

# Machine Learning With TensorFlow

X433.7 (2 semester units in COMPSCI)

Instructor Alexander I. Iliev, Ph.D.

# Course Content Outline

- **Machine Learning With TensorFlow®**
  - Introduction, Python - pros and cons
  - Python modules, DL packages and scientific blocks
  - Working with the shell, IPython and the editor
  - Installing the environment with core packages
  - Writing “Hello World”
- **Tensorflow and TensorBoard basics**
  - Ecosystem, Competition, Users
  - Linear algebra recap
  - Data types in Numpy and Tensorflow
  - Basic operations in Tensorflow
  - Graph models and structures with Tensorboard
- **TensorFlow operations**
  - Overloaded operators
  - Using Aliases
  - Sessions, graphs, variables, placeholders
  - Name scopes
- **Data Mining and Machine Learning concepts**
  - Basic Deep Learning Models, k-Means
  - Linear and Logistic Regression
  - Softmax classification
- **Neural Networks 1/2**
  - Multi-layer Neural Network
  - Gradient descent and Backpropagation

HW1 (10pts)

HW2 (10pts)

# Course Content Outline

- **Neural Networks 2/2**
    - Object recognition with Convolutional Neural Network (CNN)
    - Activation Functions
  - **Working with images**
    - Loading Images
    - Image formats and manipulation
  - **CNN Implementation**
    - Training
    - Recurrent Neural Network (RNN)
    - Project Presentations 1/2
  - **Project Presentations**
    - Project Presentation 2/2
- Midterm / Project proposal due (30pts)
- Final Project (40pts)

# Course Content Outline

- **Methods of Instruction**

Lectures and in-class discussions will be the main tools of instruction. Homework problems will help students absorb the material and get to practice in their free time. Since this is a more specialized course that naturally delves into specific topics a midterm exam will be held to assess the performance of the students midway through the course.

- **Credit Requirements**

Students must complete all homework assignments and pass all exams. They also must receive a passing grade on the final exam to receive a passing grade in the course.

- **Course Grade Weighting**

- |                            |     |   |          |   |                            |
|----------------------------|-----|---|----------|---|----------------------------|
| 1. In-class participation: | 10% |   |          |   |                            |
| 2. Homework:               | 20% | - | Thursday | - | use the weekend to prepare |
| 3. Midterm                 | 30% | - | Tuesday  | - | use the weekend to prepare |
| 4. Final Project           | 40% | - | Tuesday  |   |                            |

# Course Content Outline

- **Optional Reading for the Course**

Title: Python for Data Analysis

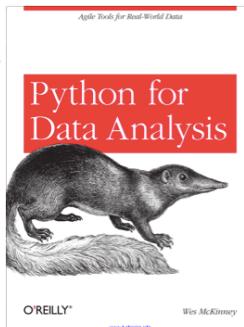
ISBN-10: 1449319793 and ISBN-13: 978-1449319793

Author(s): Wes McKinney

Publisher: O'Reilly Media

Edition Number: 1 edition

Publication Date: November 1, 2012



Title: Data Mining

ISBN: 978-0-12-374856-0

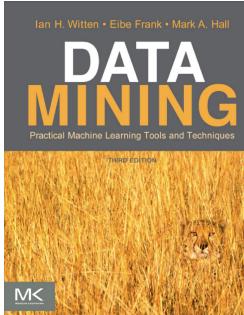
Author(s): Ian H. Witten, Eibe Frank, Mark A. Hall

Publisher: Elsevier

Edition Number: 3rd edition

Publication Date: 2011

[Free Online copy](#)



Title: Tensorflow for Machine Intelligence

ISBN: 9781939902351, ISBN10:1939902355

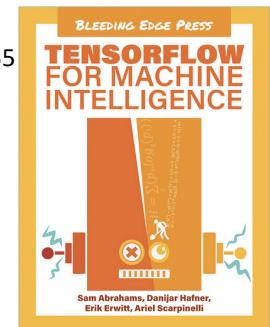
Author(s): Sam Abrahams, Danijar Hafner

Erik Erwitt, Ariel Scarpinelli

Publisher: Bleeding Edge Press

Edition: 1 edition

Publication Date: July 2016 ([too old!](#))



(optional reading)

Title: Pattern Classification

ISBN: 111858600X, 9781118586006

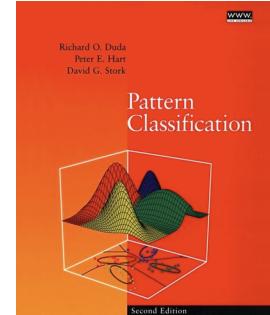
Author(s): Richard O. Duda

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 ...*

# Data

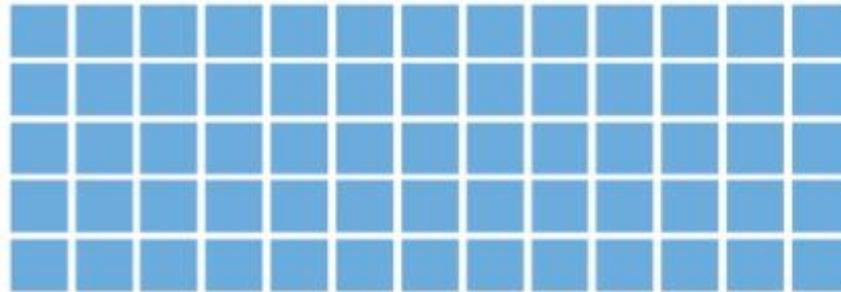
- The world's digital content and media is **growing rapidly** at a never stopping rate



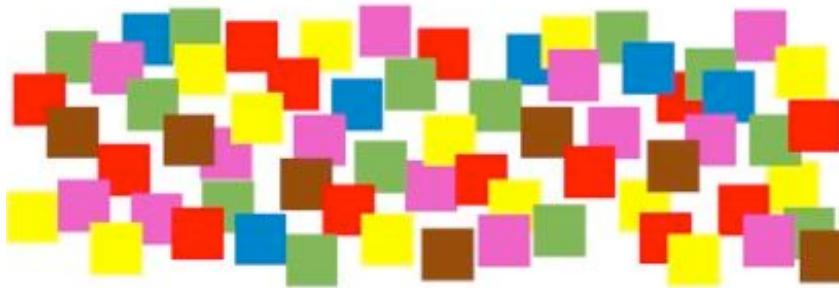
*90% of the data in the world today has been created in the last two years alone, at 2.5 quintillion bytes of data a day*

*report from IBM Marketing Cloud in 2017*

# Data



**Structured Data**

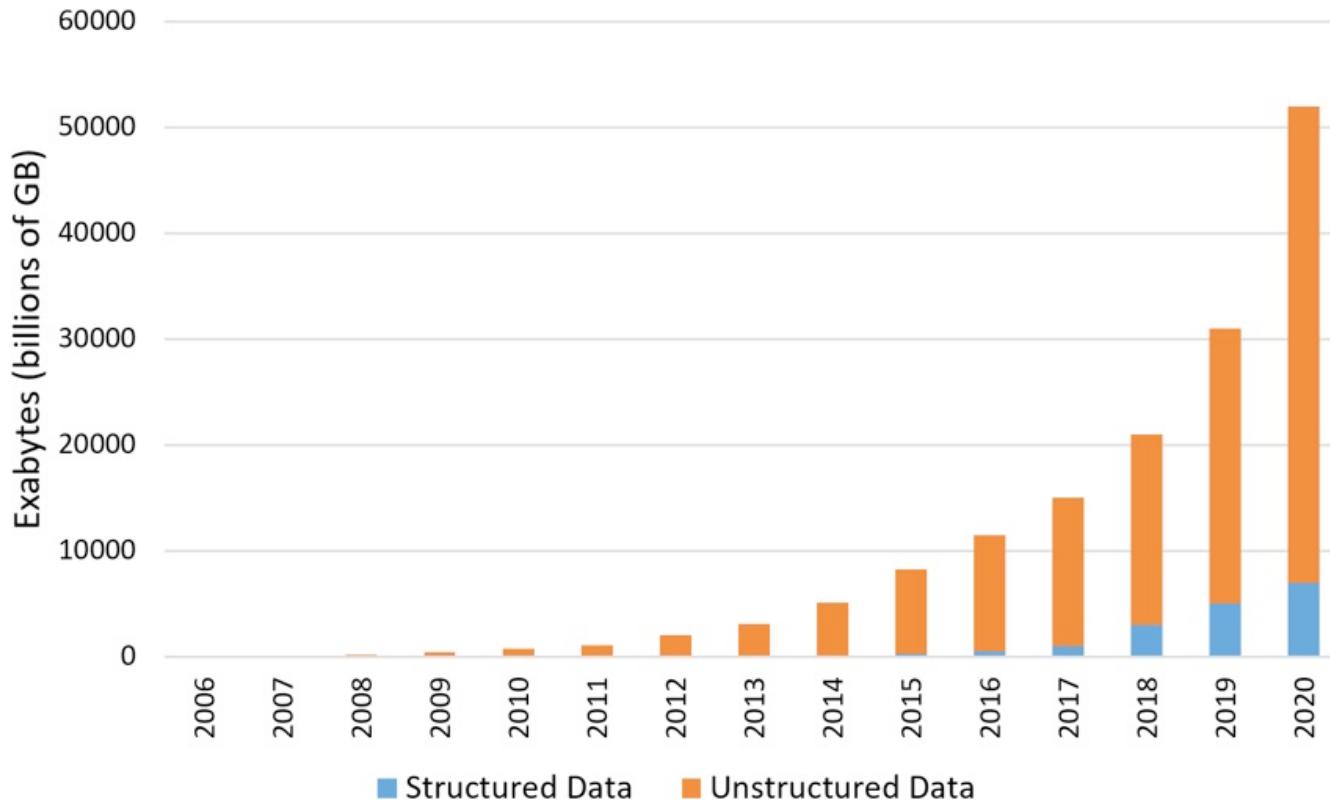


**Unstructured Data**

Graphical representations illustrate the difference between  
structured and unstructured data

# Data

## The Cambrian Explosion...of Data



# Data

Multiplying Factor	SI Prefix	Scientific Notation	Name
1 000 000 000 000 000 000 000 000	Yotta (Y)	$10^{24}$	1 septillion
1 000 000 000 000 000 000 000	Zetta (Z)	$10^{21}$	1 sextillion
1 000 000 000 000 000 000	Exa (E)	$10^{18}$	1 quintillion
1 000 000 000 000 000	Peta (P)	$10^{15}$	1 quadrillion
1 000 000 000 000	Tera (T)	$10^{12}$	1 trillion
1 000 000 000	Giga (G)	$10^9$	1 billion
1 000 000	Mega (M)	$10^6$	1 million
1 000	kilo (k)	$10^3$	1 thousand
0 001	milli (m)	$10^{-3}$	1 thousandth
0 000 001	micro (u)	$10^{-6}$	1 millionth
0 000 000 001	nano (n)	$10^{-9}$	1 billionth
0 000 000 000 001	pico (p)	$10^{-12}$	1 trillionth
0 000 000 000 000 001	femto (f)	$10^{-15}$	1 quadrillionth
0 000 000 000 000 000 001	atto (a)	$10^{-18}$	1 quintillionth
0 000 000 000 000 000 000 001	zepto (z)	$10^{-21}$	1 sextillionth
0 000 000 000 000 000 000 000 001	yocto (y)	$10^{-24}$	1 septillionth

Metric prefixes defined at the 19th General Conference on Weights and Measures in 1991

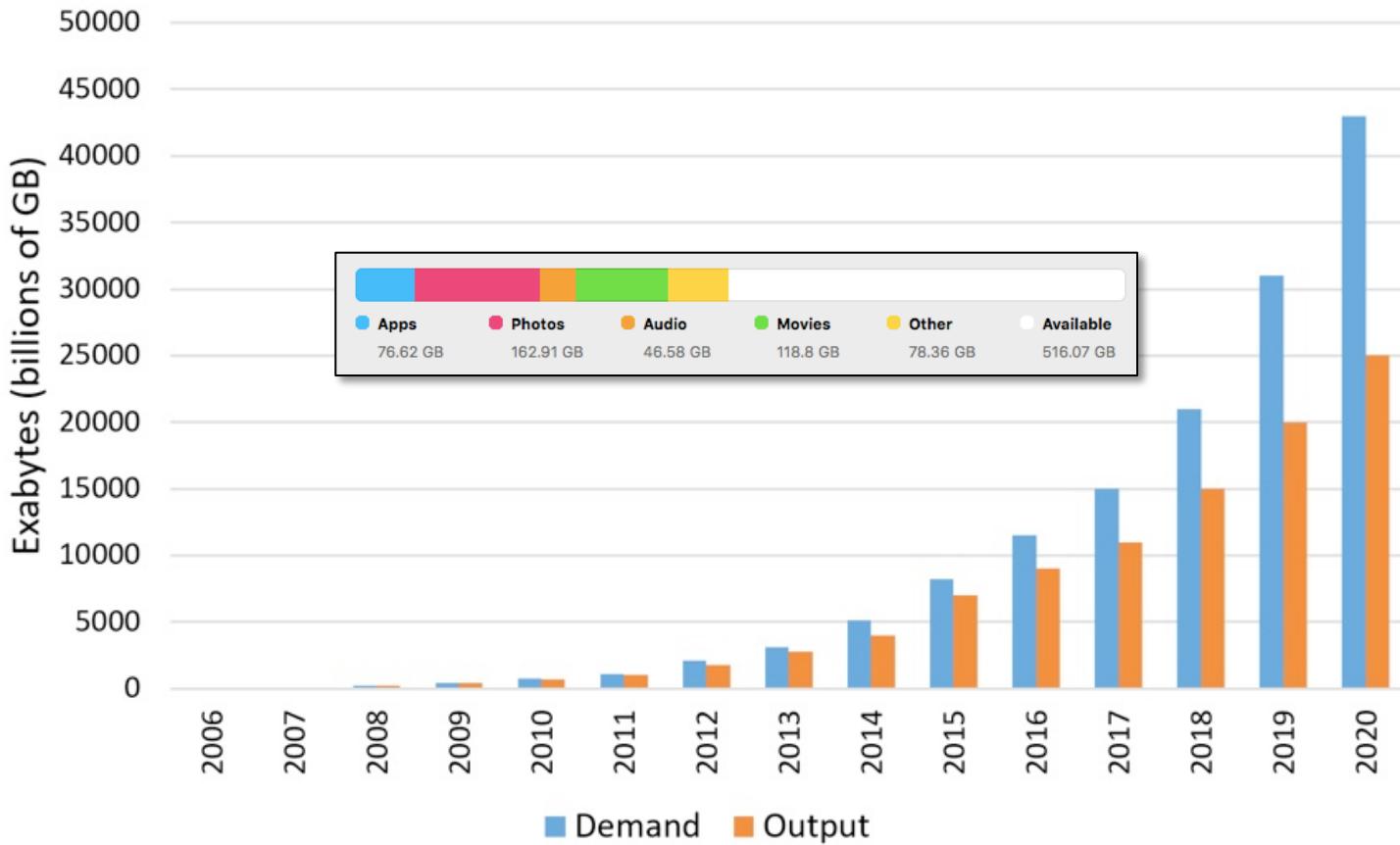
# Data

Multiplying Factor	SI Prefix	Scientific Notation	Name
1 208 925 819 614 629 174 706 176	Yottabytes	$2^{80}$	1 septillion
1 180 591 620 717 411 303 424	Zettabytes	$2^{70}$	1 sextillion
1 152 921 504 606 846 976	Exabytes	$2^{60}$	1 quintillion
1 125 899 906 842 624	Petabytes	$2^{50}$	1 quadrillion
1 099 511 627 776	Terabytes	$2^{40}$	1 trillion
1 073 741 824	Gigabytes	$2^{30}$	1 billion
1 048 576	Megabytes	$2^{20}$	1 million
1 024	kilobytes	$2^{10}$	1 thousand

Examples of prefixes used to measure digital data with a binary system

# Data

## Storage Supply & Demand



Storage supply and demand growth over two decades

# 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

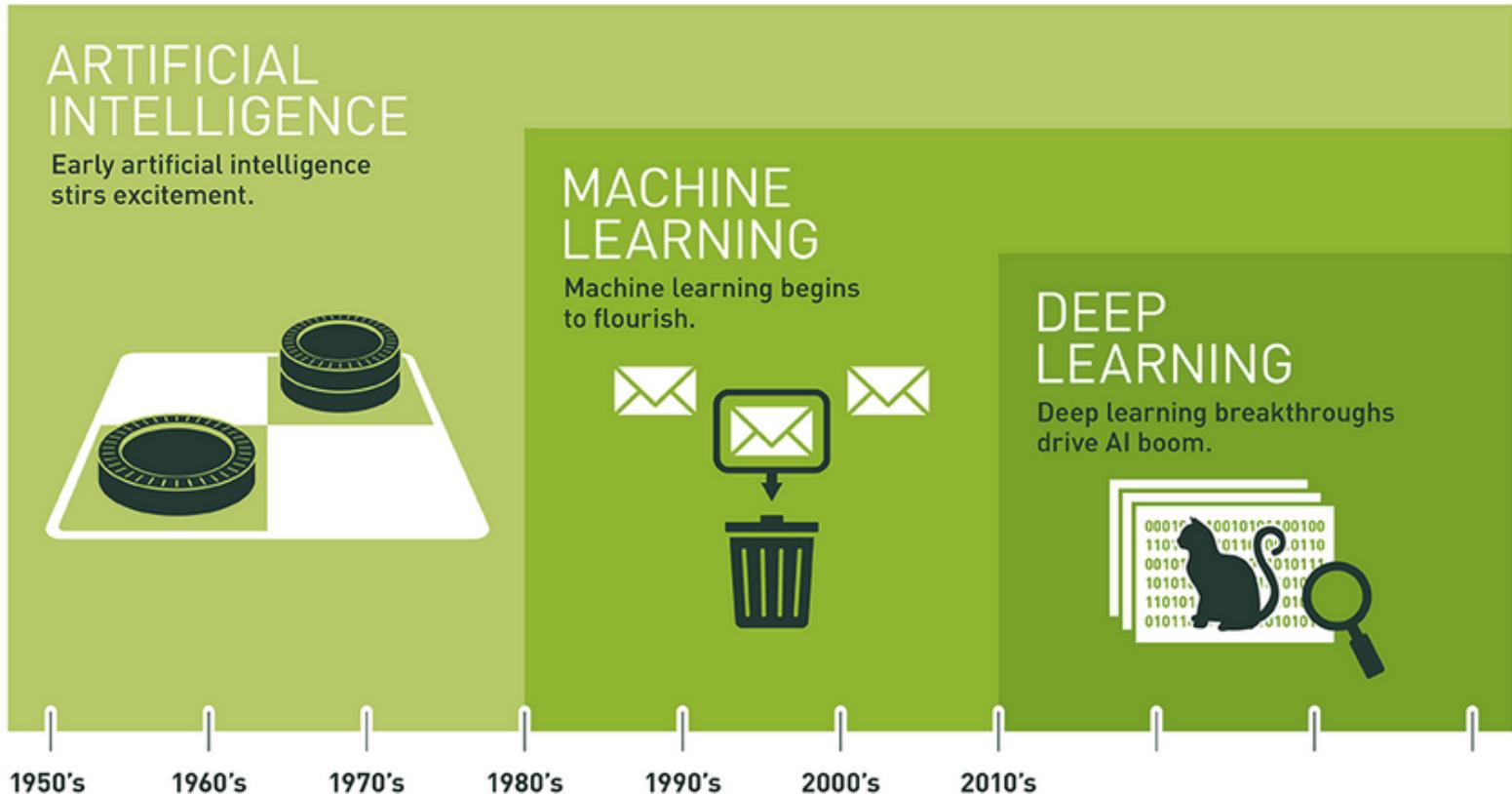
- Data Mining and Machine Learning concepts
  - Machine learning does not entail the conceptual limitations mentioned herein, and disregards any particular philosophical stance about what learning actually is
  - Machine learning is therefore a practical process connected to Data mining while using algorithms 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

- Python environment for Deep Learning – why using it?
  - Because you can develop **proof-of-concept** solutions fairly easily and free
  - **Open-source** software that is widely spread
  - **Fast** development time
  - **Many scientific libraries**: NumPy, SciPy, Matplotlib, Theano, Scikit, Tensorflow, etc.
  - Mobile computing for **smartphone development** by using Kivy for Android and iOS
  - **Web-site** incorporation by using Django
  - Try it here: <https://www.pythonanywhere.com/try-ipython/>

# Machine Learning With TensorFlow

- Python – pros and cons:

## Pros

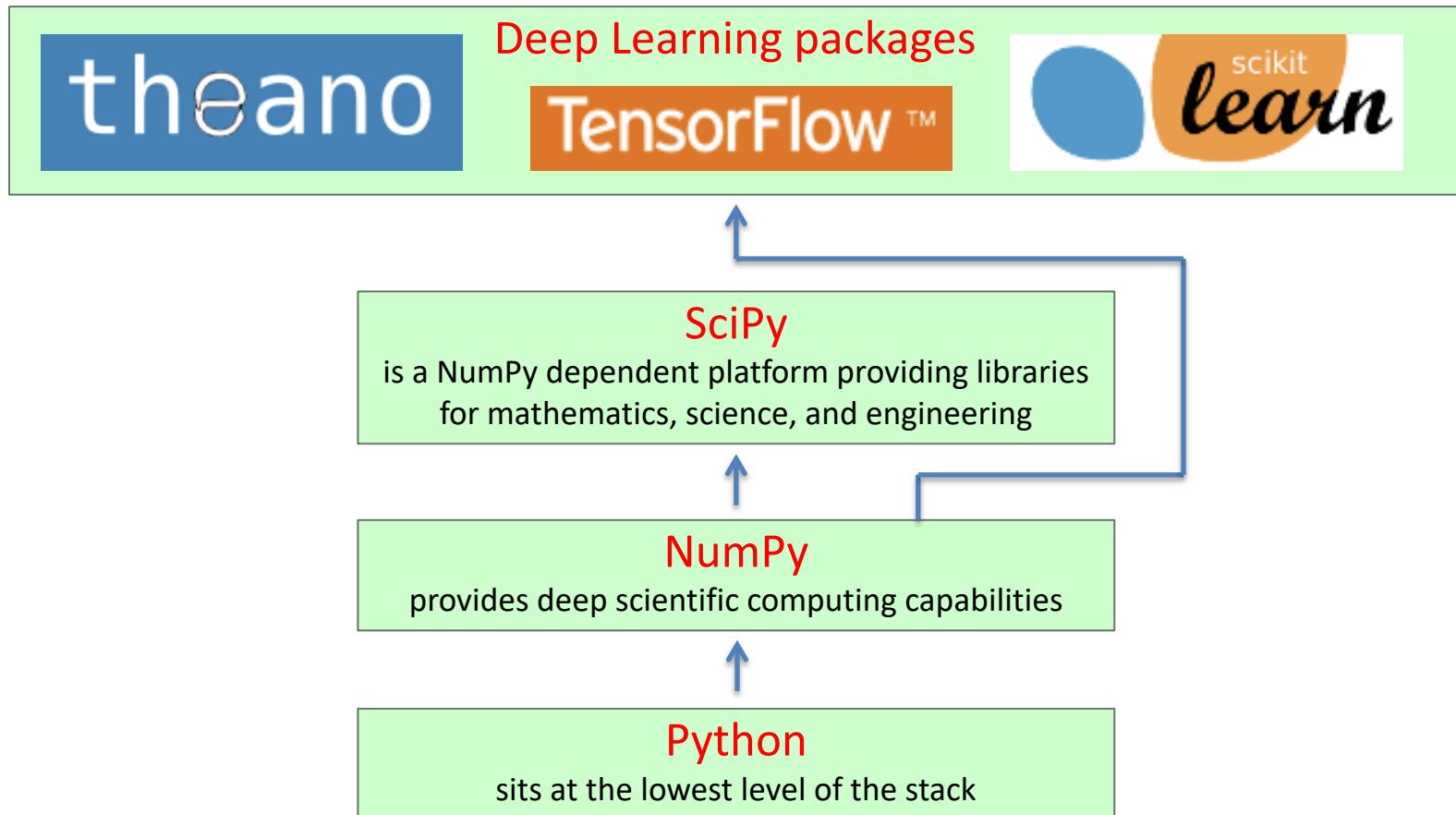
- High-level object-oriented language that is **easy to learn**
- **Good readability** of the code
- **Quick development time**
- Rich scientific libraries and many others like web server management, serial port access, etc.
- **Easy to share** and communicate results
- **Easy to read**, manipulate and process data
- A single environment for many tasks. No need to learn another language
- **Free and open-source** software that is widely spread

## Cons

- **Speed of execution** since it is higher level language
- **Mobile computing** for smartphone development (unless you use Kivy: Linux, Windows, OS X, Android and iOS)
- It is **not included in Web browsers**, because it is hard to secure
- Design flaws because it is **dynamically typed language** and some errors show at runtime such as syntactic errors (mistyping variable names)
- Python can be both **interpreted (.py)** and **compiled (.pyc)**
- **Documentation** isn't at the level of PHP or Java

# Machine Learning With TensorFlow

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



# Machine Learning With TensorFlow

- Python + NumPy + SciPy + ?
  - ... is like Matlab + toolboxes + DL processing



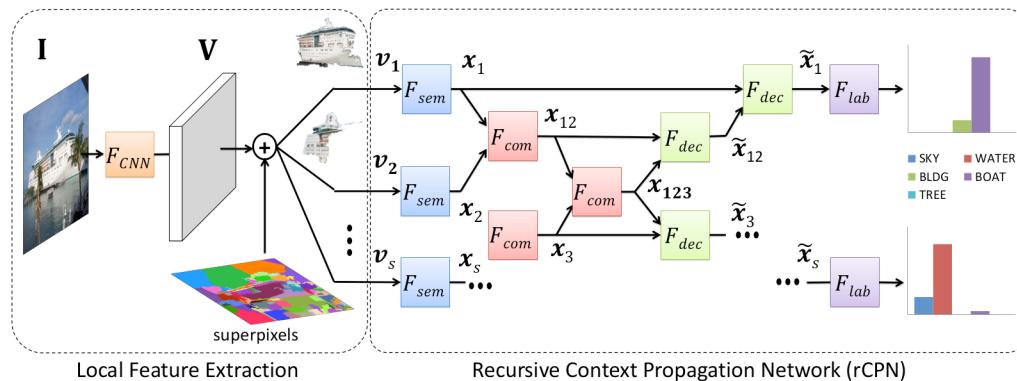
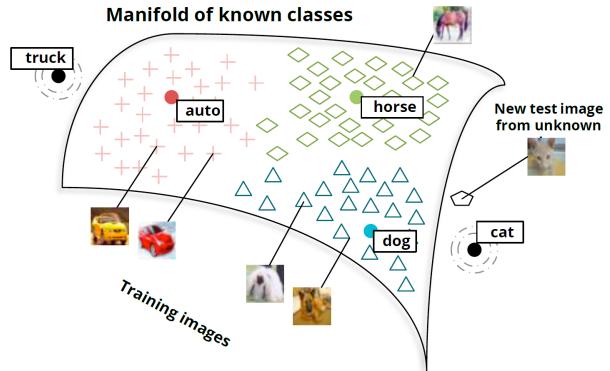
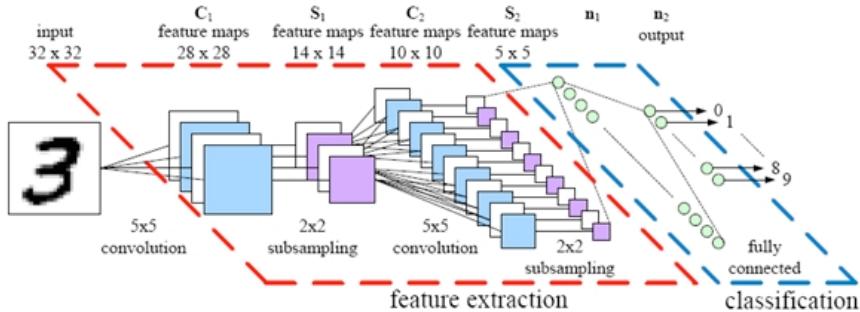
# Machine Learning With TensorFlow

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



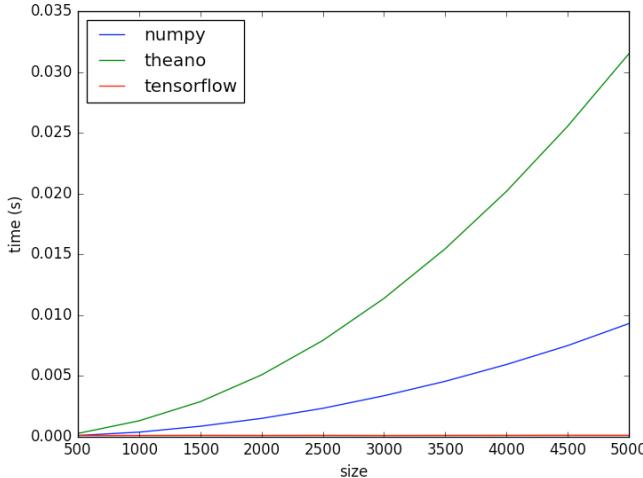
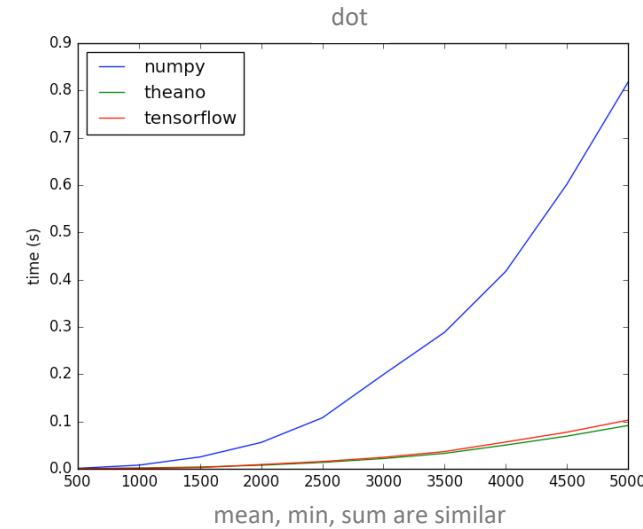
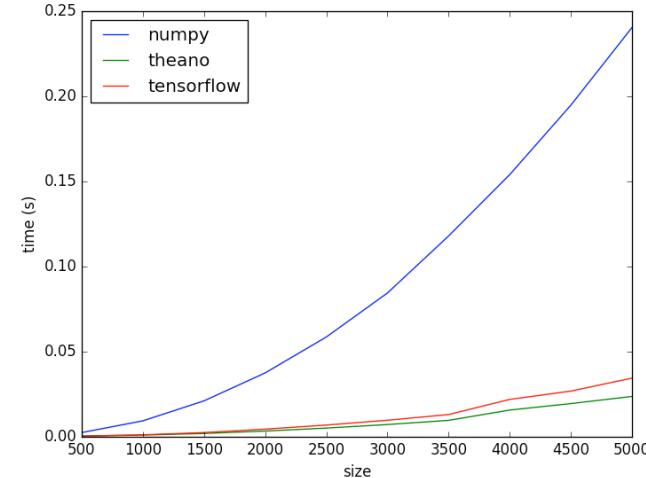
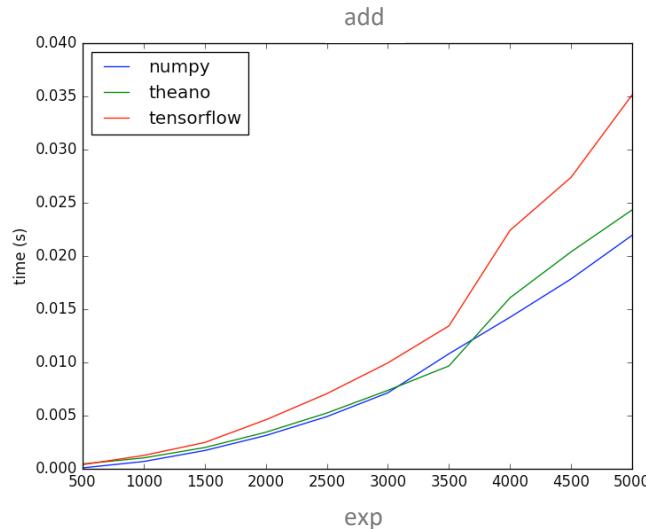
# Machine Learning With TensorFlow

- 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 With TensorFlow

## Top Deep Learning Projects

A list of popular github projects related to deep learning (ranked by stars).



Project Name	Stars	Description
TensorFlow	29622	Computation using data flow graphs for scalable machine learning.
Caffe	11799	Caffe: a fast open framework for deep learning. CNN & image processing, written in C++
Theano	4286	Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. It can use GPUs and perform efficient symbolic differentiation.
Leaf	4281	Open Machine Intelligence Framework for Hackers.
Char RNN	3820	Multi-layer Recurrent Neural Networks (LSTM, GRU, RNN) for character-level language models in Torch.
Neural Talk	3694	NeuralTalk is a Python+numpy project for learning Multimodal Recurrent Neural Networks that describe images with sentences.
deeplearning4j	3673	Deep Learning for Java, Scala & Clojure on Hadoop, Spark.
TFLearn	3368	Deep learning library featuring a higher-level API for TensorFlow.
TensorFlow Playground	3352	Play with neural networks!
Scikit Neural Net	849	Deep neural networks without the learning cliff! Classifiers and regressors compatible with scikit-learn.

Source: <https://github.com/aymericdamien/TopDeepLearning>

# Machine Learning With TensorFlow

## Top Deep Learning Projects

According to medium.com

View 1+ more



TensorFlow



Caffe



Microsoft  
Cognitive  
Toolkit



PyTorch



Apache  
MXNet



Chainer



Keras

### 8 Best Deep Learning Frameworks for Data Science enthusiasts

- 1. TensorFlow. TensorFlow is arguably one of the **best deep learning** frameworks and has been adopted by several giants such as Airbus, Twitter, IBM, and others mainly due to its highly flexible system architecture. ...
- 2. Caffe. ...
- 3. Microsoft Cognitive Toolkit/CNTK. ...
- 4. Torch/PyTorch. ...
- 5. MXNet. ...
- 6. Chainer. ...
- 7. Keras. ...
- 8. Deeplearning4j.

Apr 5, 2018

Source: <https://medium.com/the-mission/8-best-deep-learning-frameworks-for-data-science-enthusiasts-d72714157761>

# 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 - evaluating **constant** expressions at compile time (not runtime)
  - merging of similar sub-graphs, 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 decrease calculations
  - using in place 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/optimizations.html#optimizations>

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

# 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 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** – Theano does your derivatives for function with one or many inputs
  - **speed and stability optimizations** – Get the right answer for  $\log(1+x)$  even when  $x$  is really tiny
  - **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

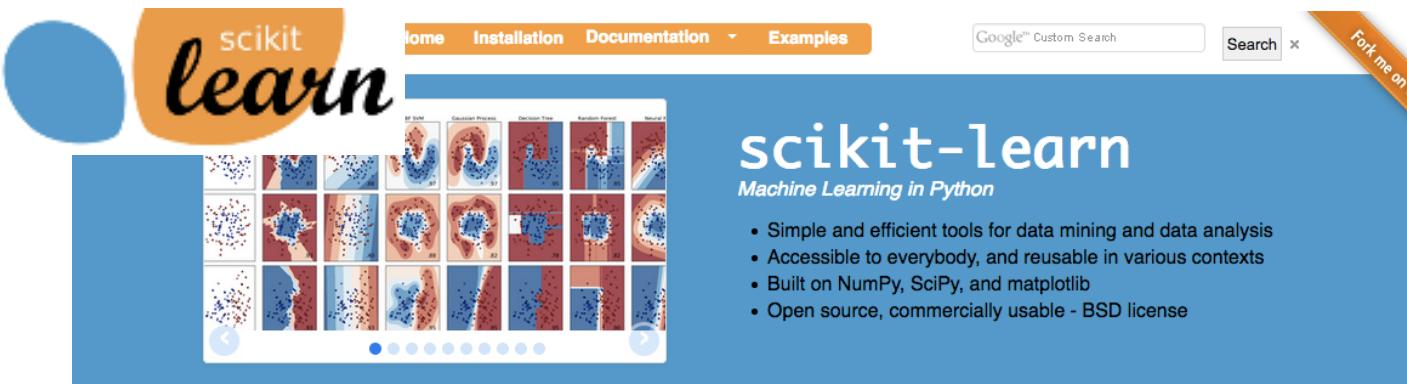
- Theano and TensorFlow are competing heavily for which one is the best platform
- Theano outperforms TensorFlow on a **single GPU**, but TensorFlow outperforms Theano for **parallel execution** on multiple GPUs
- Theano has better documentation in general than TensorFlow, although that changes quickly
- 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
  - **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.

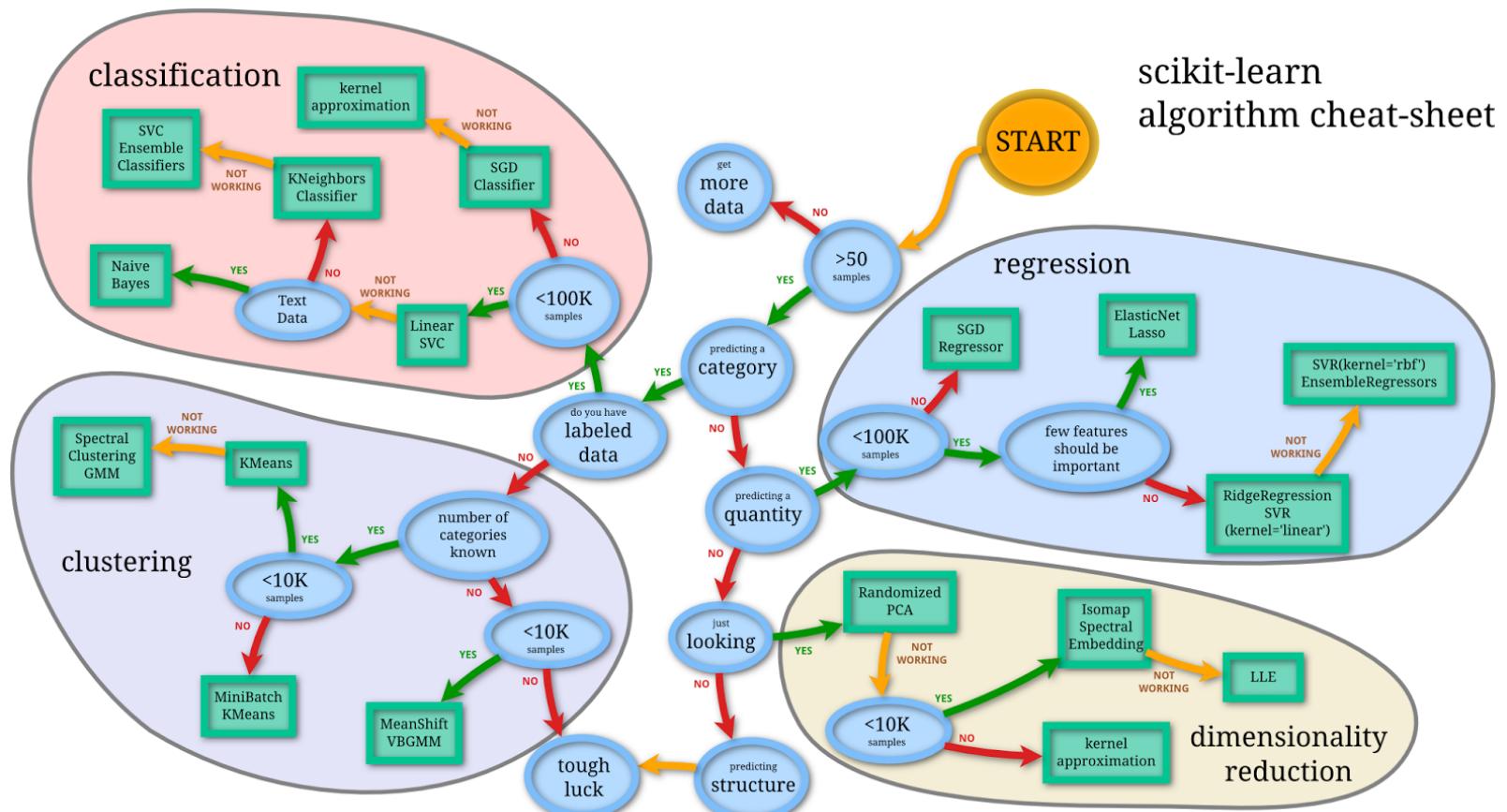
**Modules:** preprocessing, feature extraction.

— Examples

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

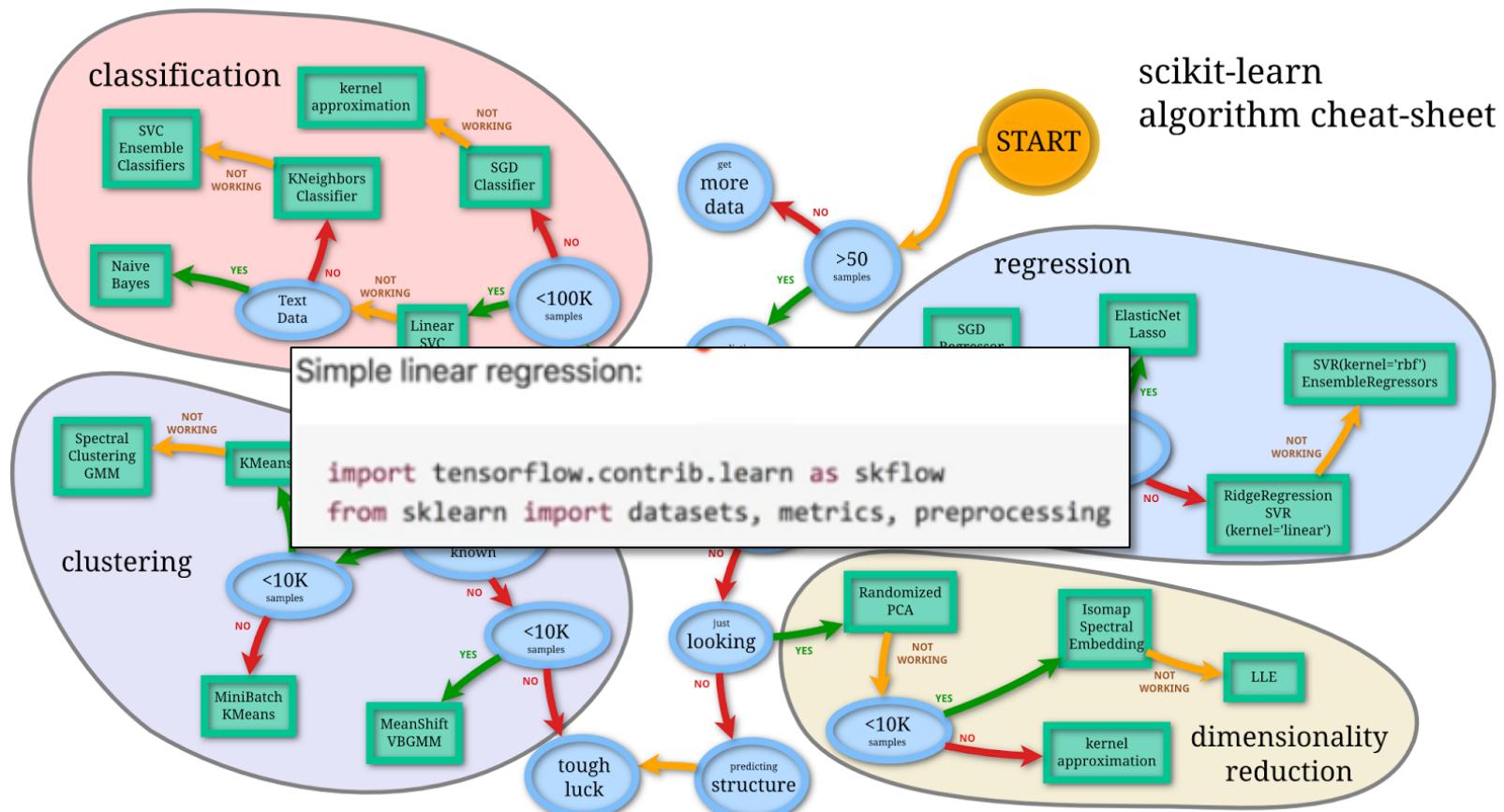
# Machine Learning

- Working with Theano, Scikit-Learn and TensorFlow



# Machine Learning

- Working with Theano, Scikit-Learn and TensorFlow



# 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™ 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'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**

# Why TensorFlow?

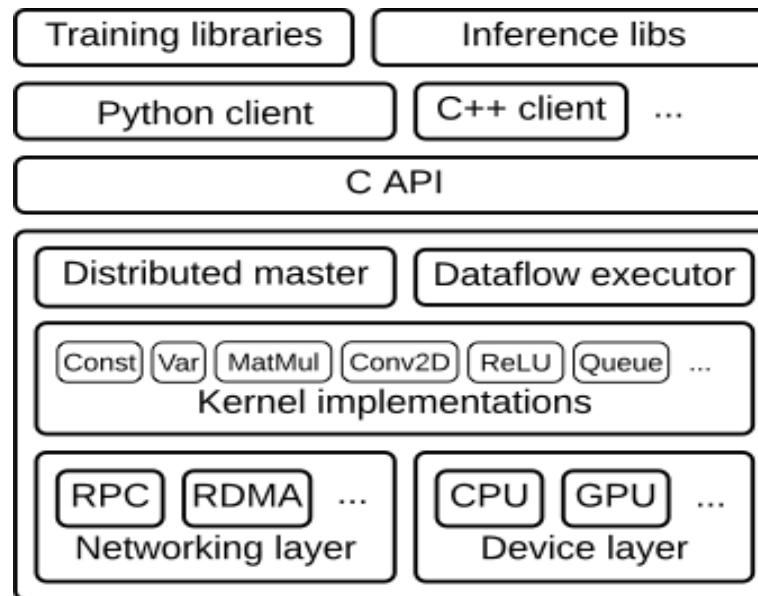
- TensorFlow provides **excellent functionalities and services** when compared to other popular deep learning frameworks.
- These **high-level operations** are essential for carrying out **complex parallel computations** and for building **advanced neural network** models.
- TensorFlow is a **low-level library** which provides **more flexibility**.
- Thus you can define your own functionalities or services for your models

# Why TensorFlow?

- Distribute the training time of a neural network model over many servers to **reduce the training time**.
- TensorFlow's API is a complete package which is easier to use and read, plus provides **helpful operators**, **debugging** and **monitoring tools**, and **deployment** features.
- TensorFlow can run on multiple CPUs and GPUs.
- TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

# TensorFlow Architecture

- The TensorFlow runtime is a **cross-platform library**. The figure below illustrates its general architecture.



# 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+

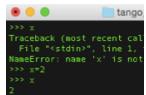


- Distributions: Python 2.7, 3.4, 3.5
- Dependencies: Numpy, bazel, six, wheel

# 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/>

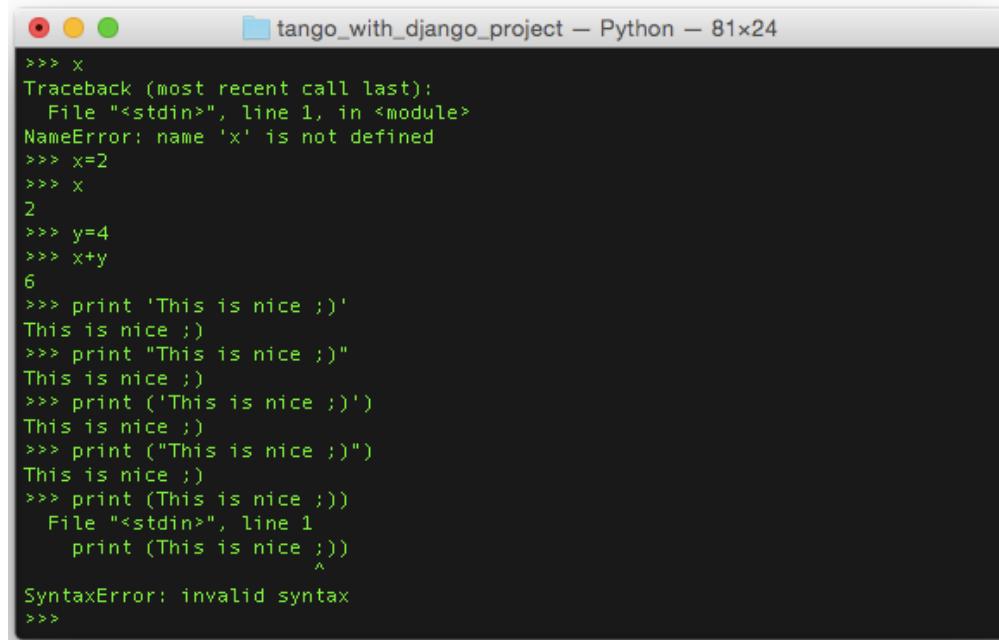


# Machine Learning With TensorFlow

- 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
```

use the latest version

pip3 install --upgrade  
https://storage.googleapis.com/tensorflow/mac/cpu/tensorflow-1.15.0-py3-none-any.whl

# 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
```

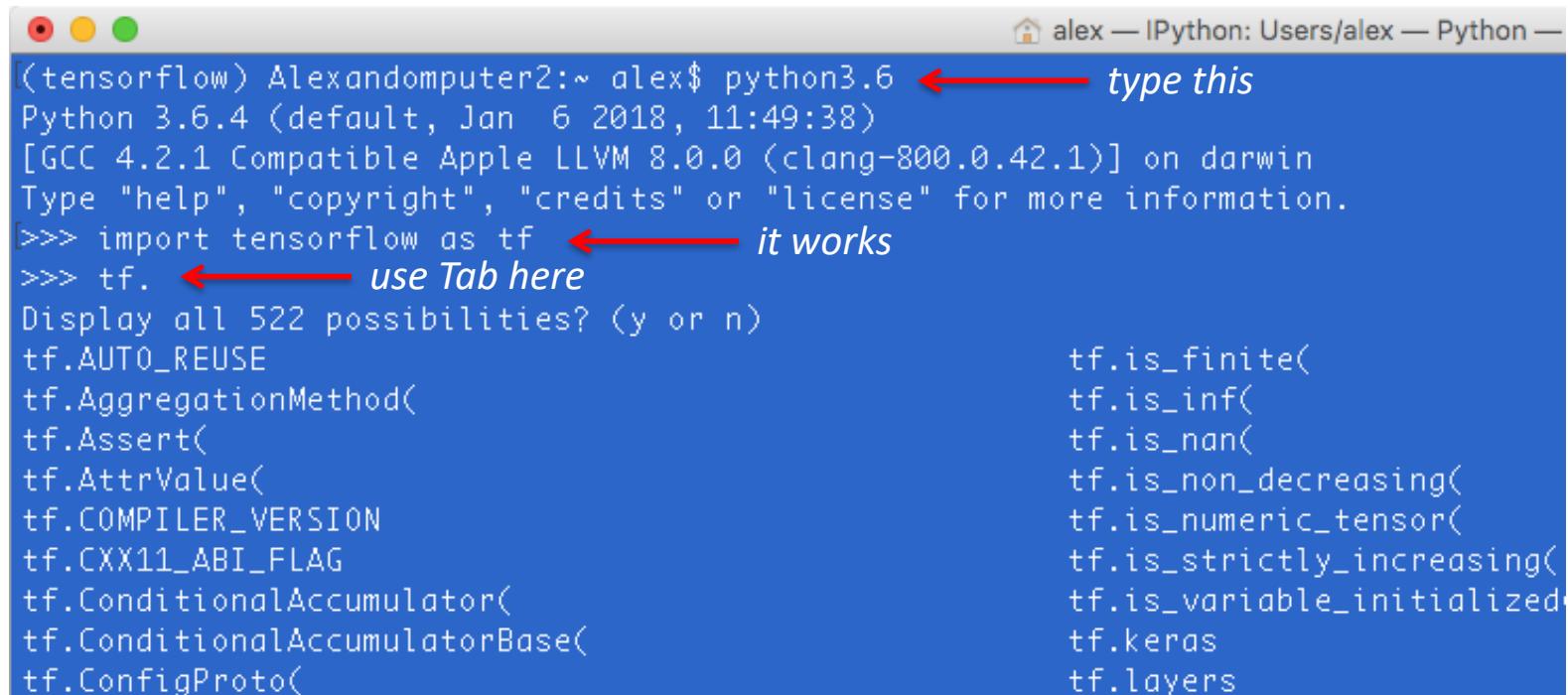
- Since they run from different locations, iPython may not have the path to run TensorFlow so it is best to reinstall iPython like this in **virtualenv**:

```
(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 [Python3.6/3.7](#):



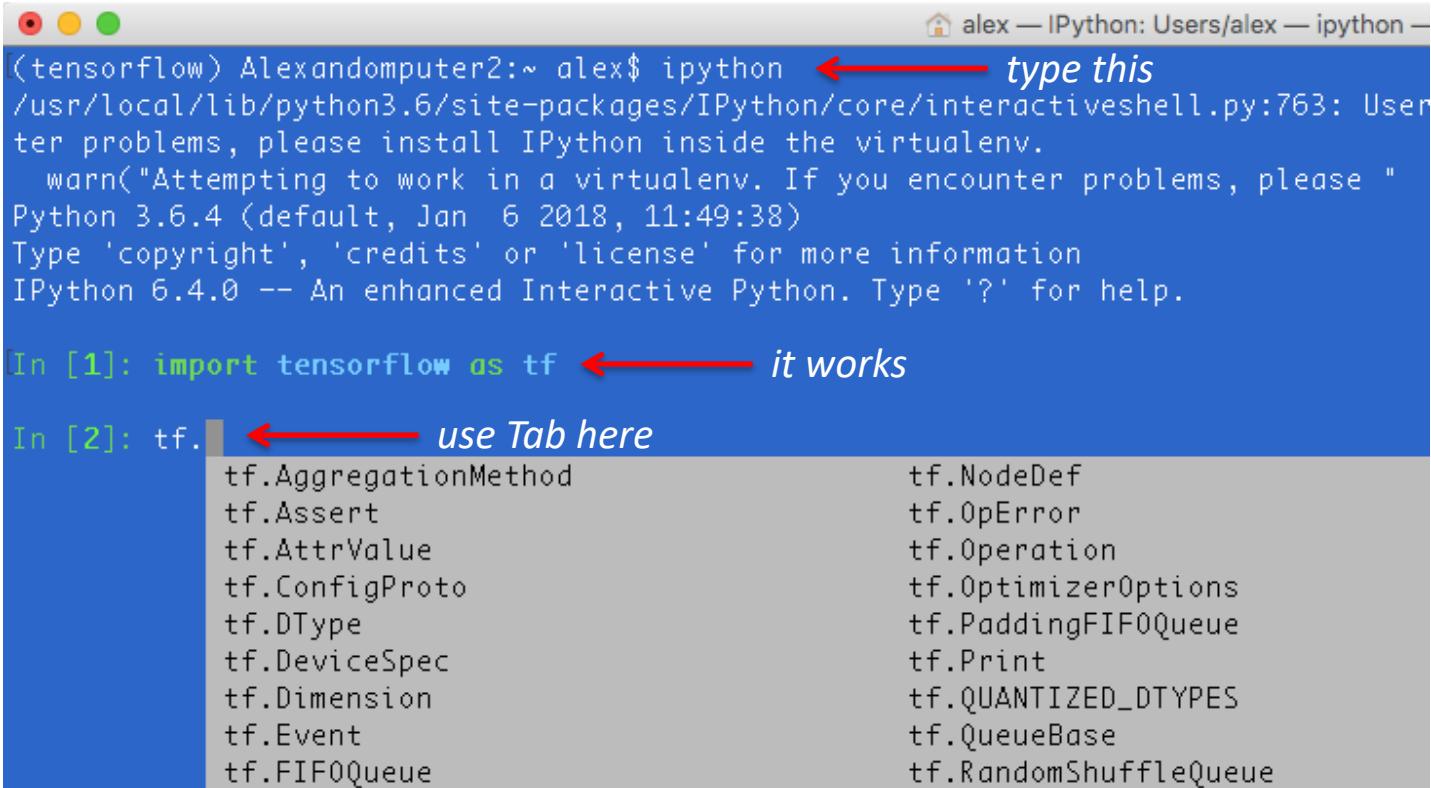
A screenshot of a terminal window titled "alex — IPython: Users/alex — Python —". The terminal shows the following session:

```
(tensorflow) Alexandromputer2:~ alex$ python3.6 ← type this
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 ← it works
>>> 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(
```

The terminal shows the user typing "python3.6" and then importing "tensorflow" as "tf". Red arrows point to the command "python3.6" with the annotation "type this", to the import statement "import tensorflow as tf" with the annotation "it works", and to the start of the "tf." prefix with the annotation "use Tab here". To the right of the "tf." prompt, a list of available methods is shown, starting with "tf.AUTO\_REUSE" and ending with "tf.layers".

# Machine Learning With TensorFlow

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



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

```
(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 install IPython inside the virtualenv.
warn("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
```

The terminal shows the user attempting to run iPython in a virtual environment. It then demonstrates importing TensorFlow and using tab completion for module names.

# 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

# JUPYTER NOTEBOOK:

- Non Profit, **open source** project.
- **Supports** Data Science and **Scientific Computing** across all programming languages
- As all **open software**, Jupyter is developed by the Jupyter community (GitHub)
- **Python** language could easily be **availed to write a machine learning code on Jupyter** since it already comes with a pre-configured data science environment as it runs on the cloud or on our own hardware.

# JUPYTER HUB:

- Without asking the users for installation or any other tasks, serves access to users for computational environment and resources.
- Everyone (students, researchers and scientists alike) get their own workspace which can be easily modified by the system administrators.

## -----Installation-----

- Can be installed using CONDA or PiP.
  - `conda install -c conda-forge jupyterlab #CONDA`
  - `pip install jupyterlab #PiP`

# 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:

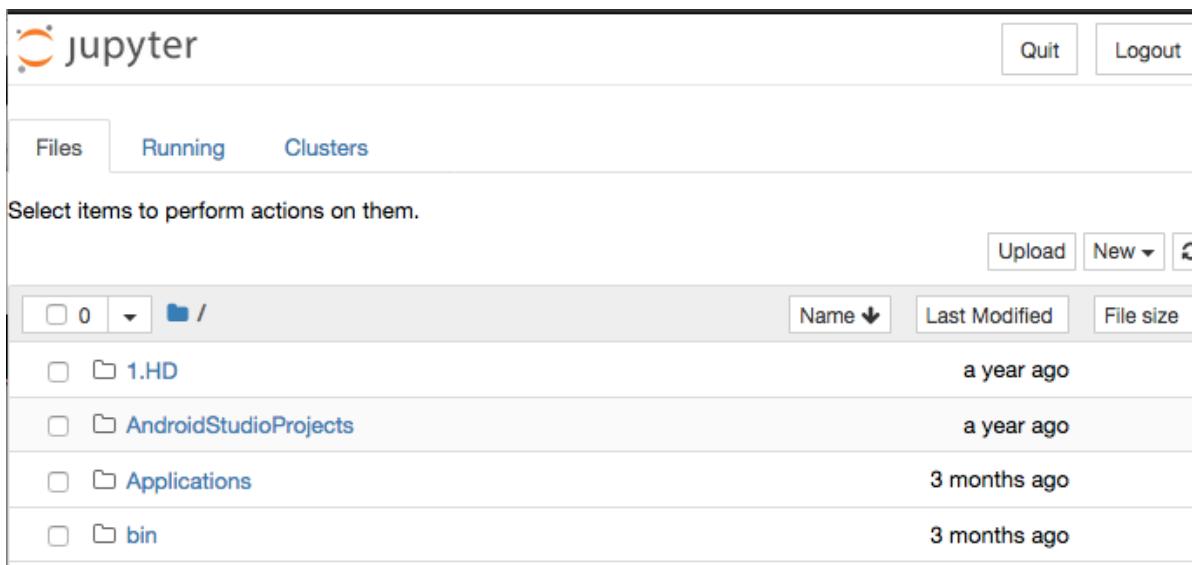
```
# Python 2.7
$ sudo apt-get build-dep python-matplotlib python-tk
# Python 3
$ sudo apt-get build-dep python3-matplotlib python3-tk
```

- 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>

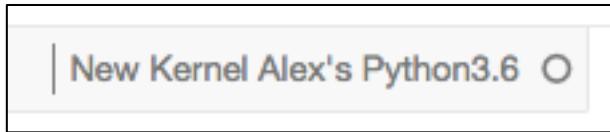
# Machine Learning With TensorFlow

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

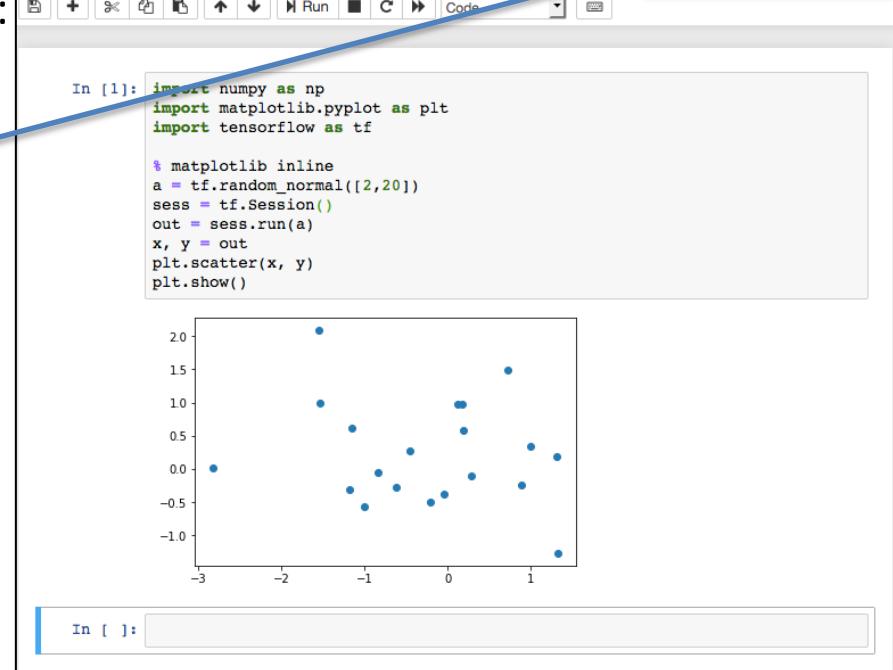
```
alex$ jupyter notebook --NotebookApp.token=qwerty
```



- Then run any file you might have:



... 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
```

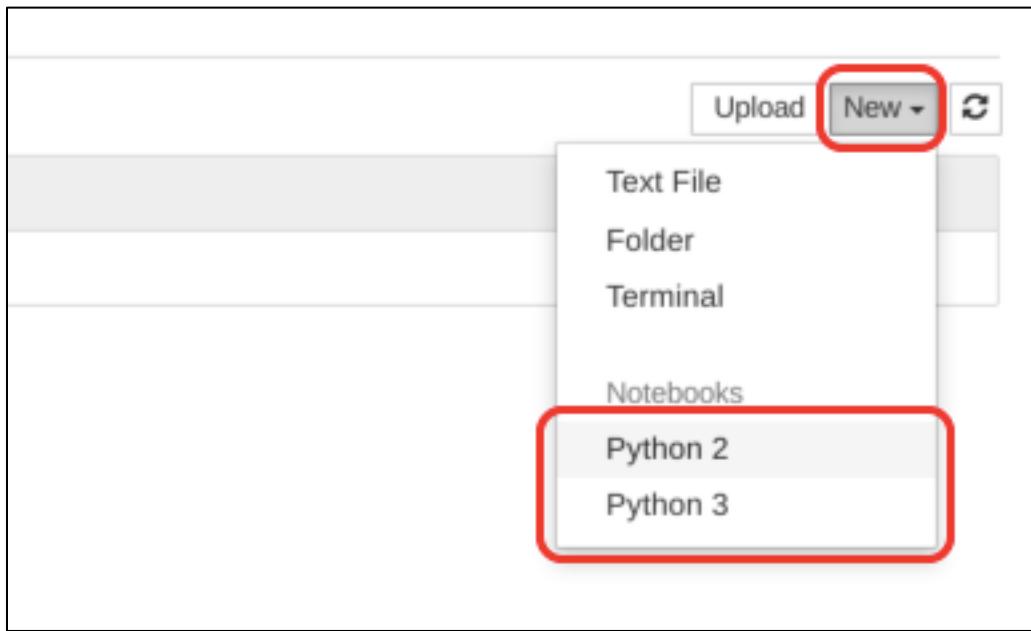


```
alex — nano ~/Library/Jupyter/kernels/python3.6/kernel.json
GNU nano 2.0.6 File: .../python3.6/kernel.json

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

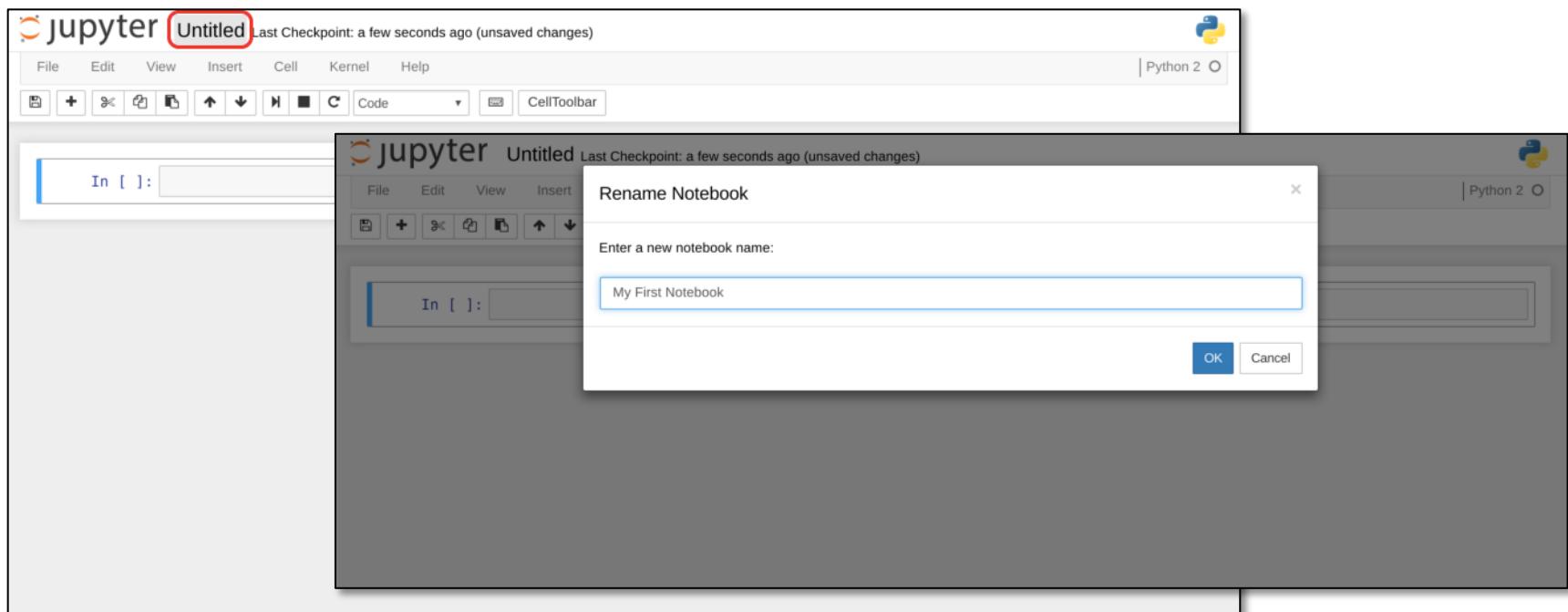
# Machine Learning With TensorFlow

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



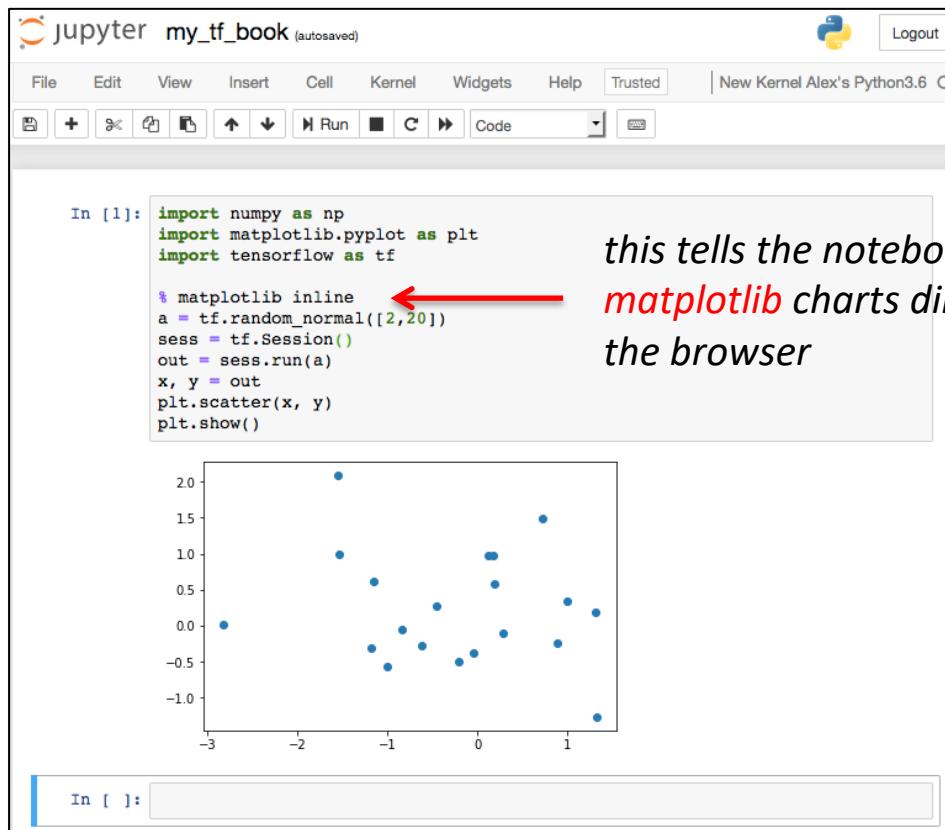
# Machine Learning With TensorFlow

- 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:

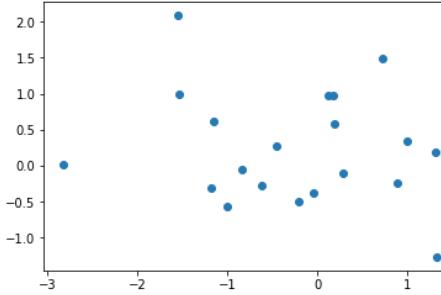


The screenshot shows a Jupyter Notebook interface with the title "jupyter my\_tf\_book (autosaved)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Logout. A toolbar below the menu has icons for file operations and cell execution. The notebook content area shows a code cell labeled "In [1]:" containing Python code to generate a scatter plot. 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. The resulting scatter plot is displayed below the 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()
```

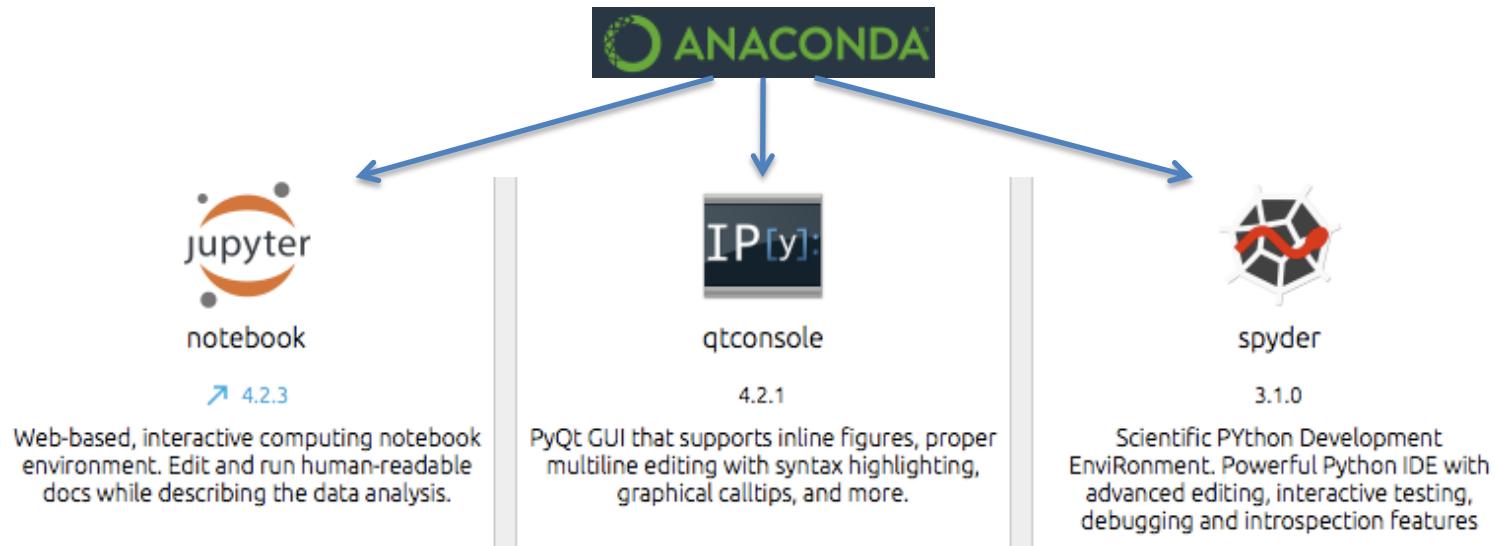
*this tells the notebook to display matplotlib charts directly inside the browser*



x	y
-2.8	0.0
-1.8	2.1
-1.5	1.0
-1.2	0.7
-1.0	-0.3
-0.8	-0.5
-0.5	0.2
-0.3	0.4
0.0	-0.4
0.2	0.9
0.5	0.6
0.8	1.5
1.0	0.4
1.2	-1.0
1.5	0.2
1.8	-0.2
2.0	0.0

# 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. It features a navigation bar at the top with tabs for 'For Individuals', 'For Enterprises', 'For Academics', and a link to download for subscribers. Below this is a large promotional banner for 'CANOPY EXPRESS' which is listed as 'FREE'. To the right are three other editions: 'CANOPY WITH PYTHON ESSENTIALS' (\$199/year), 'CANOPY WITH PYTHON FOUNDATIONS ALL ACCESS' (\$678/year), and 'CANOPY TRIPLE PLAY' (\$1,125/year). A yellow button in the center of the banner says 'Contact us for discounts for 5+ users'. Below the banner are four 'Add to Cart' buttons. A section titled 'Pre-built and tested Python packages' compares '100+ Core' packages in the Express edition against '300+ Extended' packages in the others. A link 'See included Python package details' with a plus sign icon is provided. At the bottom, a table compares features across the editions: 'Graphical Debugger' (available in Express), 'Integrated IPython' (available in Express, Python Essentials, Foundations All Access), 'Advanced Code Editor' (available in Express, Python Essentials, Foundations All Access), and 'Application Development Platform' (available in Express, Python Essentials, Foundations All Access).

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, a sidebar has tabs for '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, a detailed view for 'seaborn' is shown with sections for 'Installed' (Currently not installed), 'Available on store' (0.7.0-6, entought/free), and a large blue 'Install v0.7.0-6' button. It also notes 'No summary available.' and '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 Jupyter interface includes a toolbar at the top with file operations like File, Edit, View, Insert, Cell, Kernel, and Help, along with a CellToolbar button.

```
#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.')

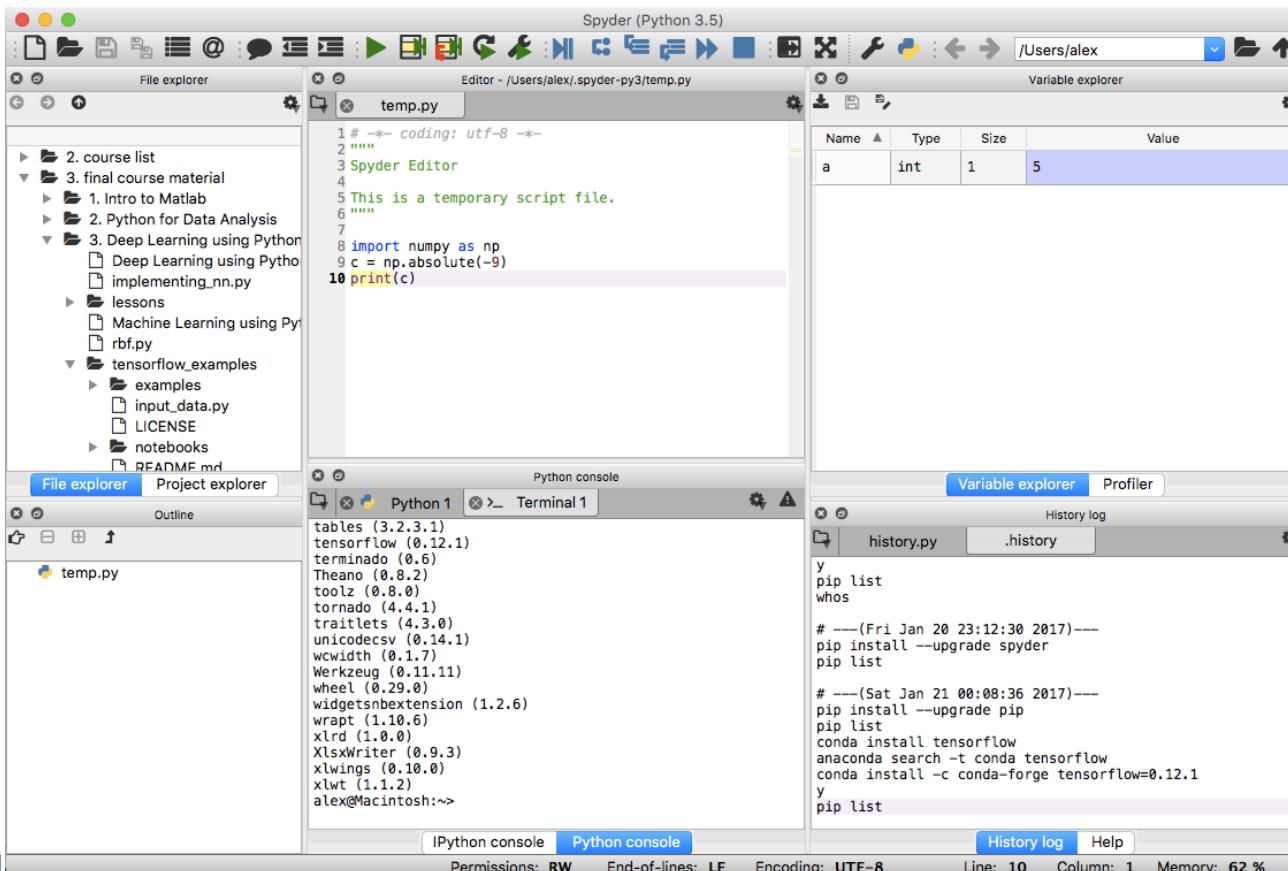
# Setting up better defaults for matplotlib
from matplotlib import rcParams

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

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

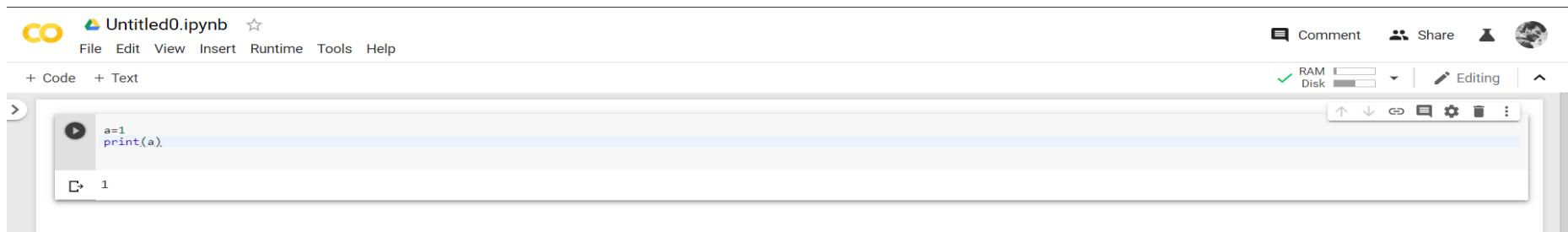
# Machine Learning With TensorFlow

- 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):



# GOOGLE COLAB:

- Previously we learnt about installing Jupyter in our hardware, Google Colab gives us access to Jupyter Notebook on our browser
- Runs entirely on the cloud. No need for installation!
- It's easy and efficient for systems which do not have a lot of memory space.
- We can easily write a python code on it and run the shell script via the play button on the left of the code.



The screenshot shows the Google Colab interface. At the top, there is a toolbar with a logo, file navigation, and various menu options like File, Edit, View, Insert, Runtime, Tools, and Help. Below the toolbar, there are two tabs: 'Code' and 'Text'. A code cell is active, containing the following Python code:

```
a=1  
print(a)
```

At the bottom of the code cell, there is a small icon with a play button symbol followed by the number '1', indicating the current cell index. On the right side of the interface, there are additional controls for commenting, sharing, and runtime settings, including sliders for RAM and Disk usage.

# 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 interface. The top bar shows the title "Anaconda Navigator - Beta". On the right, there is a search bar with the text "sciki" and a magnifying glass icon. Below the search bar is a table of installed packages. The table has columns for Name, Description, and Version. Two packages are listed: "scikit-image" (version 0.12.3) and "scikit-learn" (version 0.18.1). Arrows point from the search bar to the search icon and from the search bar to the "scikit" entry in the table.

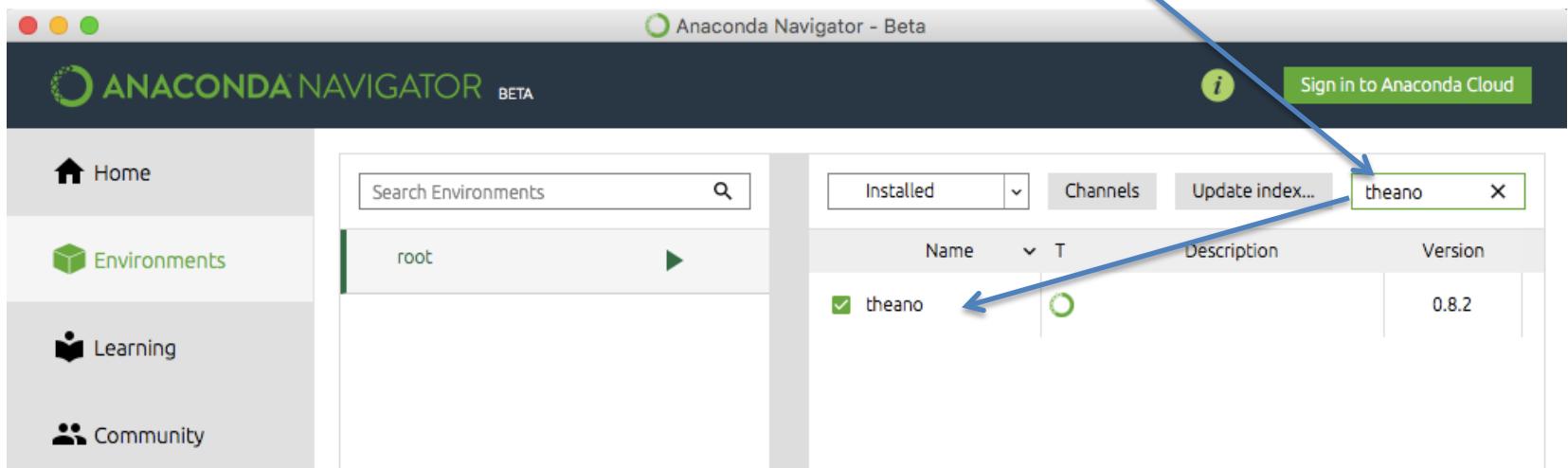
Name	Description	Version
scikit-image	Image processing routines for scipy	0.12.3
scikit-learn	Set of python modules for machine learning and data mining	0.18.1

# 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:

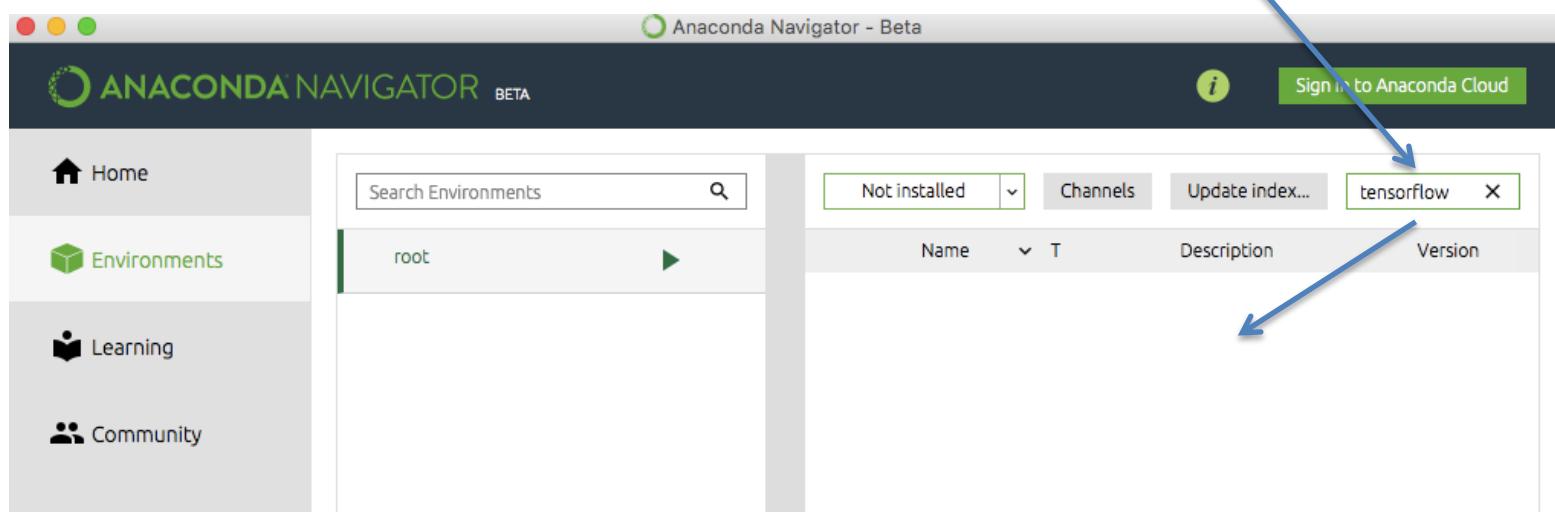


# Machine Learning With TensorFlow

- TensorFlow in Anaconda:



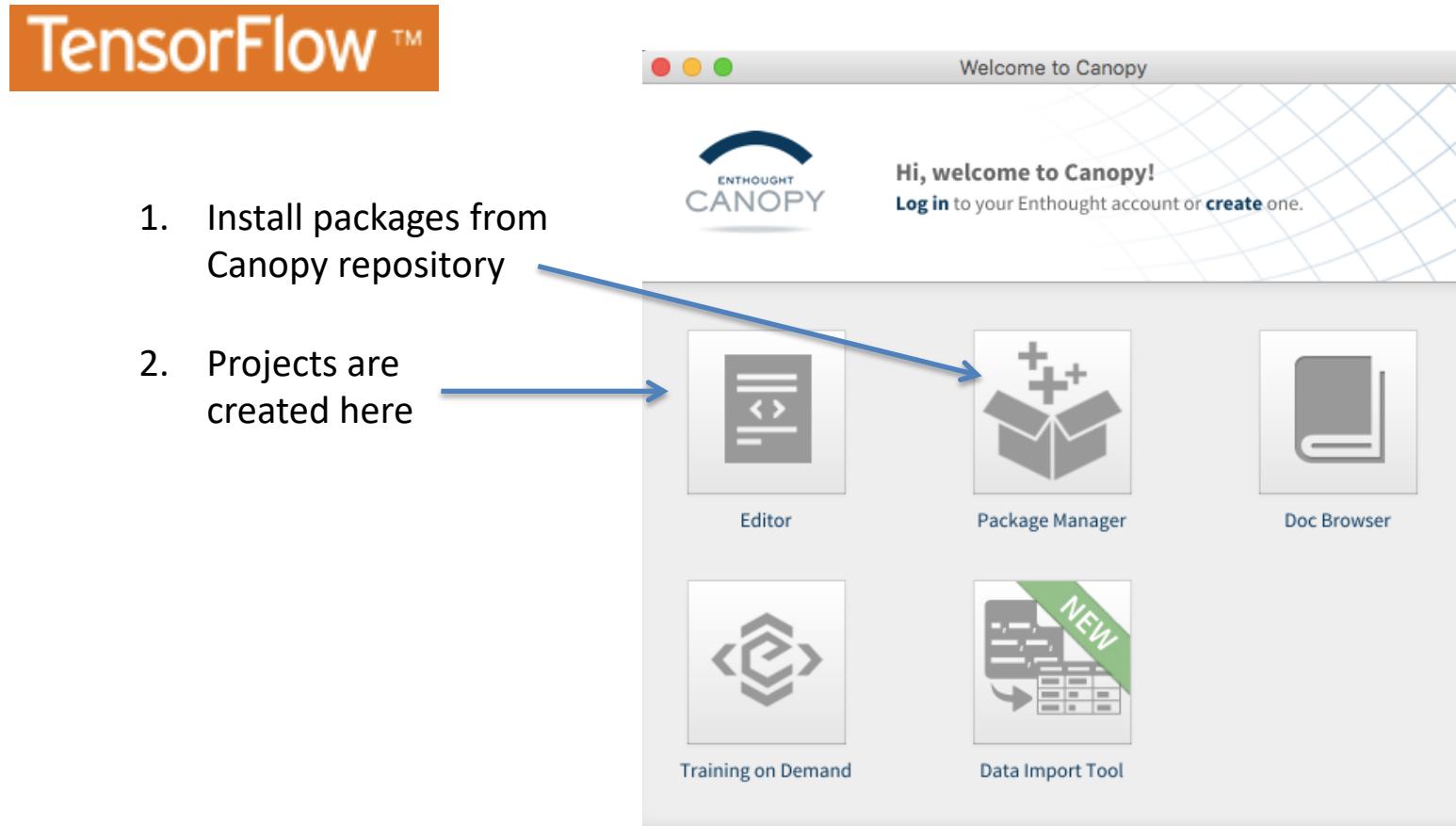
- TensorFlow is not part of the standard Anaconda repository:



- So you can install it via Spyder's terminal by typing:  
`>>> conda install -c conda-forge tensorflow=1.15.0`

# Machine Learning With TensorFlow

- TensorFlow in Canopy:



# Machine Learning With TensorFlow

- TensorFlow in Canopy:

TensorFlow™

The screenshot shows the 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 to the 'tensorflow' row in the main table. The 'tensorflow' row is highlighted with a yellow background and circled in red. The table has columns for 'Package Name' and 'Installed Version'. The 'tensorflow' row shows 'tensorflow' and '1.3.0-2'. Below the table, a summary for 'tensorflow' states 'Up-to-date.' with a green checkmark. It says 'No summary available.' and provides 'Installed:' (1.3.0-2) and 'Available on store:' (enthought/free) information. Buttons for 'Uninstall' and 'Re-install v1.3.0-2' are present.

	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	
<b>tensorflow</b>	<b>1.3.0-2</b>	
widgetsnbextension	2.0.0-1	

**tensorflow** Up-to-date.

No summary available.

Installed: 1.3.0-2

Available on store: (enthought/free)

# 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. On the left, the 'File Browser' window displays a list of recent files, many of which are Python scripts related to TensorFlow examples. A red box highlights this window. In the center, the 'Editor - Canopy' window shows the content of 'helloworld.py'. A blue arrow points from the 'Recent Files' list towards this editor window. The code in 'helloworld.py' is as follows:

```
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))
```

To the right of the editor is the 'Python' interactive console window. A red box highlights this window. It displays the welcome message and help information for IPython 4.1.2. A blue arrow points from the 'Recent Files' list towards the console window. The console history shows:

```
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]:
```

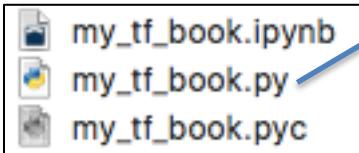
At the bottom, the status bar indicates 'Cursor pos 15 : 23' and 'Python'.

# 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'. In the center is an 'Editor - Canopy' window displaying a Python script named 'helloworld.py'. On the right is a terminal window showing command-line interactions. A blue arrow points from the 'my\_tf\_book.ipynb' file in the file browser towards the editor window.

```
File Browser
Filter: All Files (*)
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-compsci...
Soumyendu-Sarkar-compsci...
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/examples/1_Introduction
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/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 [4]:
```

Editor - Canopy

File Browser

Filter: All Files (\*)

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-compsci...

Soumyendu-Sarkar-compsci...

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

my\_tf\_book.py

my\_tf\_book.pyc

Python ~1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more\_tf\_examples/examples/1\_Introduction

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/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 [4]:

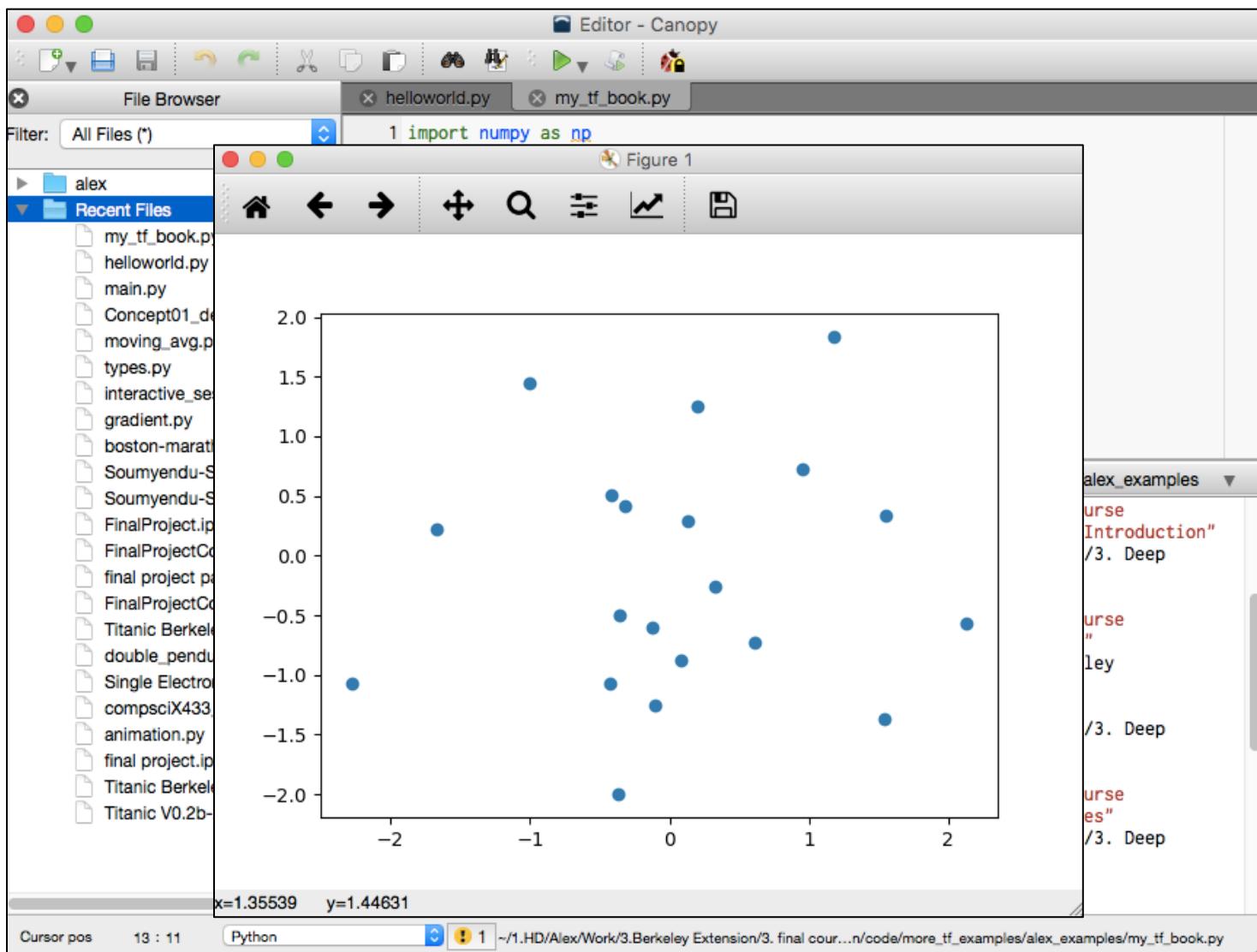
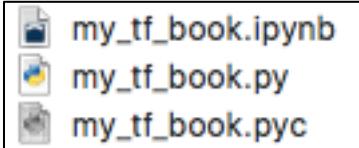
Cursor pos 13 : 11 Python 1 ~1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more\_tf\_examples/examples/1\_Introduction/my\_tf\_book.py

# Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

1. as script file

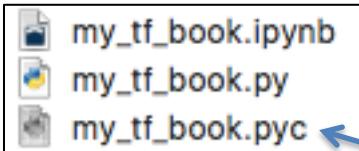


# Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

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



The screenshot shows the Canopy IDE interface. On the left is a "File Browser" window listing various Python files. In the center is a code editor window titled "Editor - Canopy" containing a script named "helloworld.py". The script imports numpy, matplotlib.pyplot, and tensorflow, then defines a session and runs a random normal distribution. On the right is a terminal window showing the user's directory path and command history. The terminal output includes commands like "pwd", "ls", and imports for "my\_tf\_book". The "my\_tf\_book.pyc" file is highlighted in the file browser, and the "my\_tf\_book.py" file is highlighted in the code editor.

```
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()

In [6]: pwd
Out[6]: '/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

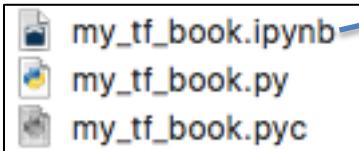
In [8]: import my_tf_book
In [9]: run my_tf_book
In [10]: |
```

# Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

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



The screenshot shows the Canopy IDE interface. On the left is the 'File Browser' pane, which lists recent files under the 'Recent Files' section. The right side contains the 'Editor - Canopy' window with two tabs: 'helloworld.py' and 'my\_tf\_book.py'. The 'my\_tf\_book.py' tab shows Python code for generating a scatter plot. A blue arrow points from the 'my\_tf\_book.ipynb' icon in the bottom-left towards the 'my\_tf\_book.py' code in the editor. Below the editor is an IPython notebook interface with several cells. Cell [6] shows the current directory ('~/1.HD/Alex/Work/3.Berkeley Extension/3. final course material/3. Deep Learning using Python/code/more\_tf\_examples/alex\_examples'). Cells [7] through [10] show the execution of the notebook file.

```
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()
```

```
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]: |
```

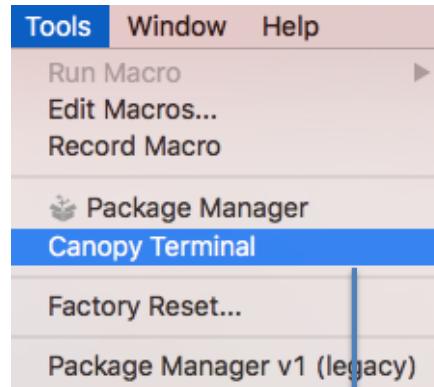
# Machine Learning With TensorFlow

- TensorFlow in Canopy:

3 ways to run code:

## 3. in notebook

(it is a .ipynb file)



```
(Canopy 64bit) Alexandomputer2:/ 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
```

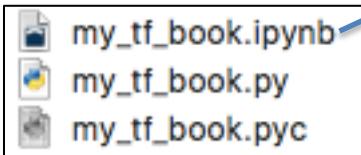
# Machine Learning With TensorFlow

- 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 Cell. The code cell 'In [1]' contains the following Python code:

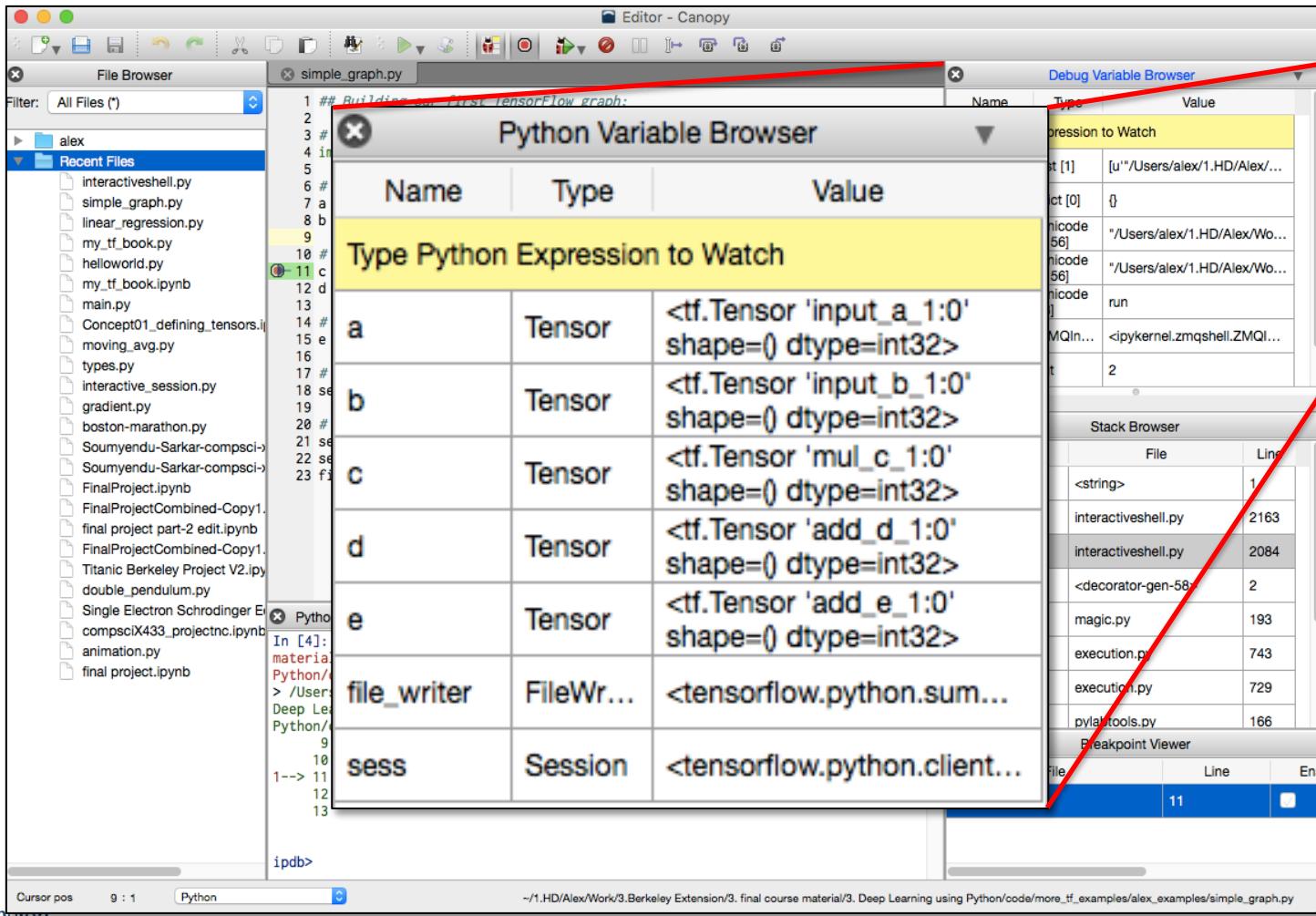
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

%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 output cell shows a scatter plot with two axes ranging from -1.5 to 2.0. The plot displays 20 data points as blue dots, showing a general positive correlation between the x and y variables.

# 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`.
```

# Installation

- TensorFlow is compatible with 4 operating Systems :
  - a) Windows OS
  - b) Mac OS
  - c) Ubuntu OS
  - d) Raspbian OS

# Machine Learning With TensorFlow

HW assignment 1:

Choose, install and configure your Python-Tensorflow environment