Browser



KEXP

Getting Started by Examples

- In the WorkshpScipy/Notebooks folder you will find many example Jupyter notebooks featuring different aspect of working with Python for Scientific Computing.
- The next few slides give brief summaries of each and what you can learn from them.

Notebooks

- **Beginner MNIST** - A TensorFlow tutorial on how to make a simple neural network for classifying MNIST digits.
- **Exercise - Deriving the Quadratic Formula with SymPy** - A tutorial on using SymPy to derive the quadratic formula.
- **Exploring MNIST Manifolds** - Exploring MNIST with Scikit-Learn by applying PCA and K-Means. Also has interactive components.
- **Latex Essentials**- Shows you the basics of using LaTex for typesetting and mathematical notes.
- **Linear Regression** - The Solution Space - Interactive components allow you to explore linear regression. Also shows how to do 3D plotting in matplotlib.
- **Linear Regression** - Gradient Descent - A tutorial on how gradient descent is used to find an optimal linear regression.

Notebooks

- **Linear Vs. Non-Linear Functions** - Shows how to plot in 2D and basic 3D. Also gives you an intuition of the difference between linear and non-linear functions.
- **Matrix as a Function & Plotting Vectors** - Shows how to plot vectors with Matplotlib and shows how a matrix can be thought of as a linear transformation. Uses a lot of Matplotlib.
- **MNIST Probability Experiments 1** - Shows different experiments of computing various statistics on MNIST.
- **Neural Boolean Connectives 1** - Shows a very simple single hidden layer neural network and how it can represent the XOR function. Also shows how it can represent AND.

Notebooks

- **SymPy Basics** - Shows you some fundamental features of SymPy.
- **The Taylor Series** - Uses SymPy to explore the Taylor Series. Also makes use of Matplotlib.
- **Poke Pandas -** A notebook using Pandas to analyse data about pokemon from the pokemon games.

KEXP

Resources & Sources

KEXP

Links

- SciPy - https://www.scipy.org/about.html
- A more complete list of mathematical software - https://en.wikipedia.org/wiki/Mathematical_software

Data: From Secondary Storage to Main Memory

- Generally you do not want to use a database (i.e MySql) or blob storage (i.e. S3) in the loop of some analysis algorithm.
  - For example, imagine having a database where each record is a training example for a neural network. If you queried the database for every example during training, it would be ridiculously inefficient.
  - You want fast access to your data. The fastest access would be data that is already in your system's main memory. Thus having a system that can transfer data from secondary storage to main memory is vital. HDF5 is the SciPy solution to this.
- If your data is absolutely massive in scale, you would need to consider systems designed specifically for this case. Apache Spark is one of the most popular tools for this. Spark bridges the gap between hadoop and fast in-memory computing.
- TensorFlow is also a good choice for cluster computing.

KEXP

Use Colaboratory

- https://colab.research.google.com
- Supports only Python 2.7 and only works on Chrome.
- May not run some of my examples, since I used Python 3.5.

Environment Setup - More Advanced

1. Download Anaconda Navigator (AN).
2. Create a conda environment or use the default environment with many tools already installed.
3. Open a terminal from *AN* and install SciPy. - https://www.scipy.org/install.html

```
python -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose
```

4. Launch the Jupyter notebook server from *AN*.
5. (optional) Install PyCharm Community Edition. A great free IDE for python. Not open source.

KEXP