

Machine Learning With TensorFlow

X433.7-001 (2 semester units in COMPSCI)

Instructor Alexander I. Iliev, Ph.D.

Course Content Outline

- **Optional Reading for the Course**

Title: Python for Data Analysis

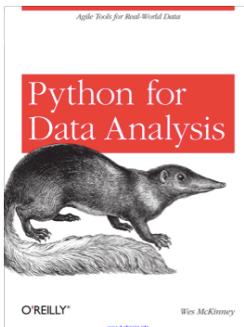
ISBN-10: 1449319793 and ISBN-13: 978-1449319793

Author(s): Wes McKinney

Publisher: O'Reilly Media

Edition Number: 1 edition

Publication Date: November 1, 2012



Title: Tensorflow for Machine Intelligence

ISBN: 9781939902351, ISBN10: 1939902355

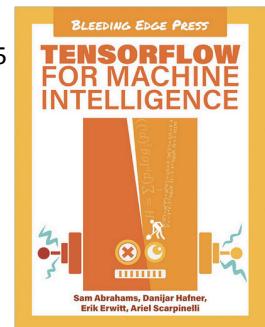
Author(s): Sam Abrahams, Danijar Hafner

Erik Erwitt, Ariel Scarpinelli

Publisher: Bleeding Edge Press

Edition: 1 edition

Publication Date: July 2016



Title: Data Mining

ISBN: 978-0-12-374856-0

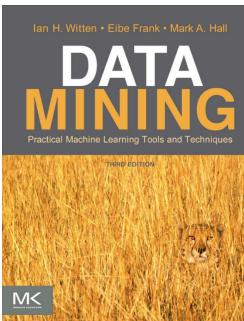
Author(s): Ian H. Witten, Eibe Frank, Mark A. Hall

Publisher: Elsevier

Edition Number: 3rd edition

Publication Date: 2011

Free Online copy



Title: Pattern Classification

ISBN: 111858600X, 9781118586006

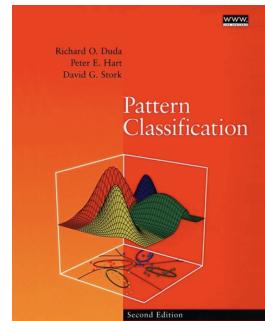
Author(s): Richard O. Duda

Peter E. Hart, David G. Stork

Publisher: John Wiley & Sons, 2012

Edition: 2nd edition

Publication Date: 2012



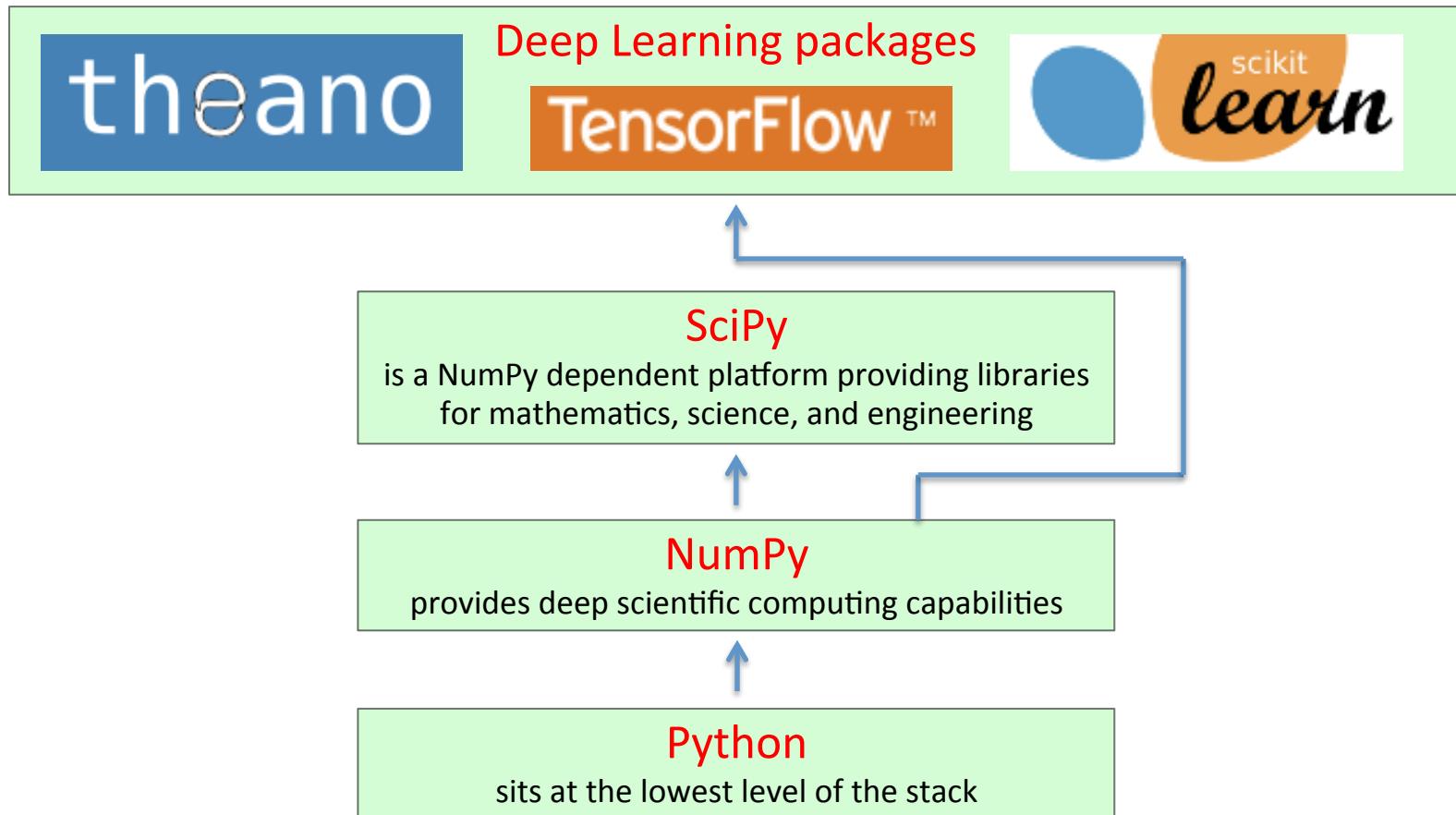
Data Mining

Class 1 ...

Python, TensorFlow, installation and general concepts ...

Machine Learning With TensorFlow

- Basic stack using NumPy, SciPy and Deep Learning in Python:



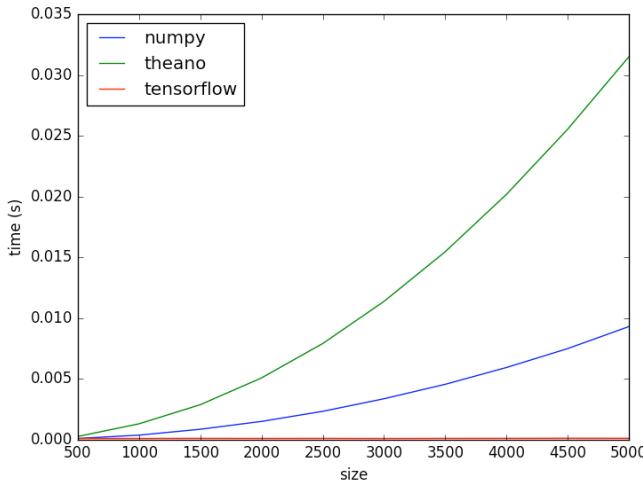
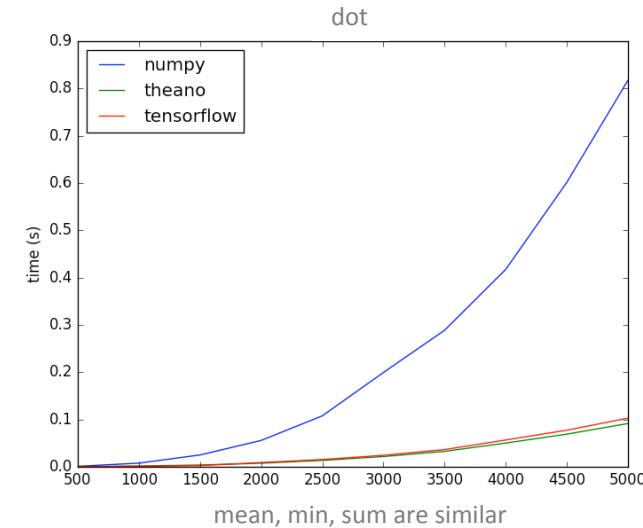
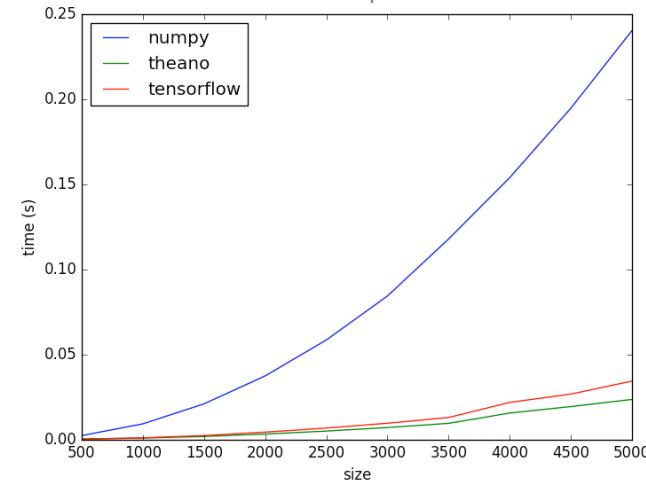
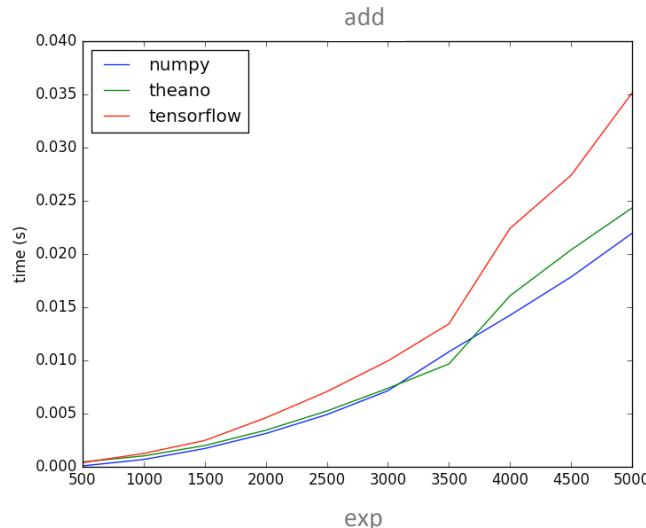
Machine Learning With TensorFlow

- Python + NumPy + SciPy + Theano + Scikit-Learn + Tensorflow
 - ... is like Matlab + toolboxes + DL processing



Machine Learning With TensorFlow

- Testing performance for NumPy + Theano + Tensorflow



Machine Learning

- Working with **Theano**, Scikit-Learn and TensorFlow

Theano logo, the word "theano" in white lowercase letters on a blue rectangular background.

- Theano is a **Python library** that lets you to define, optimize, and evaluate mathematical expressions, especially ones with multi-dimensional arrays (numpy.ndarray)
- Speed is **rivaling C implementations** for problems involving large amounts of data
- It can surpass C on a CPU by many orders of magnitude by **taking advantage of GPUs**
- Theano combines aspects of a **computer algebra system** (CAS) with **optimized compiler**
- It can also **generate customized C code** for many mathematical operations
- Theano can minimize the amount of compilation/analysis overhead

Source: <http://deeplearning.net/software/theano/introduction.html>

Machine Learning

- Working with **Theano**, Scikit-Learn and TensorFlow



- Theano's compiler applies many optimizations of varying complexity that include, but are not limited to:
 - use of GPU for computations
 - constant folding
 - merging of similar subgraphs, to avoid redundant calculation
 - arithmetic simplification (e.g. $x*y/x \rightarrow y$, $--x \rightarrow x$)
 - inserting efficient **BLAS** operations (e.g. **GEMM**) in a variety of contexts
 - using memory aliasing to avoid calculation
 - using inplace operations wherever it does not interfere with aliasing
 - loop fusion for elementwise sub-expressions
 - improvements to numerical stability (e.g. $\log(1 + \exp(x))$ and $\log(\sum_i \exp(x[i]))$)
 - for a complete list, see <http://deeplearning.net/software/theano/introduction.html>

Source: <http://deeplearning.net/software/theano/introduction.html>

Machine Learning

- Working with **Theano**, Scikit-Learn and TensorFlow

theano

- **Theano** is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving **multi-dimensional arrays efficiently**. It features:
 - **tight integration with NumPy** – Use `numpy.ndarray` in Theano-compiled functions
 - **transparent use of a GPU** – Perform data-intensive calculations up to 140x faster than with CPU.(float32 only)
 - **efficient symbolic differentiation**
 - **speed and stability optimizations**
 - **dynamic C code generation** – Evaluate expressions faster
 - **extensive unit-testing and self-verification** – Detect and diagnose many types of errors

Machine Learning

- Working with **Theano**, Scikit-Learn and TensorFlow

Theano logo, the word "theano" in white lowercase letters on a blue rectangular background.

- Theano and TensorFlow are competing heavily for which one is the best platform
- Theano outperforms TensorFlow on a single GPU, but TensorFlow outperforms Theano for parallel execution on multiple GPUs
- Theano has better documentation in general than TensorFlow, although that changes quickly
- Theano has a **native Windows** support
- Both use Numpy arrays
- TensorFlow is more elegant conceptually cleaner than Theano

Machine Learning

- Working with Theano, **Scikit-Learn** and TensorFlow
 - **Classification:** Identifying to which category an object belongs to
 - **Regression:** Predicting a continuous-valued attribute associated with an object
 - **Clustering:** Automatic grouping of similar objects into sets
 - **Dimensionality reduction:** Reducing the number of random variables to consider
 - **Model selection:** Comparing, validating and choosing parameters and models
 - **Preprocessing:** Feature extraction and normalization

Machine Learning

- Working with Theano, **Scikit-Learn** and TensorFlow



Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: SVM, nearest neighbors, random forest, ...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, ridge regression, Lasso, ...

— Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation,

Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, ...

— Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.

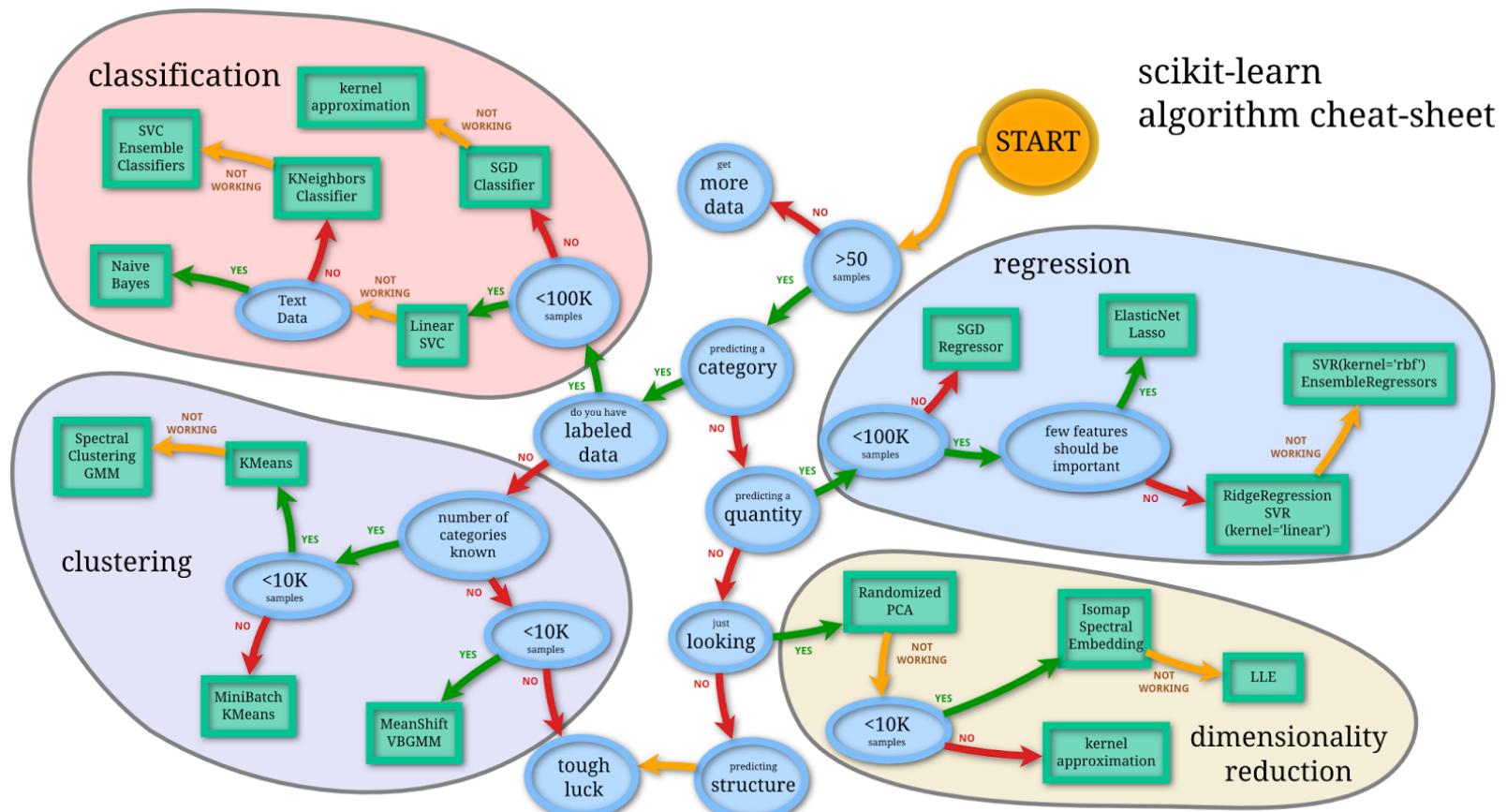
Modules: preprocessing, feature extraction.

— Examples

Source: <http://scikit-learn.org/stable/>

Machine Learning

- Working with Theano, Scikit-Learn and TensorFlow



Machine Learning

- Working with Theano, Scikit-Learn and **TensorFlow**



- TensorFlow™ is an **open source** software library for numerical computation using data flow graphs
- TensorFlow was developed by researchers and engineers working on the **Google Brain Team** within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research
- **Nodes in the graph represent mathematical operations**, while **the graph edges represent the multidimensional data arrays (tensors)** communicated between them
- The **flexible architecture** allows you to deploy computation to one or more **CPUs** or **GPUs** in a desktop, server, or mobile device with a single API

Source: <https://www.tensorflow.org/>

Machine Learning

- Working with Theano, Scikit-Learn and **TensorFlow**



- TensorFlow was originally created by Google as an **internal machine learning tool**
- An implementation of it was open sourced under the Apache 2.0 **License in November 2015**
- TensorFlow is an interface for numerical computation as described in the TensorFlow white paper, and **Google still maintains its own internal implementation** of it
- Google is constantly **pushing internal improvements** to the public repository
- The open source release contains the **same capabilities as Google's internal version**

Machine Learning

- Working with Theano, Scikit-Learn and **TensorFlow**



- TensorFlow's API is **most similar to Theano's**
- TensorFlow does contain a package, "learn" (AKA "Scikit Flow"), that emulates the one-line **modeling functionality of Scikit-Learn**
- TensorFlow provides an extensive suite of functions and classes that allow users to define **models from scratch mathematically**
- This allows users with the appropriate technical background to create customized, flexible models quickly and intuitively
- **TensorFlow is the best library to use for both Research and Production because it scales across multiple GPUs better than Theano does**

Machine Learning With TensorFlow

- Distributions and Dependencies:



- Distributions: Python 2.6+, 3.3+
- Dependencies: Numpy 1.6.1+, Scipy 0.9+



- Distributions: Python 2.6+, 3.3+
- Dependencies: Numpy 1.7.1+, Scipy 0.11+

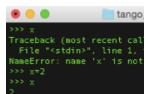


- Distributions: Python 2.7, 3.4, 3.5
- Dependencies: Numpy, bazel, six, wheel

Machine Learning With TensorFlow

- Choosing your environment:

- Using the **shell** (primarily for Linux and OSX users):



- Obtaining Python with core packages:
<https://www.python.org/downloads/>
 - Obtaining the NumPy & SciPy libraries:
<http://www.scipy.org/scipylib/download.html>

- GUI environments using an interactive editor and debuggers for Python:



- **Pyzo** – a free open-source environment based on Python:
<http://www.pyzo.org/downloads.html>

- **Anaconda** – Anaconda is the leading open data science platform powered by Python:
<https://www.continuum.io/>

- **Canopy** – Python based environment providing core scientific analytic and scientific Python packages, for scientific development and visualization:
<https://store.enthought.com/>

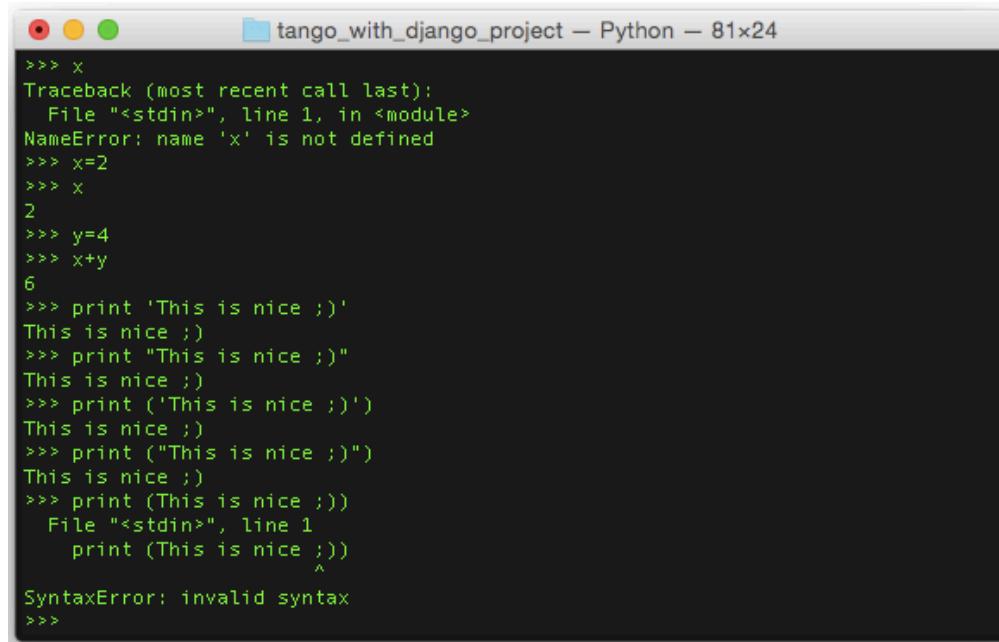


Machine Learning With TensorFlow

- Working with the shell

- The shell:

- present on OSX, Unix and Linux systems
 - provides a quick and easy to use plain text environment for swift execution of simple commands
 - it is flexible and lightweight
 - it has a Unix/Linux look and feel and is not used only for Python and its packages and modules



```
>>> x
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'x' is not defined
>>> x=2
>>> x
2
>>> y=4
>>> x+y
6
>>> print 'This is nice ;)'
This is nice ;)
>>> print "This is nice ;)"
This is nice ;)
>>> print ('This is nice ;)')
This is nice ;)
>>> print ("This is nice ;)")
This is nice ;)
>>> print (This is nice ;))
  File "<stdin>", line 1
    print (This is nice ;))
                           ^
SyntaxError: invalid syntax
>>>
```

Machine Learning With TensorFlow

- Working with the shell
 - Creating the **Virtualenv** environment:
 - to keep our dependencies nice and clean, we're going to be using **virtualenv** to create a virtual Python environment
 - first, we need to make sure that **Virtualenv** is installed along with **pip**, Python's package manager
 - run the following commands (on Mac OS X):

```
$ sudo easy_install pip  
$ sudo pip install --upgrade virtualenv
```

- create a directory to contain this environment:

```
$ sudo mkdir ~/env
```

Machine Learning With TensorFlow

- Working with the shell
 - Creating the **Virtualenv** environment:
 - now we'll create the environment using either the:
 - `virtualenv` command in Python 2 or
 - `venv` module built in Python 3, both located in `~/env/tensorflow`

```
# Python 2.7
$ virtualenv --system-site-packages ~/env/tensorflow
# Python 3
$ sudo python3 -m venv --system-site-packages ~/env/tensorflow
```

- Once it has been created, we can activate the environment using the `source` command:

```
$ source ~/env/tensorflow/bin/activate
# Notice that your prompt now has a '(tensorflow)' indicator
(tensorflow)$
```

Machine Learning With TensorFlow

- Working with the shell
 - Creating the **Virtualenv** environment:
 - make sure that the **environment is active** when we install anything with pip, since that is how **Virtualenv** keeps track of various dependencies
 - when done, shut it off by using the **deactivate** command:

```
(tensorflow)$ deactivate
```
 - create a shortcut for activating it by creating **alias** to your `~/.bashrc` file:

```
$ sudo printf '\nalias tensorflow="source ~/env/tensorflow/bin/activate"' >> ~/.bashrc
```
 - restart your bash:

```
:~ alex$ . ~/.bashrc
```
 - now test it:

```
$ tensorflow
# The prompt should change, as before
(tensorflow)$
```

Machine Learning With TensorFlow

- Working with the shell
 - Installing **TensorFlow**:
 - make sure that your **Virtualenv** environment from the previous section **is active and run the following** command corresponding to your operating system (Mac OS X) and version of Python:

```
# Mac OS X, Python 2.7:  
(tensorflow)$ pip install --upgrade https://storage.googleapis.com/tensorflow/mac/tensorflow-0.9.0-py2-none-any.whl  
  
# Mac OS X, Python 3.4+  
(tensorflow)$ pip3 install --upgrade https://storage.googleapis.com/tensorflow/mac/tensorflow-0.9.0-py3-none-any.whl
```

Machine Learning With TensorFlow

- Working with the shell
 - Troubleshooting:
 - firstly, you might want to upgrade your pip:
`sudo pip install --upgrade pip`
 - you might get the following message, despite that you "Successfully installed six-1.11.0 tensorflow-1.6.0" in Virtualdev:
Illegal instruction: 4 when importing tensorflow in python 3.6
 - there is a bug in TF1.6, so change the TensorFlow version by typing:

`>>> sudo pip install -Iv tensorflow==1.5`
`>>> sudo pip install -Iv numpy==1.13 → ... may not be necessary`
 - you can also try to uninstall TensorFlow first, but it may not be necessary:

`>>> sudo pip uninstall tensorflow → ... you may use pip3 too`

Machine Learning With TensorFlow

- Working with the shell
 - Running TensorFlow on **iPython**:
 - in case you'd like to run TensorFlow in iPython you may have to check where TensorFlow is running from:

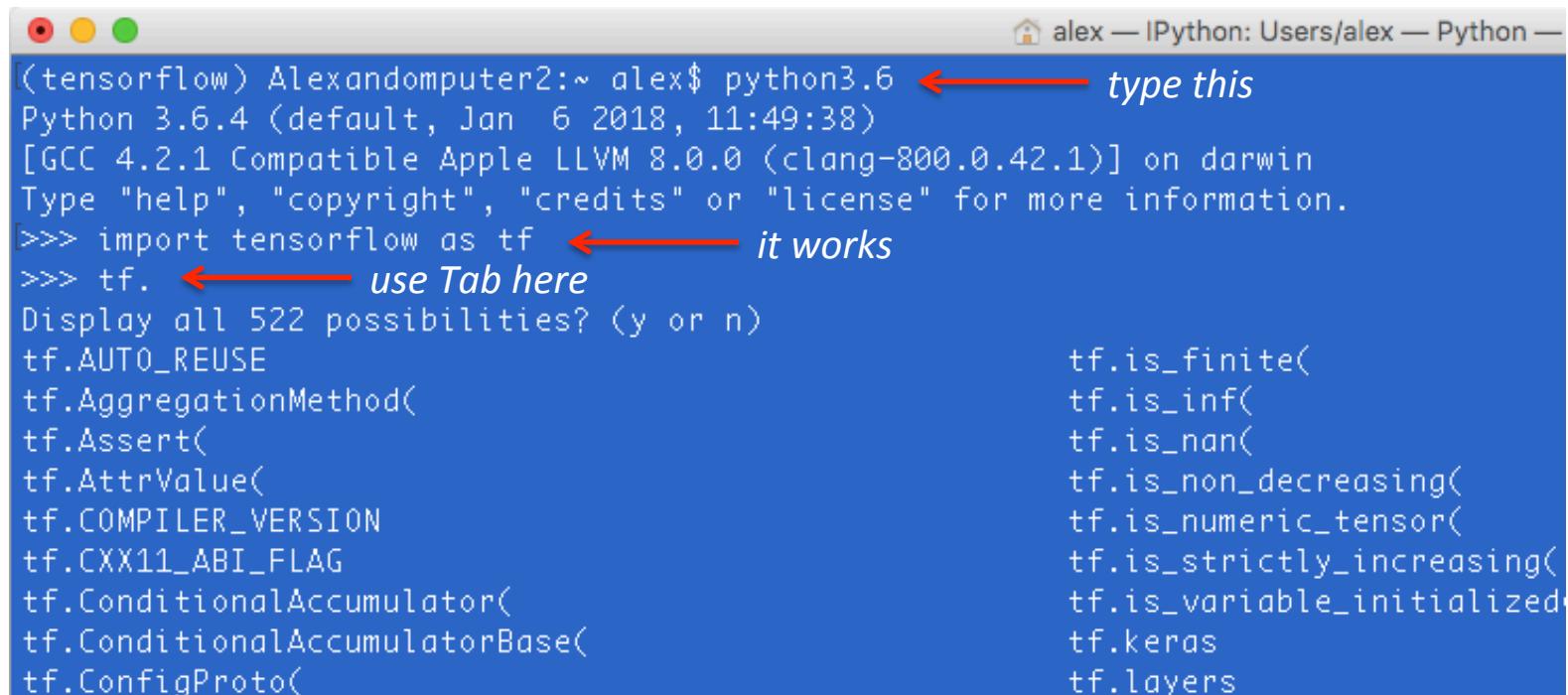
```
(tensorflow) Alexandomputer2:~ alex$ which python  
→ /Users/alex/env/tensorflow/bin/python  
(tensorflow) Alexandomputer2:~ alex$ which ipython  
→ /usr/local/bin/ipython
```
 - Since they run from different locations, iPython may not have the path to run TensorFlow so it is best to reinstall iPython like this in **virtualenv**:

```
(tensorflow) Alexandomputer2:~ alex$ sudo pip3 install --ignore-installed ipython
```

- You can also try to change the path too

Machine Learning With TensorFlow

- Working with the shell
 - Try TensorFlow on **virtualenv** and **Python3.6**:



A screenshot of a terminal window titled "alex — IPython: Users/alex — Python —". The window shows the following Python session:

```
(tensorflow) Alexandromputer2:~ alex$ python3.6
Python 3.6.4 (default, Jan  6 2018, 11:49:38)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> tf.  ← use Tab here
Display all 522 possibilities? (y or n)
tf.AUTO_REUSE
tf.AggregationMethod(
tf.Assert(
tf.AttrValue(
tf.COMPLIER_VERSION
tf.CXX11_ABI_FLAG
tf.ConditionalAccumulator(
tf.ConditionalAccumulatorBase(
tf.ConfigProto(
tf.is_finite(
tf.is_inf(
tf.is_nan(
tf.is_non_decreasing(
tf.is_numeric_tensor(
tf.is_strictly_increasing(
tf.is_variable_initialized(
tf.keras
tf.layers
```

Annotations with red arrows and text:

- An arrow points to "python3.6" with the text "type this".
- An arrow points to "tf." with the text "it works".
- An arrow points to "tf." with the text "use Tab here".

Machine Learning With TensorFlow

- Working with the shell
 - Try TensorFlow on **virtualenv** and **iPython**:

The screenshot shows a Jupyter Notebook cell with the following content:

```
(tensorflow) Alexandomputer2:~ alex$ ipython ← type this
/usr/local/lib/python3.6/site-packages/IPython/core/interactiveshell.py:763: UserWarning: Attempting to work in a virtualenv. If you encounter problems, please "Python 3.6.4 (default, Jan 6 2018, 11:49:38)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.4.0 -- An enhanced Interactive Python. Type '?' for help.

[In [1]: import tensorflow as tf ← it works

In [2]: tf. ← use Tab here
          tf.AggregationMethod           tf.NodeDef
          tf.Assert                     tf.OpError
          tf.AttrValue                  tf.Operation
          tf.ConfigProto                tf.OptimizerOptions
          tf.DType                      tf.PaddingFIFOQueue
          tf.DeviceSpec                 tf.Print
          tf.Dimension                  tf.QUANTIZED_DTYPES
          tf.Event                      tf.QueueBase
          tf.FIFOQueue
```

Annotations with red arrows:

- An arrow points to the command `ipython` in the terminal prompt, labeled *type this*.
- An arrow points to the successful import statement `[In [1]: import tensorflow as tf`, labeled *it works*.
- An arrow points to the start of the code block in In [2], labeled *use Tab here*.

Machine Learning With TensorFlow

- Working with the shell

- To see the available packages on `virtualenv` and `iPython`:

```
[In [2]: pip list ← this will not work, so we use 'help' ...
```

The following command must be run outside of the IPython shell:

```
$ pip list
```

- To check all available modules we type either:

```
[In [3]: help('modules')
```

Please wait a moment while I gather a list of all available modules...

we see ... `numpy` ... `matplotlib` ... `tensorflow` ... `tensorboard`

- Or:

```
In [4]: import ← this is a tab completion
```

IPython	alembic
abc	antigravity
absl	appnope
aifc	argparse

Machine Learning With TensorFlow

- Working with the shell
 - Installing Jupyter notebook:
 - First, make sure you have installed iPython (on both Python 2 and 3):

```
# Python 2.7
$ sudo python2 -m pip install ipykernel
$ sudo python2 -m ipykernel install
# Python 3
$ sudo python3 -m pip install jupyterhub notebook ipykernel
$ sudo python3 -m ipykernel install
```

- Then install all dependencies:
- use pip to install the Jupyter Notebook (or pip3 for Python 3):

```
# For Python 2.7
$ sudo pip install jupyter
# For Python 3
$ sudo pip3 install jupyter
```

Jupyter

Machine Learning With TensorFlow

- Working with the shell
 - Installing **Matplotlib** and final check:
 - You should be able to see Jupyter in your environment:

```
In [3]: help('modules')
Please wait a moment while I gather a list of all available modules...
... jupyter
```

- Then install Matplotlib:

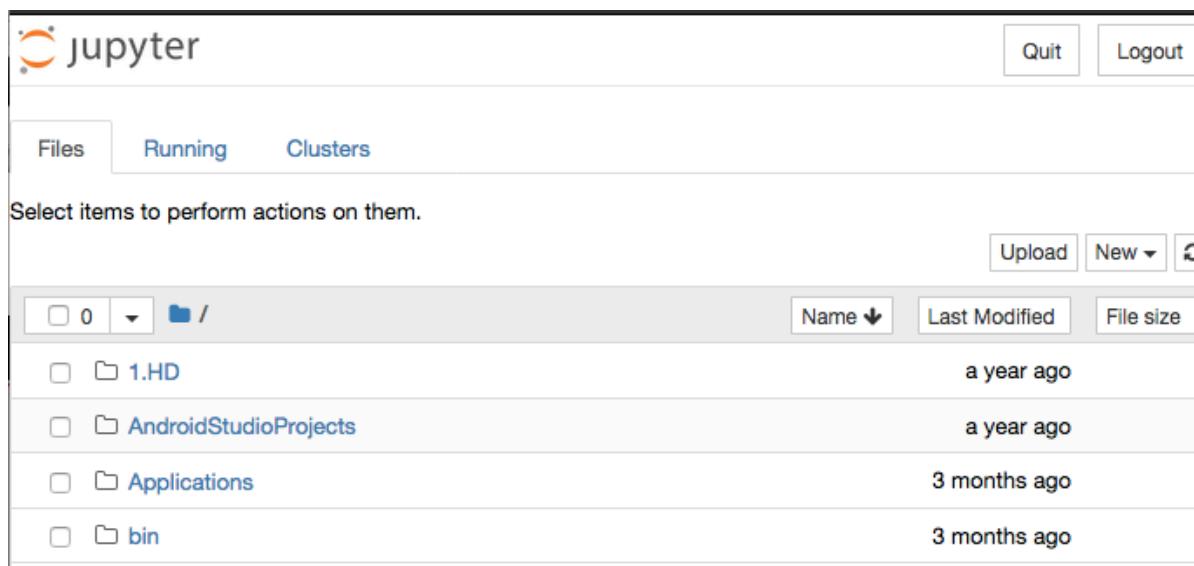
```
# Python 2.7
$ sudo apt-get build-dep python-matplotlib python-tk
# Python 3
$ sudo apt-get build-dep python3-matplotlib python3-tk
```

- Now check if it works:

```
(tensorflow)$ jupyter notebook
```

Machine Learning With TensorFlow

- Working with the shell
 - **Testing** your Jupyter environment:
 - You should get a page like this in your browser:

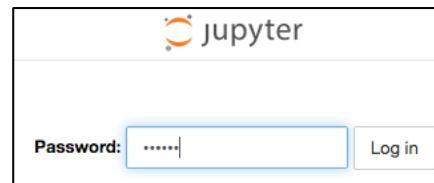


- With URL: <http://localhost:8888/tree>

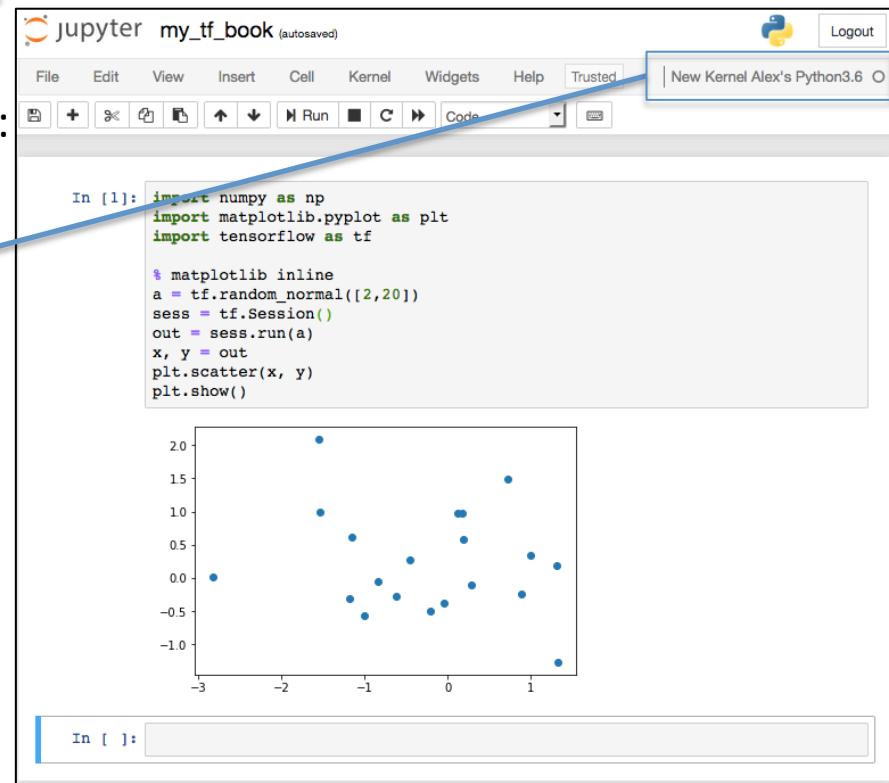
Machine Learning With TensorFlow

- Working with the shell
 - Testing your Jupyter environment:
 - You will be prompted to login:

```
alex$ jupyter notebook --NotebookApp.token=qwerty
```



- Then run any file you might have:



The screenshot shows a Jupyter Notebook interface with the following details:

- Kernel:** New Kernel Alex's Python3.6
- Code Cell:** In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

%matplotlib inline
a = tf.random_normal([2,20])
sess = tf.Session()
out = sess.run(a)
x, y = out
plt.scatter(x, y)
plt.show()
```

- Output:** A scatter plot showing 20 data points in a 2D space. The x-axis ranges from -3 to 1, and the y-axis ranges from -1.0 to 2.0. The points are scattered randomly.

... see next slide

Machine Learning With TensorFlow

- Working with the shell
 - Testing your Jupyter environment:
 - Here is how to customize your **preferred kernel**, its **source** and **name**:

```
(tensorflow) Alexandromputer2:~ alex$ nano ~/Library/Jupyter/kernels/python3.6/kernel.json
```

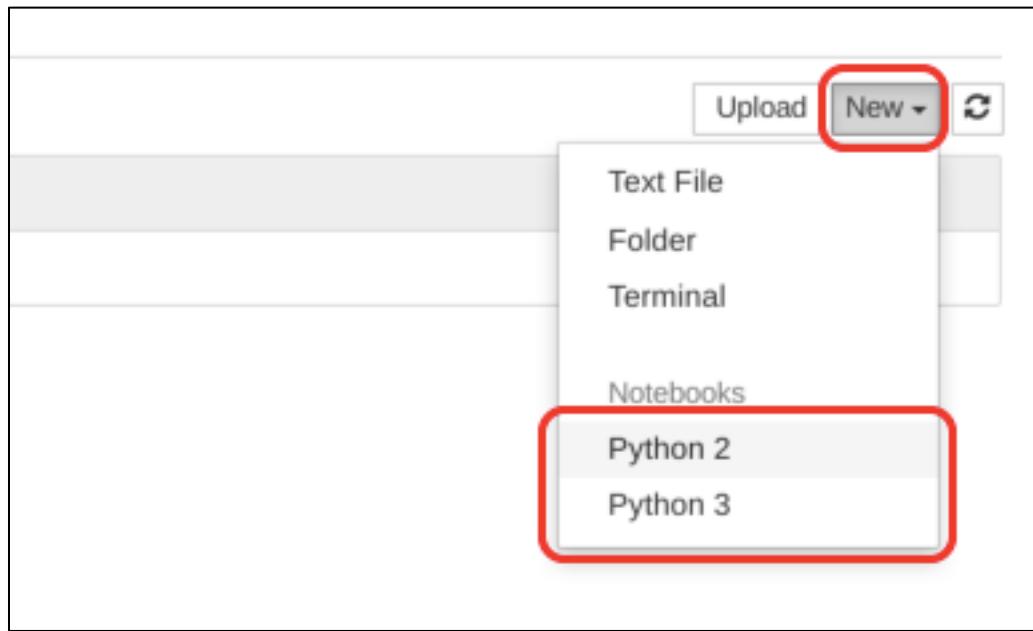


```
alex — nano ~/Library/Jupyter/kernels/python3.6/kernel.json
GNU nano 2.0.6 File: .../python3.6/kernel.json

{
  "display_name": "New Kernel Alex's Python3.6",
  "language": "python",
  "argv": [
    "/usr/local/bin/python3.6",
    "-m",
    "ipykernel",
    "-f",
    "{connection_file}"
  ]
}
```

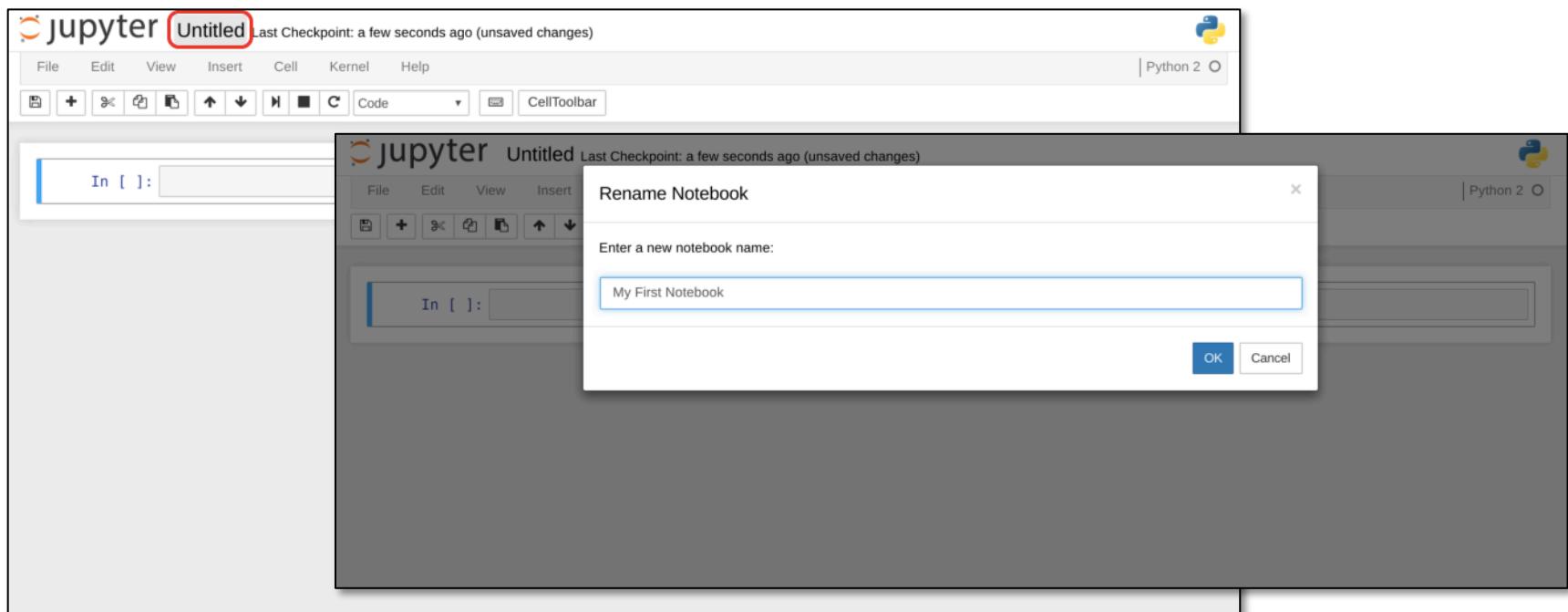
Machine Learning With TensorFlow

- Working with the shell
 - Testing your Jupyter environment:
 - You can start a new file using a different Python version (2 or 3):



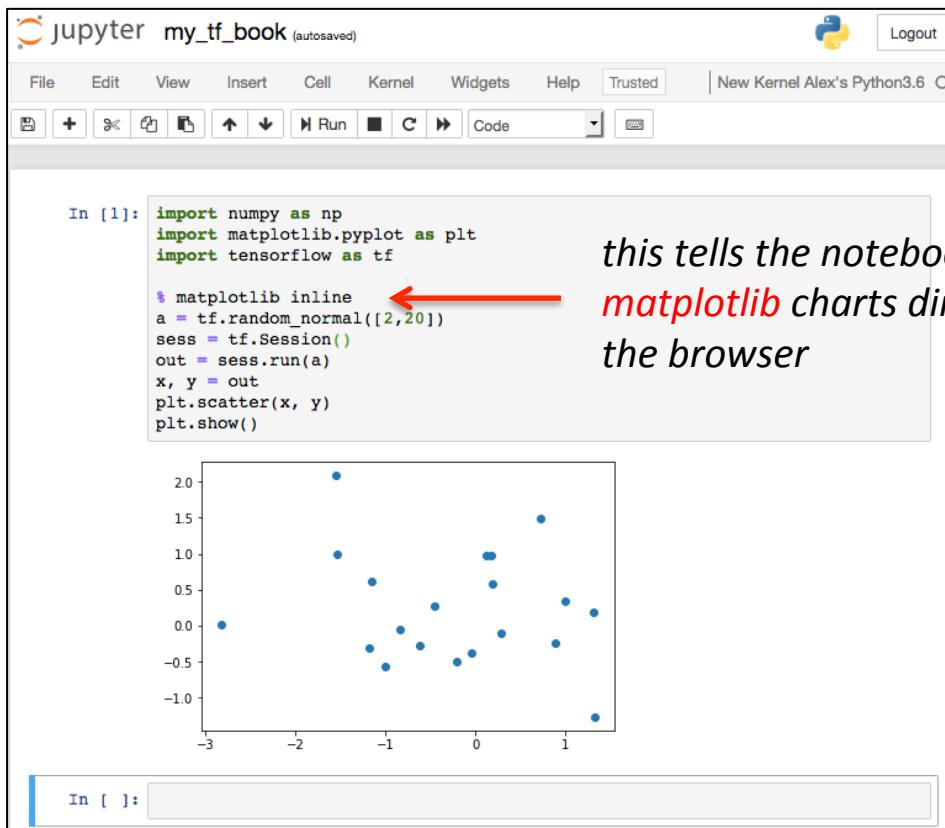
Machine Learning With TensorFlow

- Working with the shell
 - Testing your Jupyter environment:
 - Then you can begin working with it after giving it a name:



Machine Learning With TensorFlow

- Working with the shell
 - Testing your Jupyter environment:
 - You can type the code directly in the field or cut and paste it:



The screenshot shows a Jupyter Notebook interface with the title "jupyter my_tf_book (autosaved)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Logout. Below the toolbar is a toolbar with various icons for file operations. The main area shows a code cell labeled "In [1]:" containing Python code. A red arrow points to the line "%matplotlib inline". The code imports numpy, matplotlib.pyplot, and tensorflow, generates random data, creates a session, runs the session, and plots the data using plt.scatter. To the right of the code cell, a text box contains the annotation: "this tells the notebook to display matplotlib charts directly inside the browser". Below the code cell is a scatter plot with x and y axes ranging from -3 to 1. The plot shows several blue data points.

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

%matplotlib inline
a = tf.random_normal([2,20])
sess = tf.Session()
out = sess.run(a)
x, y = out
plt.scatter(x, y)
plt.show()
```

this tells the notebook to display matplotlib charts directly inside the browser

Machine Learning With TensorFlow

- Working with Pyzo

- Pyzo ... what?

- **Pyzo** is a **free Open-source** BSD-licensed
 - It is an easy to use **GUI** computing environment **based on Python 3**
 - It is built to make Python's potential for **scientific and engineering computing** to be easily accessible
 - Pyzo is combined with IEP to make a powerful easy to use comprehensive environment for scientific computing and visualization
 - Pyzo is a **comprehensive development environment** pre-built with:
 - many scientific packages including **NumPy**, **SciPy** and **Matplotlib**
 - uses **IEP as the front-end IDE**
 - **IPython** and a **notebook**
 - Complete packages for Pyzo are given here: <http://www.pyzo.org/packages.html>



PYZO: PYTHON TO THE PEOPLE

Machine Learning With TensorFlow

- Working with Pyzo



BETA

- [Download](#)
- [About Python](#)
- [About Pyzo](#)
- [Learn](#)
- [FAQ](#)

Packages

Pyzo comes with a range of packages for general functionality, science, development, testing, UI design, etc. ... Click on any of the links to go to the projects website where you will find more information such as tutorials, documentation, and mailing lists.

The Scipy Stack:

- [numpy](#) - using arrays in Python
- [scipy](#) - core scientific functionality
- [matplotlib](#) - 2D plotting library
- [pandas](#) - data structures and analysis
- [sympy](#) - symbolic math
- [nose](#) - software testing
- [IPython](#) - advanced interactive shell

Further scientific packages:

- [pyopengl](#) - talk to your GPU
- [visvis](#) - 3D plotting and visualization
- [skimage](#) - 2D and 3D image processing
- [sklearn](#) - machine learning
- [imageio](#) - read and write image data

Machine Learning With TensorFlow

- Working with Pyzo
 - Useful magic commands:

Linux-like:

- pwd
- ls
- mkdir
- rmdir
- whos
- dir()
- cd
- history

Python IEP/Pyzo specific:

- edit *file*
- del(parameter)
- reset
- reset -f
- run *file*
- time
- colors (nocolor,linux,lightbg)
- dhist



PYZO: PYTHON TO THE PEOPLE

Tip: Using a magic command followed by '?' will give you help for it. Example: 'cd?'

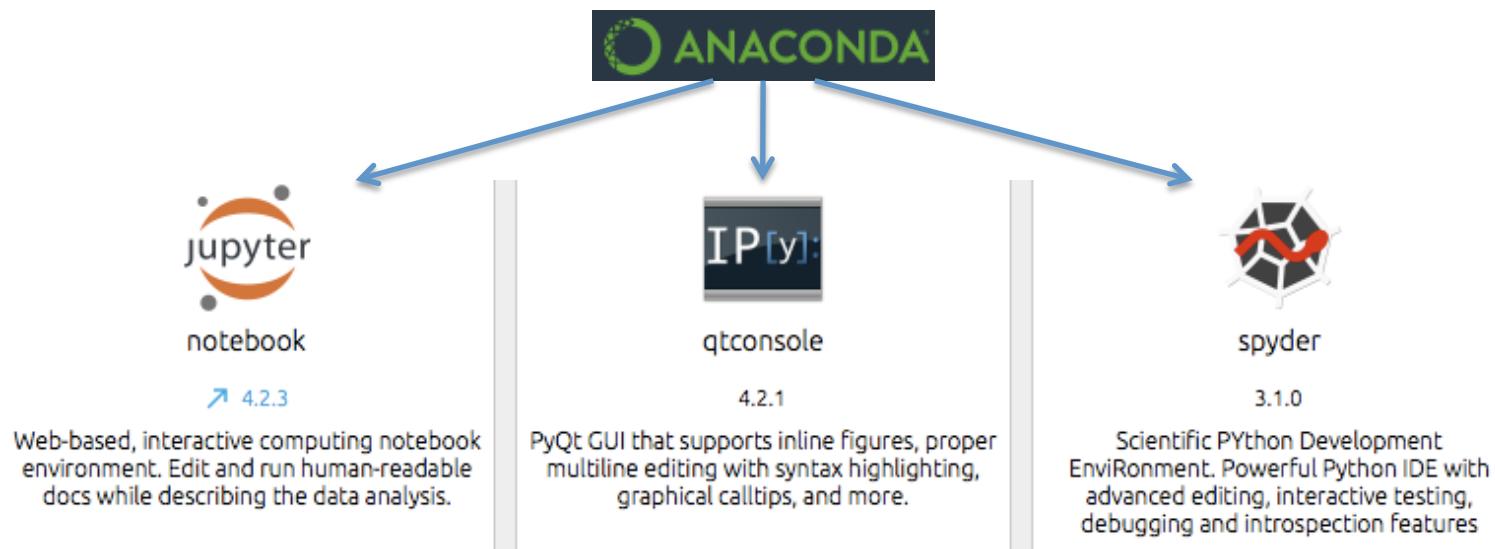
magic comprehensive list: <http://ipython.org/ipython-doc/3/interactive/magics.html>

Machine Learning With TensorFlow

- Working with Pyzo
 - Troubleshooting (*recall*)
 - you might get the following message, while trying to import tensorflow in Pyzo:
Illegal instruction: 4 when importing tensorflow in python 3.6
 - Just as before, there is a bug in TensorFlow 1.6, so change the TensorFlow version by typing:
`>>> sudo pip install -Iv tensorflow==1.5`

Machine Learning With TensorFlow

- Choosing your environment:
 - GUI environments using an interactive editor and debuggers for Python:
 - **Anaconda** – Anaconda is the leading open data science platform powered by Python:
<https://www.continuum.io/>



Machine Learning With TensorFlow

- Working with alternative environments:
 - Canopy – scientific and analytic Python package distribution. **has a free version, but limited**

It offers:

- object-oriented prog.
- MATLAB-like syntax
- many packages
- graphical debugger is **NOT** free

The screenshot shows the Enthought Canopy product page. At the top, there are tabs for 'For Individuals', 'For Enterprises', 'For Academics', and a link 'Already a Canopy subscriber? Download here'. Below this is a grid of four pricing plans: 'CANOPY EXPRESS' (FREE), 'CANOPY WITH PYTHON ESSENTIALS' (\$199 / year), 'CANOPY WITH PYTHON FOUNDATIONS ALL ACCESS' (\$678 / year), and 'CANOPY TRIPLE PLAY' (\$1,125 / year). A yellow button below the plans says 'Contact us for discounts for 5+ users'. Below the plans are buttons for 'Free Download' (blue) and 'Add to Cart' (green) for each plan. A section titled 'Pre-built and tested Python packages' compares '100+ Core' (light gray) and '300+ Extended' (light green). A blue button 'See included Python package details' with a plus sign is located to the right. At the bottom, a section titled 'Integrated Analysis Environment' lists features: 'Graphical Debugger', 'Integrated IPython', 'Advanced Code Editor', and 'Application Development Platform'. Each feature has a column of five dots representing different levels or editions across the four plans.

For Individuals	For Enterprises	For Academics	Already a Canopy subscriber? Download here	
	CANOPY EXPRESS FREE	CANOPY WITH PYTHON ESSENTIALS \$199 / year	CANOPY WITH PYTHON FOUNDATIONS ALL ACCESS \$678 / year	CANOPY TRIPLE PLAY \$1,125 / year
		Contact us for discounts for 5+ users		
	Free Download	Add to Cart	Add to Cart	Add to Cart
Pre-built and tested Python packages	100+ Core	300+ Extended		
			See included Python package details	+
Integrated Analysis Environment	Graphical Debugger	Integrated IPython	Advanced Code Editor	Application Development Platform
	● ● ● ● ●	● ● ● ● ●	● ● ● ● ●	● ● ● ● ●

Machine Learning With TensorFlow

- Working with alternative environments:
 - Canopy – scientific and analytic Python package distribution. **has a free version, but limited**

Package Manager:

- Packages are installed through Package Manager

The screenshot shows the Enthought Canopy Package Manager interface. At the top, there's a navigation bar with 'Refresh' and 'Not logged in.' status. A search bar contains the text 'seaborn'. Below the search bar, the main title is 'Package Manager' with the subtitle 'Install, update or remove your Python packages'. On the left, there's a sidebar with tabs: 'Installed' (0/106), 'Available' (1/494, highlighted in green), 'Updates' (0/0), 'History' (document icon), and 'Settings'. The main content area displays a table with columns 'Package Name' and 'Latest Available Version'. One row is shown for 'seaborn' with version '0.7.0-6'. At the bottom, there's a detailed view for the 'seaborn' package. It says 'No summary available.' Under 'Installed:', it says 'Currently not installed.' Under 'Available on store: (enthought/free)', it shows '0.7.0-6' with a dropdown arrow. There's a blue 'Install v0.7.0-6' button. At the very bottom, it says 'No description available.'

Machine Learning With TensorFlow

- Working with alternative environments:
 - Canopy – scientific and analytic Python package distribution. **has a free version, but limited**

Using the browser

The screenshot shows a Jupyter Notebook interface with two code cells. The top cell, labeled 'In [*]', contains code to import various Python libraries and set up a Matplotlib font cache. A blue arrow points from the text 'the line is in a process of executing' to the first line of this code cell. The bottom cell, labeled 'In [15]', contains code to set better defaults for Matplotlib, including setting the axes face color to white and the font size to 18. The notebook title is 'Titanic Berkeley Project V2 - final' and the kernel is 'Python 2'. The status bar at the bottom right shows 'Python 2'.

```
#Importing all the libraries needed
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import brewer2mpl
import seaborn as sns
from scipy import stats
from scipy import linalg

/Users/alex/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-packages/matplotlib/font_manager.py:273: UserWarning: Matplotlib is building the font cache using fc-list. This may take a moment.
    warnings.warn('Matplotlib is building the font cache using fc-list. This may take a moment.')

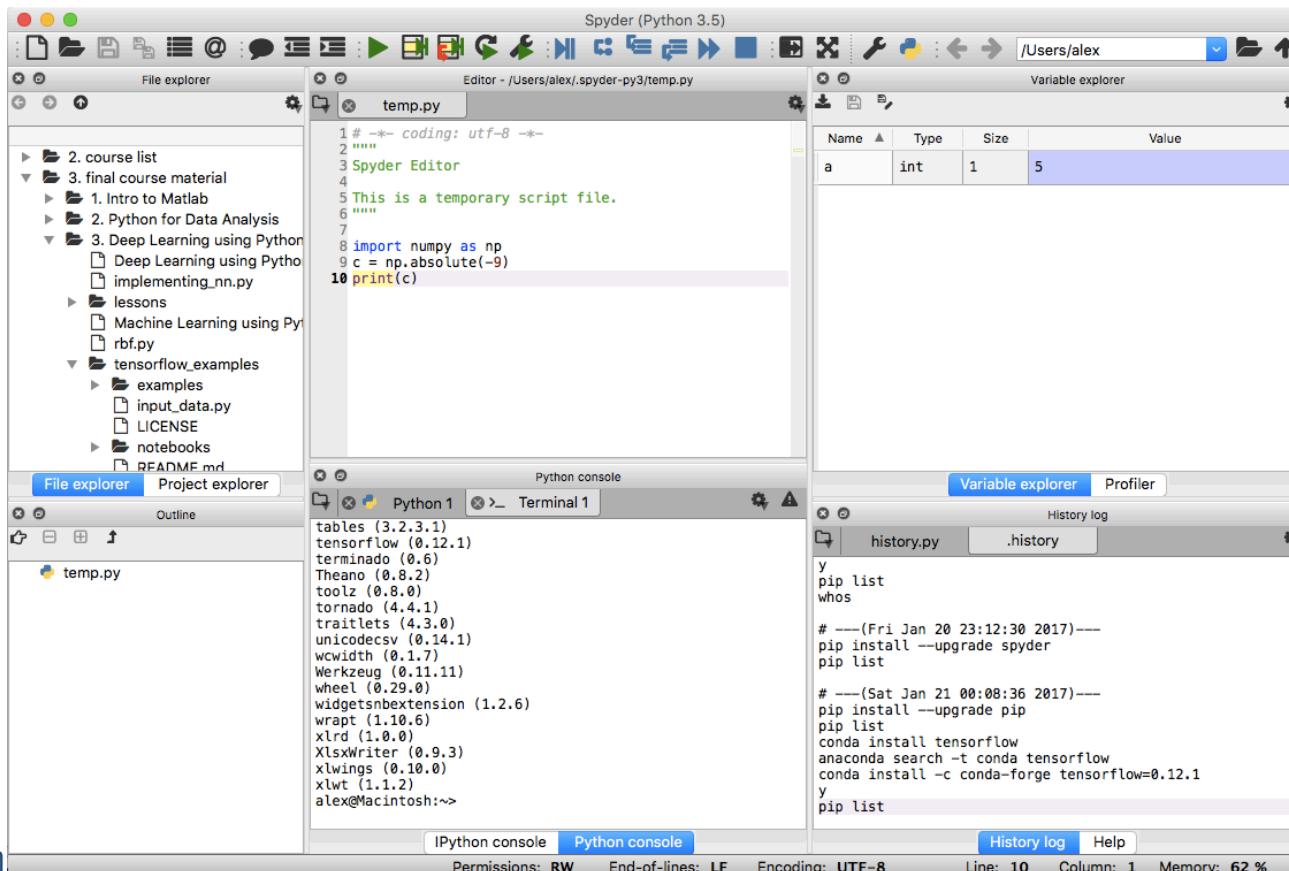
# Setting up better defaults for matplotlib
from matplotlib import rcParams

rcParams['axes.facecolor'] = 'white'
rcParams['font.size'] = 18

#colorbrewer2 Dark2 qualitative color table
dark2_colors = brewer2mpl.get_map('Dark2', 'Qualitative', 7).mpl_colors
```

Machine Learning With TensorFlow

- Choosing your environment:
 - GUI environments using an interactive editor and debuggers for Python:
 - **Spyder** – a GUI environment for interactive Python use with shells (terminal- and Qt-based):



Machine Learning With TensorFlow

- Working with Canopy
 - Useful magic commands:

Linux-like:

- pwd
- ls
- mkdir
- rmdir
- whos
- who()
- dir()
- cd
- history

iPython specific:

- edit *file*
- del(parameter)
- reset
- reset -f
- run *file*
- time
- dhist

How to create an alias:

- [35] %alias clc clear

- Tips:
1. Using a magic command followed by ‘?’ will give you help for it. Example: ‘cd?’
 2. Esc will get rid of the help menu

magic comprehensive list: <http://ipython.org/ipython-doc/3/interactive/magics.html>

Machine Learning

- Working with Anaconda Navigator:



- Scikit-learn **comes installed** with Anaconda distribution **by default**:

A screenshot of the Anaconda Navigator application interface. The top bar shows the title "Anaconda Navigator - Beta". On the left, there's a sidebar with icons for Home, Environments, Learning, and Community. The main area has a search bar labeled "Search Environments" and a dropdown menu set to "Installed". Below that, a table lists packages. A search bar at the top of the table has "sciki" typed into it, with a blue arrow pointing to it from the text above. The table has columns for Name, Description, and Version. Two rows are visible: "scikit-image" (Version 0.12.3) and "scikit-learn" (Version 0.18.1). Both rows have a green checkmark icon and a green circular icon next to the package name.

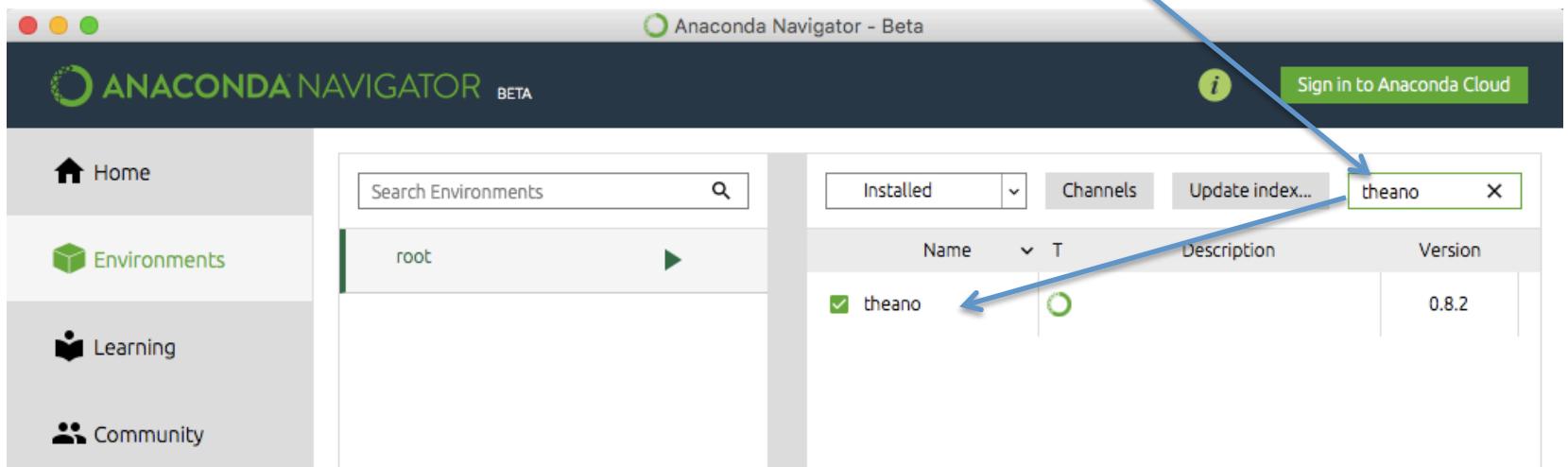
Name	Description	Version
scikit-image	Image processing routines for scipy	0.12.3
scikit-learn	Set of python modules for machine learning and data mining	0.18.1

Machine Learning

- Working with Anaconda Navigator:



- Theano is **not installed in Anaconda by default**, but is part of the standard repository and can be installed through the menu:

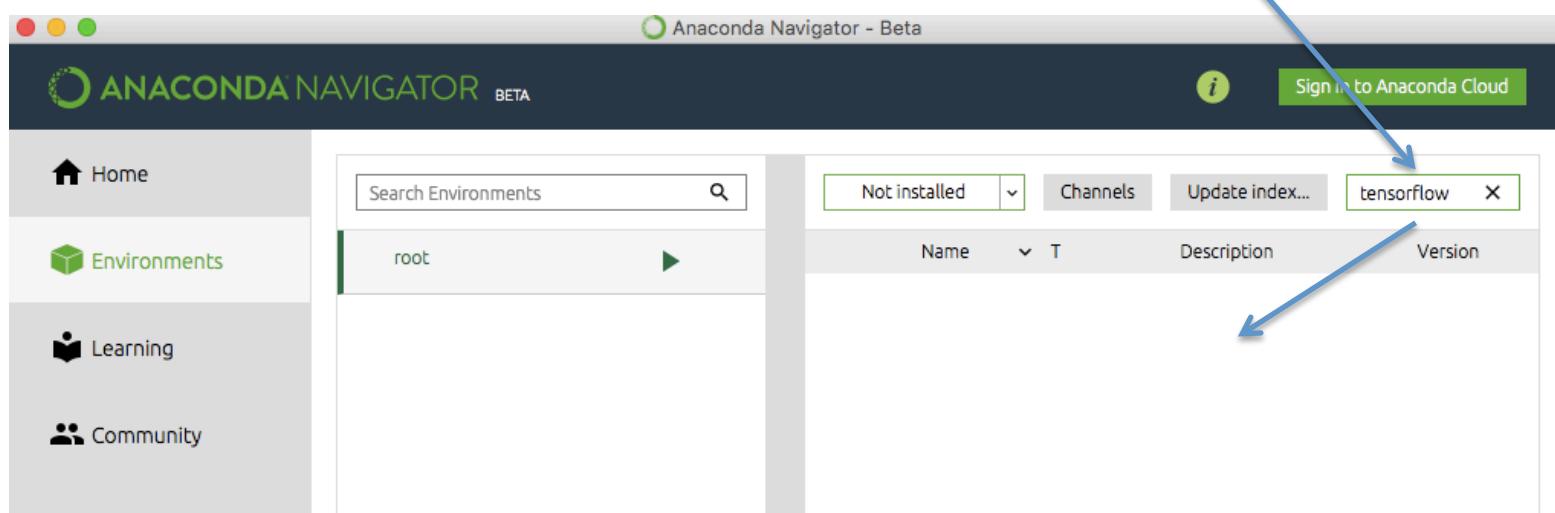


Machine Learning With TensorFlow

- TensorFlow in Anaconda:



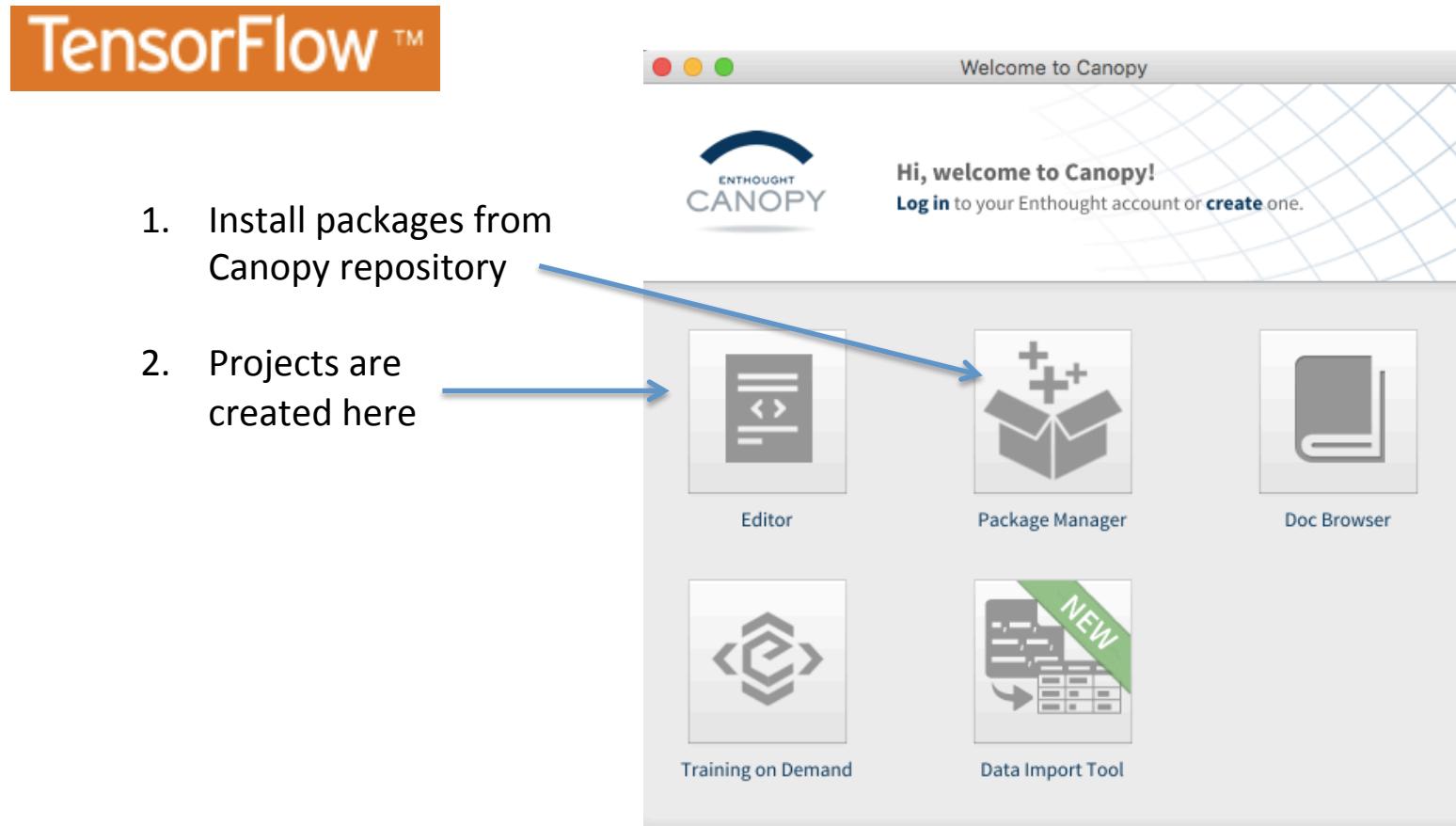
- TensorFlow **is not part of the standard Anaconda repository**:



- So you can **install it via Spyder's terminal** by typing:
`>>> conda install -c conda-forge tensorflow=0.12.1`

Machine Learning With TensorFlow

- TensorFlow in Canopy:



Machine Learning With TensorFlow

- TensorFlow in Canopy:

TensorFlow™

The screenshot shows the Enthought Canopy Package Manager interface. On the left, a sidebar has tabs for 'Installed' (7/137), 'Available' (9/564), 'Updates' (0/0), 'History', and 'Settings'. A blue arrow points from the 'Settings' tab towards the main content area. The main area is titled 'Package Manager' and says 'Install, update or remove your Python packages'. A search bar contains 'ten'. The table lists installed packages with their names, versions, and details. The 'tensorflow' row is highlighted with a yellow background and circled in red. The 'tensorflow' row details show it is up-to-date (1.3.0-2) and provide options to 'Uninstall' or 'Re-install v1.3.0-2'. The bottom section for 'tensorflow' shows 'No summary available.'

	Package Name	Installed Version
envisage	4.6.0-1	
nose	1.3.7-3	
protobuf	3.3.1-1	
pygments	2.1.3-1	
python_dateutil	2.6.0-1	
tensorflow	1.3.0-2	
widgetsnbextension	2.0.0-1	

Machine Learning With TensorFlow

- TensorFlow in Canopy:

TensorFlow™

→	numpy	1.13.3-1
→	matplotlib	2.0.0-5
→	scikit_learn	0.19.1-2
	scikits.image	0.13.0-5
	scikits.learn	0.8-2
→	scipy	1.0.0-1

Machine Learning With TensorFlow

- TensorFlow in Canopy:

The screenshot shows the Canopy IDE interface. A blue arrow points from the text "TensorFlow in Canopy:" to the left side of the interface, which contains the File Browser and the code editor. Another blue arrow points from the text "Machine Learning With TensorFlow" to the right side of the interface, which contains the interactive console.

File Browser: Shows a list of recent files under the "Recent Files" section, including "helloworld.py", "my_tf_book.py", "main.py", "Concept01_defining_tensors.ipynb", "moving_avg.py", "types.py", "interactive_session.py", "gradient.py", "my_tf_book.ipynb", "my_tf_book.py", "boston-marathon.py", "Soumyendu-Sarkar-compsci.ipynb", "Soumyendu-Sarkar-compsci.ipynb", "FinalProject.ipynb", "FinalProjectCombined-Copy1.ipynb", "final project part-2 edit.ipynb", "FinalProjectCombined-Copy1.ipynb", "Titanic Berkeley Project V2.ipynb", "double_pendulum.py", "Single Electron Schrodinger Eqn.ipynb", "compsciX433_projectnc.ipynb", "animation.py", "final project.ipynb", "Titanic Berkeley Project V2 - final.ipynb", and "Titanic V0.2b-2.ipynb".

Code Editor: The file "helloworld.py" is open, displaying the following code:

```
1 ''' HelloWorld example using TensorFlow library '''
2
3 import tensorflow as tf
4
5 # 1. Create a Constant op
6 # The op is added as a node to the default graph.
7 #
8 # 2. The value returned by the constructor is the output of the Constant op.
9 hello = tf.constant('Hello, TensorFlow!')
10
11 # Start tf session
12 sess = tf.Session()
13
14 # Run the op
15 print(sess.run(hello))
```

Interactive Console: Displays the following text:

```
Welcome to Canopy's interactive data-analysis environment!
Type '?' for more information.
Python 2.7.11 | 64-bit | (default, Jun 11 2016, 03:41:56)
Type "copyright", "credits" or "license" for more information.

IPython 4.1.2 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

In [1]: %cd "/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/examples/1_Introduction"
/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/examples/1_Introduction

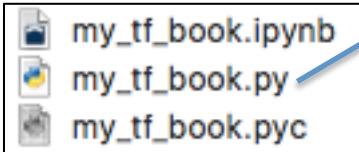
In [2]:
```

Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

1. as script file



The screenshot shows the Canopy IDE interface. On the left is a 'File Browser' window showing a directory structure under 'alex' with various Python files listed. A blue arrow points from the 'my_tf_book.ipynb' file in the browser to the code editor. The code editor window has tabs for 'helloworld.py' and 'my_tf_book.py'. The 'my_tf_book.py' tab is active, displaying the following Python code:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import tensorflow as tf
4
5 # This is only for Jupyter plotting in-line:
6 # %matplotlib inline
7
8 a = tf.random_normal([2,20])
9 sess = tf.Session()
10 out = sess.run(a)
11 x, y = out
12 plt.scatter(x, y)
13 plt.show()
```

Below the code editor is a terminal window showing Python command-line interactions:

```
In [1]: %cd "/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/examples/1_Introduction"
/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/examples/1_Introduction

In [2]: %cd "/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/tensorflow_examples/alex_examples"
[Errno 2] No such file or directory: '/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/tensorflow_examples/alex_examples'
/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/examples/1_Introduction

In [3]: %cd "/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/alex_examples"
/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/alex_examples

In [4]:
```

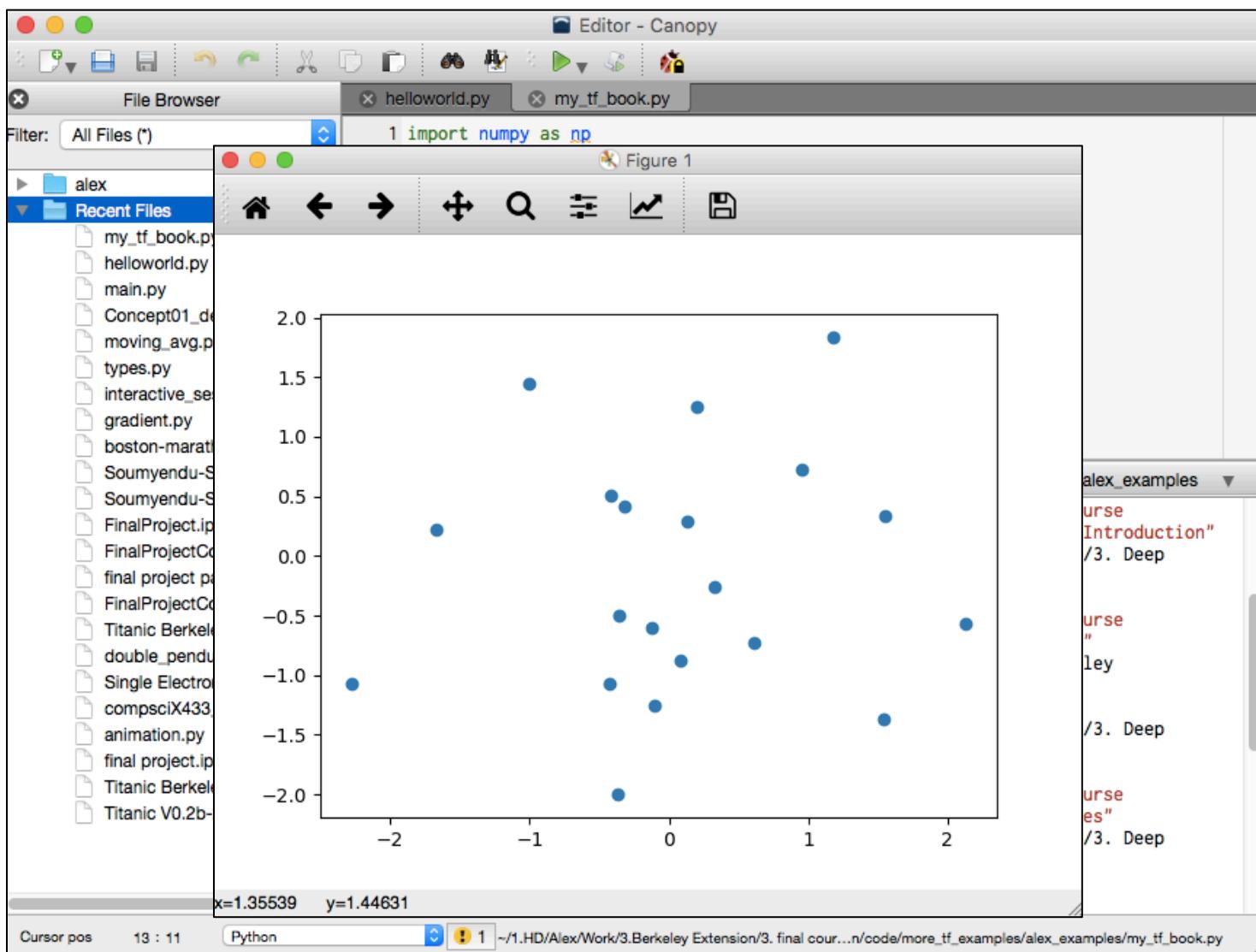
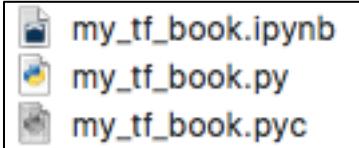
At the bottom, status bar information includes 'Cursor pos 13 : 11', 'Python', and the file path '~1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/alex_examples/my_tf_book.py'.

Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

1. as script file

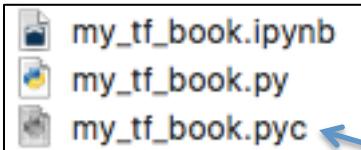


Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

2. as imported file
(creates compiled version .pyc)



File Browser

- alex
- Recent Files
 - my_tf_book.py
 - helloworld.py
 - main.py
 - Concept01_defining_tensors.ipynb
 - moving_avg.py
 - types.py
 - interactive_session.py
 - gradient.py
 - boston-marathon.py
 - Soumyendu-Sarkar-compsciX433_projectnc.ipynb
 - Soumyendu-Sarkar-compsciX433_projectnc.ipynb
 - FinalProject.ipynb
 - FinalProjectCombined-Copy1.ipynb
 - final project part-2 edit.ipynb
 - FinalProjectCombined-Copy1.ipynb
 - Titanic Berkeley Project V2.ipynb
 - double_pendulum.py
 - Single Electron Schrodinger Eqn.ipynb
 - compsciX433_projectnc.ipynb
 - animation.py
 - final project.ipynb
 - Titanic Berkeley Project V2 - final.ipynb
 - Titanic V0.2b-2.ipynb

Editor - Canopy

helloworld.py

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import tensorflow as tf
4
5 # This is only for Jupyter plotting in-line:
6 # %matplotlib inline
7
8 a = tf.random_normal([2,20])
9 sess = tf.Session()
10 out = sess.run(a)
11 x, y = out
12 plt.scatter(x, y)
13 plt.show()
```

Python ~/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/alex_examples

```
In [6]: pwd
Out[6]: u'/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/alex_examples'

In [7]: ls
My First Notebook.ipynb  my_tf_book.py
my_tf_book.ipynb        my_tf_book.pyc

In [8]: import my_tf_book
In [9]: run my_tf_book
In [10]: |
```

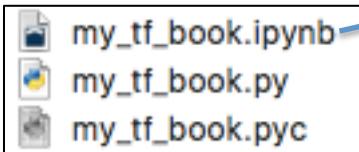
Cursor pos 9 : 18 Python 1 ~/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/alex_examples/my_tf_book.py

Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

3. in notebook
(it is a .ipynb file)



The screenshot shows the Canopy IDE interface. On the left is the 'File Browser' pane, which lists several Python files and notebooks. In the center is the 'Editor - Canopy' window, showing a snippet of Python code for generating a scatter plot. On the right is the IPython notebook interface, displaying the execution history of cells.

File Browser:

- alex
- Recent Files
 - my_tf_book.ipynb
 - helloworld.py
 - main.py
 - Concept01_defining_tensors.ipynb
 - moving_avg.py
 - types.py
 - interactive_session.py
 - gradient.py
 - boston-marathon.py
 - Soumyendu-Sarkar-compsci-1.ipynb
 - Soumyendu-Sarkar-compsci-1.ipynb
 - FinalProject.ipynb
 - FinalProjectCombined-Copy1.ipynb
 - final project part-2 edit.ipynb
 - FinalProjectCombined-Copy1.ipynb
 - Titanic Berkeley Project V2.ipynb
 - double_pendulum.py
 - Single Electron Schrodinger Eqn.ipynb
 - compsciX433_projectnc.ipynb
 - animation.py
 - final project.ipynb
 - Titanic Berkeley Project V2 - final.ipynb
 - Titanic V0.2b-2.ipynb

Editor - Canopy:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import tensorflow as tf
4
5 # This is only for Jupyter plotting in-line:
6 # %matplotlib inline
7
8 a = tf.random_normal([2,20])
9 sess = tf.Session()
10 out = sess.run(a)
11 x, y = out
12 plt.scatter(x, y)
13 plt.show()
```

IPython Notebook:

```
In [6]: pwd
Out[6]: u'/Users/alex/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more_tf_examples/alex_examples'

In [7]: ls
My First Notebook.ipynb  my_tf_book.py
my_tf_book.ipynb        my_tf_book.pyc

In [8]: import my_tf_book

In [9]: run my_tf_book

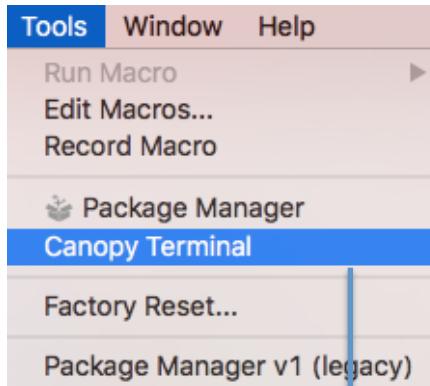
In [10]: |
```

Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

3. in notebook
(it is a .ipynb file)



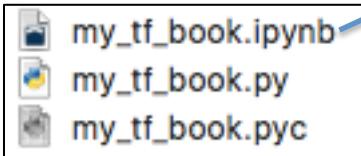
```
(Canopy 64bit) Alexandromputer2:/ alex$ jupyter notebook --NotebookApp.token=qwerty
[I 12:54:41.481 NotebookApp] The port 8888 is already in use, trying another port.
[I 12:54:41.552 NotebookApp] Serving notebooks from local directory: /Users/alex
[I 12:54:41.552 NotebookApp] 0 active kernels
[I 12:54:41.552 NotebookApp] The Jupyter Notebook is running at: http://localhost:8889/?token=...
[I 12:54:41.552 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to
skip confirmation).
[I 12:54:42.385 NotebookApp] 302 GET /tree (::1) 2.08ms
[I 12:54:45.776 NotebookApp] 302 POST /login?next=%2Ftree (::1) 2.72ms
```

Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

3. in notebook
(it is a .ipynb file)



The screenshot shows a Jupyter Notebook interface titled "jupyter my_tf_book". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Help, and a Python 2 option. The toolbar includes icons for file operations like Open, Save, and Run, along with CellToolbar and Cell buttons.

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

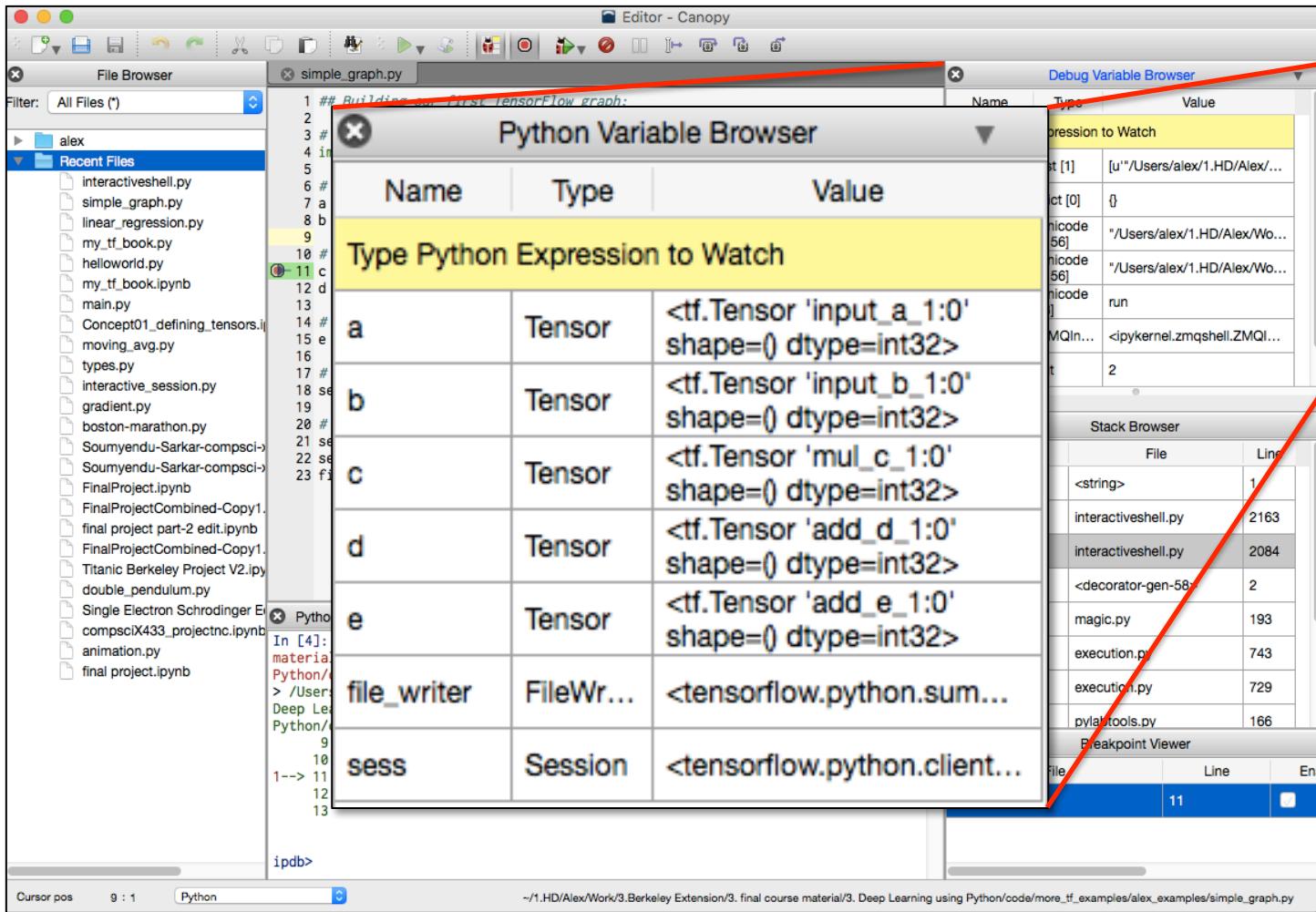
%matplotlib inline
a = tf.random_normal([2,20])
sess = tf.Session()
out = sess.run(a)
x, y = out
plt.scatter(x, y)
plt.show()
```

The code imports numpy, matplotlib.pyplot, and tensorflow, then uses %matplotlib inline to display plots directly in the notebook. It generates a tensor of random normal numbers and runs it through a session to get the actual values, which are then plotted as a scatter plot of x vs y. The plot shows two clusters of points centered around (0,0) and (1,1).

In []:

Canopy debugger

- Running in debug mode:



Machine Learning With TensorFlow

- **tf.Session:**

- **Type:** type
- A class for running TensorFlow operations.
- A `Session` object encapsulates the environment in which `Operation` objects are executed, and `Tensor` objects are evaluated.

For example:

```
# Build a graph.  
a = tf.constant(5.0)  
b = tf.constant(6.0)  
c = a * b  
sess = tf.Session() # Launch the graph in a session.  
print(sess.run(c)) # Evaluate the tensor `c`.
```

Machine Learning With TensorFlow

- Artificial Intelligence concept
 - It is a concept conceived in the mid 50s with the intention to construct complex machines (computers) that possessed the same characteristics of human intelligence
 - The main idea is to create technologies able to perform specific tasks better than humans can
 - Some of the first intelligent robots emerged in movies such as Star Wars – C-3PO
 - Artificial intelligence (AI) is already part of our everyday lives
 - Some examples of AI are: image classification, face recognition, voice recognition, speaker recognition, emotion recognition, etc.

Machine Learning With TensorFlow

- **Data Mining** and Machine Learning concepts
 - In data mining, the data is stored electronically and the search is automated (or semi-automated)
 - It has been estimated that the amount of data stored in the world's databases doubles every 20 months ...
 - ... Thus the opportunities for data mining increase
 - **Data mining** is about solving problems by analyzing data already present in databases
 - **Data mining** finds patterns that can be analyzed to identify distinguishing characteristics

Machine Learning With TensorFlow

- Data Mining and Machine Learning concepts
 - Machine learning came directly from the early AI scientists. The algorithmic approaches over the years included: *decision tree learning, logic programming, clustering, reinforcement learning, Bayesian networks*, etc.
 - To “learn” by definition for us humans means to:
 - Obtain knowledge of something and being aware of something
 - Be informed, receive instructions, commit to memory
 - An operational definition for “learn” can be formulated like:
 - Things learn when they change their behavior in a way that makes them perform better in the future. The definition of “better” is given by us and means “performance” rather than “knowledge”

Machine Learning With TensorFlow

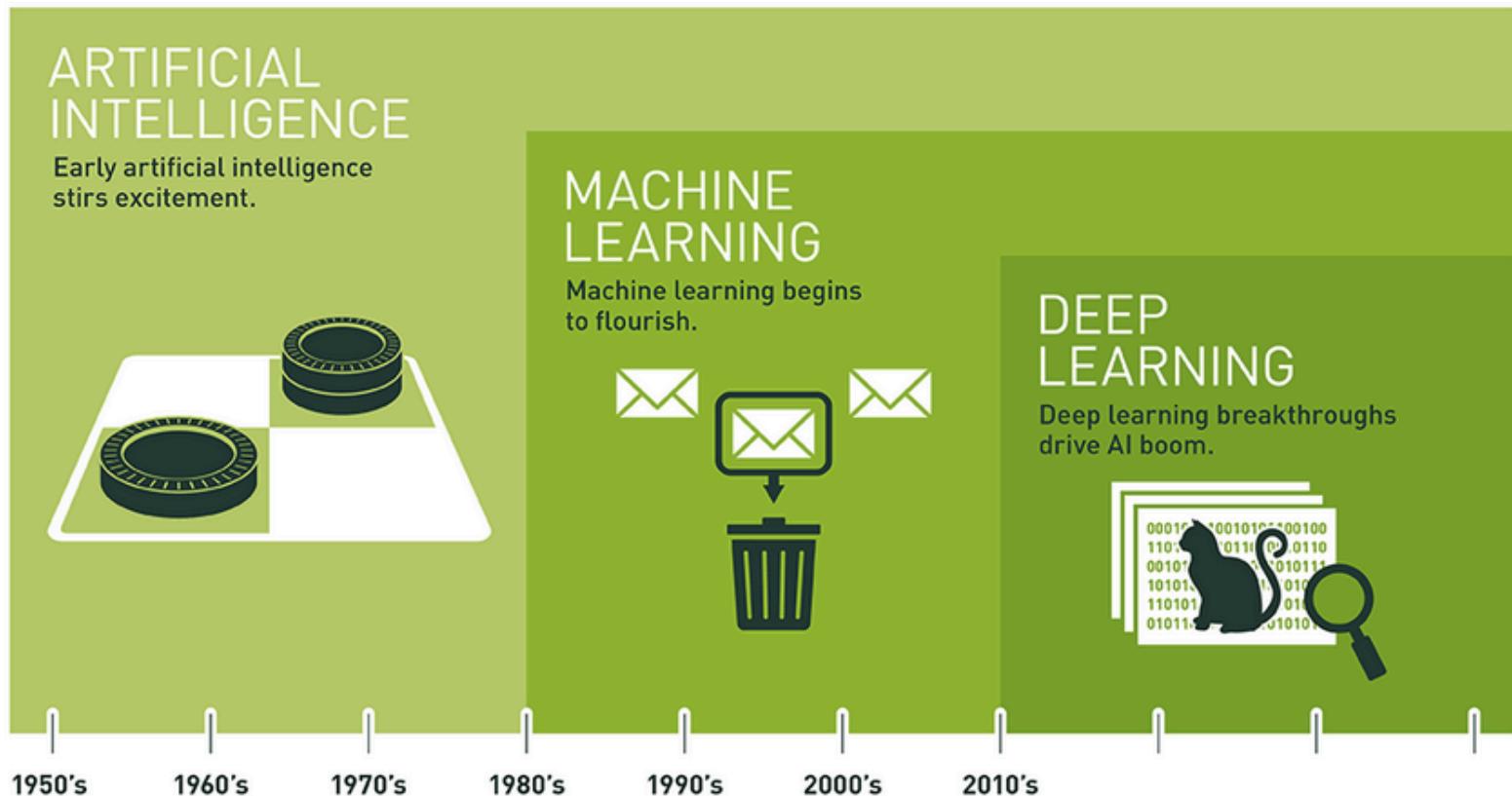
- Data Mining and Machine Learning concepts
 - Machine learning does not entail the conceptual limitations mentioned herein, and disregards any particular philosophical stance about what learning actually is
 - Machine learning is therefore a practical process connected to Data mining while using algorithms to parse data, learn from it, and then make a prediction
 - Performance:
 - the machine is “trained” using large amounts of data and algorithms that give it the ability to learn;
 - it is then “tested” on a new (usually unknown) set of data

Machine Learning With TensorFlow

- Deep Learning concept
 - Deep Learning is an area in Machine Learning that has been introduced with the objective of moving Machine Learning closer to its goal: Artificial Intelligence
 - Deep Learning has enabled many practical applications of Machine Learning and by extension the overall field of AI
 - Deep Learning breaks down tasks in ways that makes all kinds of machine assists seem possible
 - Areas of implementation are infinite, but some of them are: preventive healthcare, security systems, robotics, cars without drivers, better media recommendations, etc.

Machine Learning With TensorFlow

- Deep Learning concept



Machine Learning With TensorFlow

HW assignment 1:

Chosen, install and configure your Python-Tensorflow environment