UC Berkeley Extension

Python for Data Analysis and Scientific Computing

X433.3 (2 semester units in COMPSCI)

Instructor Alexander I. Iliev, Ph.D.

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% - Wednesdays - use the weekend to prepare
  3. Midterm 30% - Monday - use the weekend to prepare
  4. Final Project 40% - Wednesday

- **Optional Reading for the Course**

  Title: Python for Data Analysis

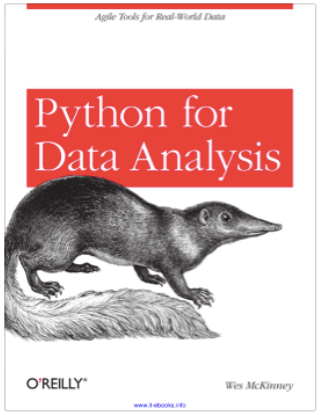
  ISBN-10: 1449319793 and ISBN-13: 978-1449319793

  Author(s): Wes McKinney

  Publisher: O'Reilly Media

  Edition Number: 1 edition

  Publication Date: November 1, 2012



Course Content Outline

- **Python for Data Analysis and Scientific Computing®**
- Python - pros and cons
- Installing the environment with core packages
- Python modules, packages and scientific blocks
- Working with the shell, IPyton and the editor HW1

- **Basic language specifics 1/2**
- Basic arithmetic operations, assignment operators, data types, containers
- Iterative programming (if/elif/else)
- Conditional expressions
- Recursion programming (for/continue/while/break)
- Functions: definition, return values, local vs. global variables

- **Basic language specifics 2/2**
- Functions (cont.): objects, methods, passing by value and reference
- Scripts, modules, packages
- I/O interaction with files
- Standard library
- Exceptions

- **NumPy 1/3**
- Why NumPy?
- Data type objects
- NumPy arrays
- Indexing and slicing of arrays

- **Matplotlib**
- What is Matplotlib?
- Basic plotting
- Tools: title, labels, legend, axis, points, subplots, etc.
- Advanced plotting: scatter, pie, bar, 3D plots, etc.

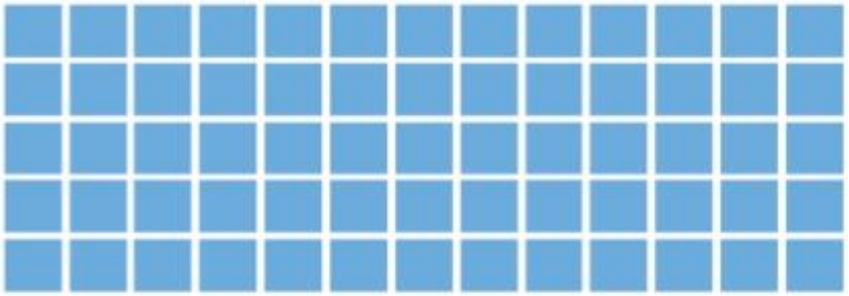
Data

- 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



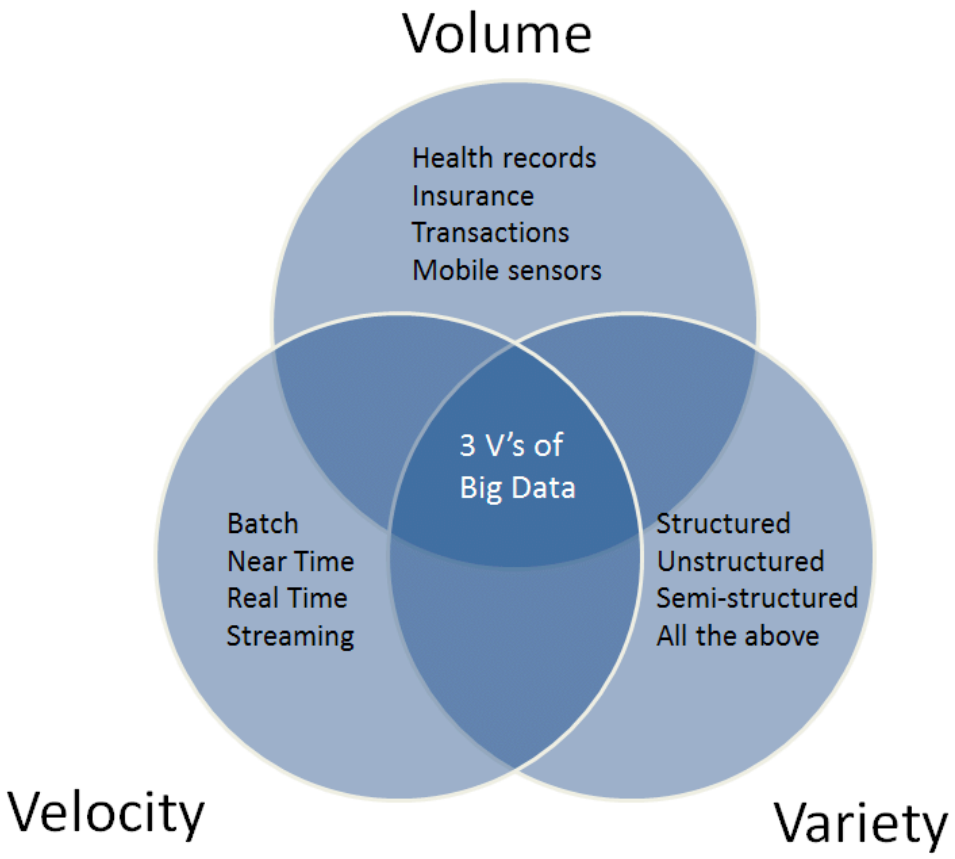
Structured Data

Unstructured Data

Graphical representations illustrate the difference between structured and unstructured data

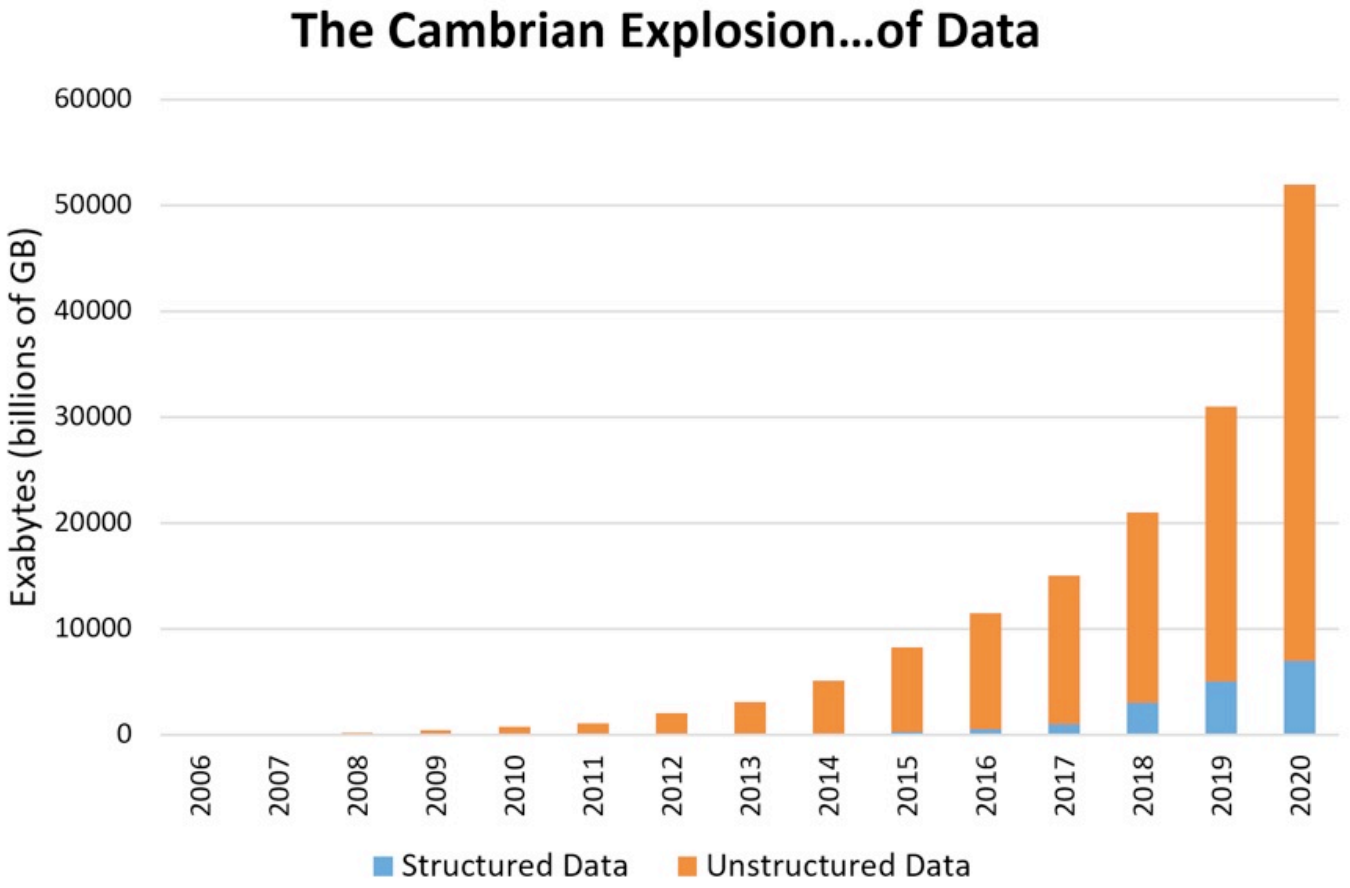
Source: https://www.eetimes.com/ and Sherpa Software

Big Data



Variability, Veracity, and Value

Data



Source: https://www.eetimes.com/

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

Source: https://www.eetimes.com/

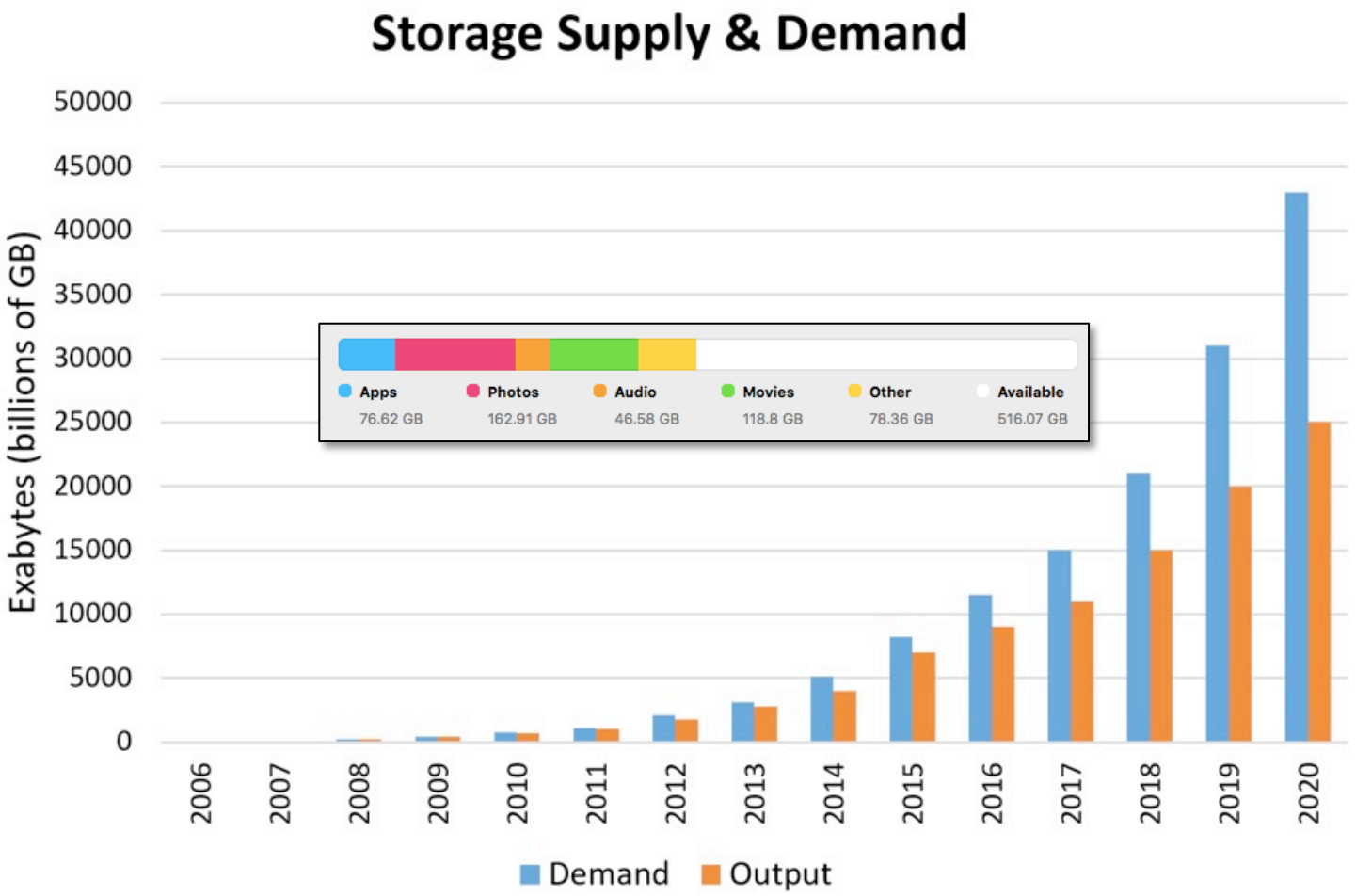
Data

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

Source: https://www.eetimes.com/

Data

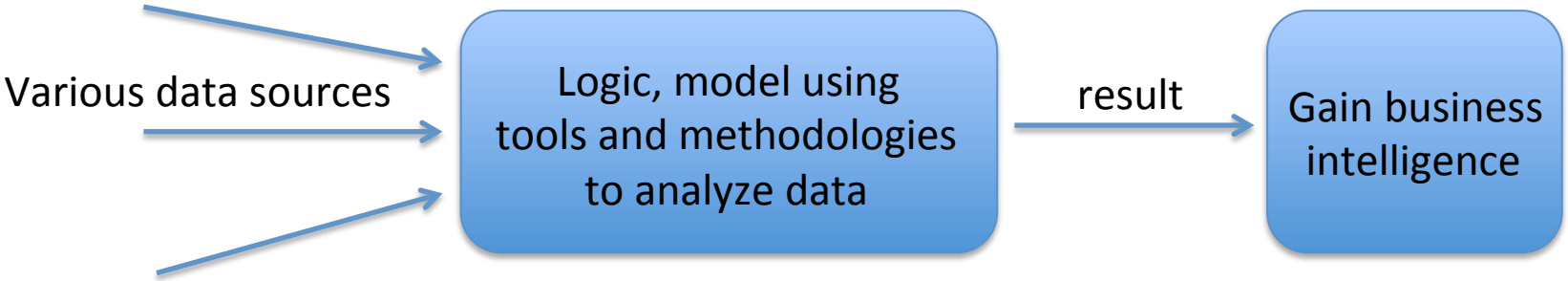


Storage supply and demand growth over two decades

Source: https://www.eetimes.com/

Why using Data Analytics?

- To solve complex scientific or business tasks
- Using complex data from various sources
- Using tools and methods to extract meaningful scientific or business insights



Data Analytics vs Data Analysis

- Data analytics is a broader field, which includes data analysis as part of its core
- Data analytics defines the methodology used behind the analysis
- Methodology means to comprehend the cognitive processes that are used in order to understand any given problem and explore data in logical fashion
- Data analytics also includes:
  - data collection, creation and extraction;
  - transformation using specific tool sets and techniques;
  - different methods for analysis;
  - communication of results.

The Problem

The Problem

- Example: Facebook 
  - Over 2 billion active users worldwide generate enormous amounts of user data
  - Every 60 seconds there are:
    - 136,000 photos are uploaded
    - 510,000 comments are posted
    - 283,000 status updates are made
  - lots of unstructured data (audio, photos, videos)

Data Analysis

- What is it?

Data analysis, is a process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making.

source: Wikipedia

Data Analysis

- How does it work?
  - First user data is collected
  - Then user behavior is studied
  - Data is packaged for advertisers
  - Enhance useful functionality (ex. shopping) and gather even more data from users (habits, preferences, etc.)

Why using Data Analytics?

- Data can be very noisy and hard to use

Types of Data Analytics

- Descriptive data analytics studies data based on previous samples
  (what happened?)
- Diagnostic data analytics studies the reasons and isolates the problem
  (why it happened?)
- Predictive data analytics predicts the possible future outcome using statistical modeling
  (what might happen?)
- Prescriptive data analytics makes logical decisions to suggest best course of action
  (what to do?)

Planning your strategy

- Problem analysis to mitigate risks
- Create your action plan
- Effort estimation
- Visualize and organize the results
- Creating business logic
- Report the results to leadership using analysis for:
  - various approach
  - different cost
  - benefits for each case

More about the data…

- Is the data available?
- How do we access it?
- How big is the data?
- What type(s) of data is it?

- How good is the source of data?
- What is the scope (local vs global)?
- How frequently is collected?
- … more

More about the data…

- Plan ahead how to work with the data:
  - Collect it
  - Store it (is possible)
  - Study the dataset
  - Prepare and clean it
  - Analyze it
  - Organize and visualize it
  - Make conclusions about the data
  - Analyze small pieces more often than wait and process the entire dataset. It just might be too late!

More about the data…

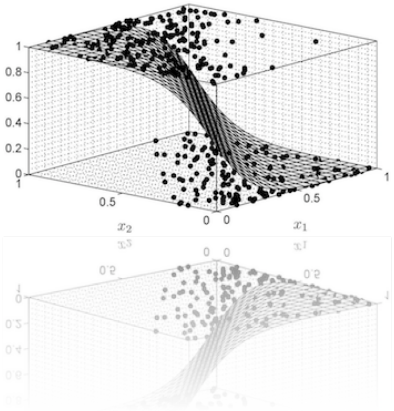
- Create and keep data dictionary containing:
  - databases
  - tables
  - fields
  - keys
  - schemas
  - data types
  - scope (local vs. global)
  - how it was loaded (using web services or ingested)
  - frequency (how often was the data collected?)

Case study

- Example 1:

data source

statistical modeling



tailored offering

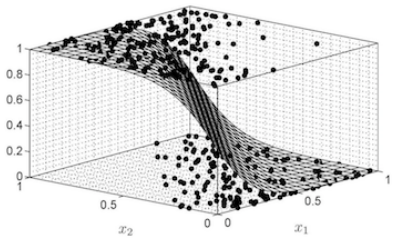
Case study

- Example 2:

Netflix

user preferences

data analysis



Result: 3,000,000 new subscribers,
who paid back the show cost !!!

The Excel power user

- Using Microsoft Excel spreadsheets containing data for analysis

Pros

- Most business will use this method because it is easy
- It is affordable
- It is versatile as can be applied quickly

Cons

- It provides a very limited toolset for analyzing and visualizing data
- Not for large and complex datasets

Python for Data Analysis and Scientific Computing

- Python implementation started in 1989 by Guido Van Rossum
- Python intended to be the successor of the ABC language
- Python 2.0 released in late 2000
- Python 3.0 released in late 2008 and was mostly incompatible with its predecessor
- EOL for Python 2.7 was initially set at 2015, but postponed to 2020
- Google announced to work on a transcompiler for Python 2.7 – Jan.2017

Python for Data Analysis and Scientific Computing

- Python comparison to Java:
  - Dynamic vs static typing
  - Braces vs indentation
  - Speed and portability
  - Interpreted vs compiled

Python for Data Analysis and Scientific Computing

- Python with NumPy and SciPy – 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
  - Mobile computing for smartphone development by using Kivy for Android and iOS
  - Web-site incorporation by using Django

Python for Data Analysis and Scientific Computing

- Python – pros and cons:

  Pros

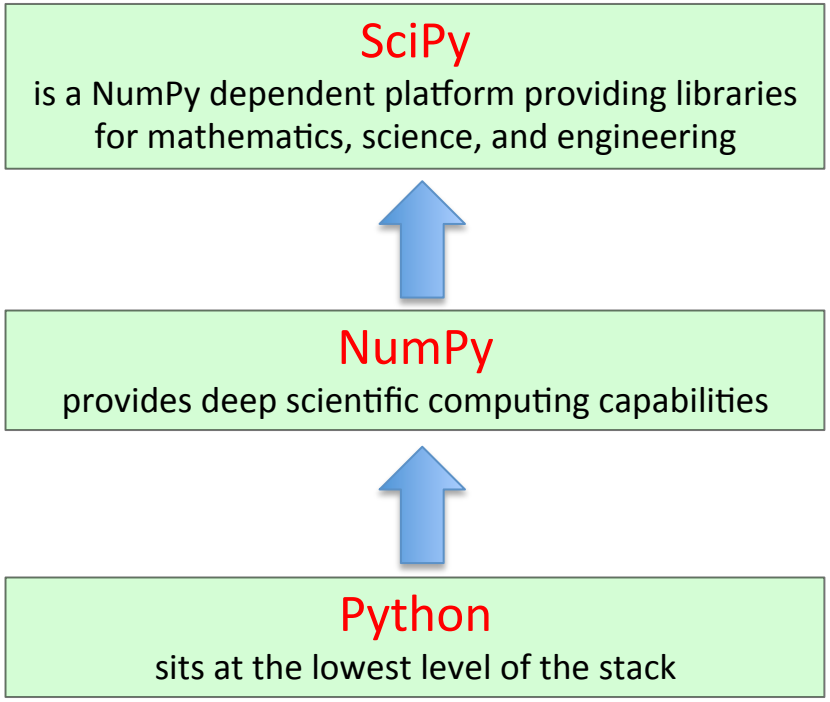
  - High-level object-oriented language that is easy to learn
  - Good readability of the code
  - Quick development time
  - Rich scientific libraries and many others like web server management, serial port access, etc.
  - Easy to share and communicate results
  - Easy to read, manipulate and process 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 flows 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
  - Not all algorithms found in other specialized platforms or toolboxes (Matlab) are available

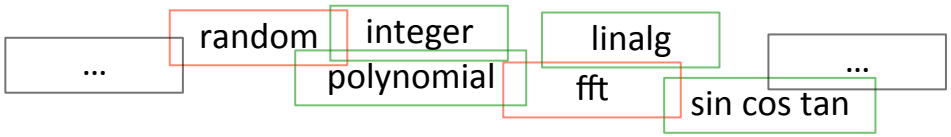
Python for Data Analysis and Scientific Computing

- Basic stack using NumPy and SciPy for Python:



Python for Data Analysis and Scientific Computing

- Python + NumPy
  - … is like Matlab
  - NumPy provides the code for any fundamental data structure:



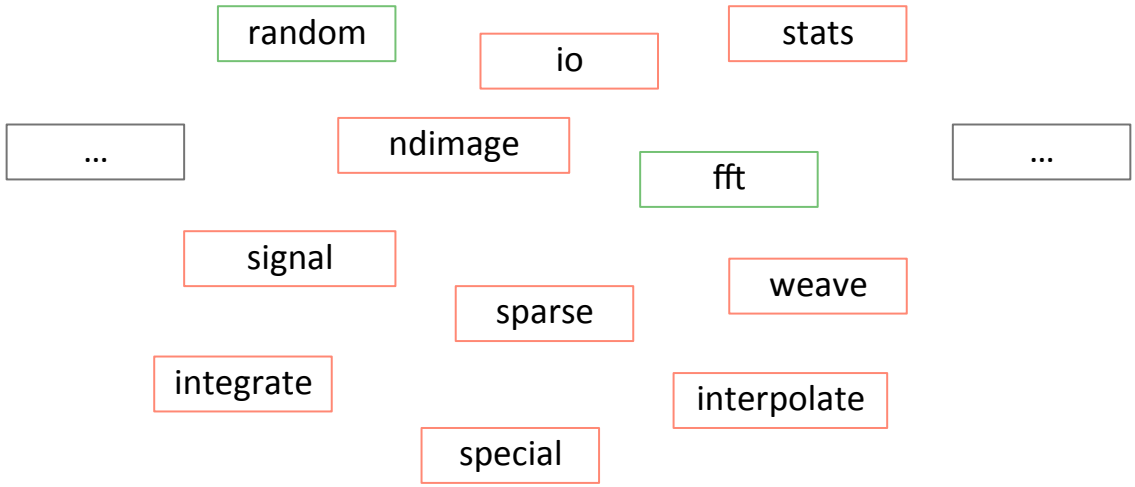
  - There two basic data structures in NumPy:

NDArray
- is an N-dimensional array type
- describes a collection of "items" of the same type
- the items are indexed using N integers

UFunc
- provides fast operations on arrays
- operates element-by-element on arrays
- example are: *sin, cos, tan, +, -, etc.*

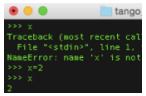
Python for Data Analysis and Scientific Computing

- Python + NumPy + SciPy
  - … is like Matlab + toolboxes
  - SciPy provides many specialized packages, which makes it very competitive and powerful platform



Python for Data Analysis and Scientific Computing

- 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:

IP[y]:

    - IPython – a GUI environment for interactive Python use with shells (terminal- and Qt-based):
      http://www.ipython.org/

    - Python(x,y) – free scientific and engineering environment for numerical computations, data analysis and data visualization:

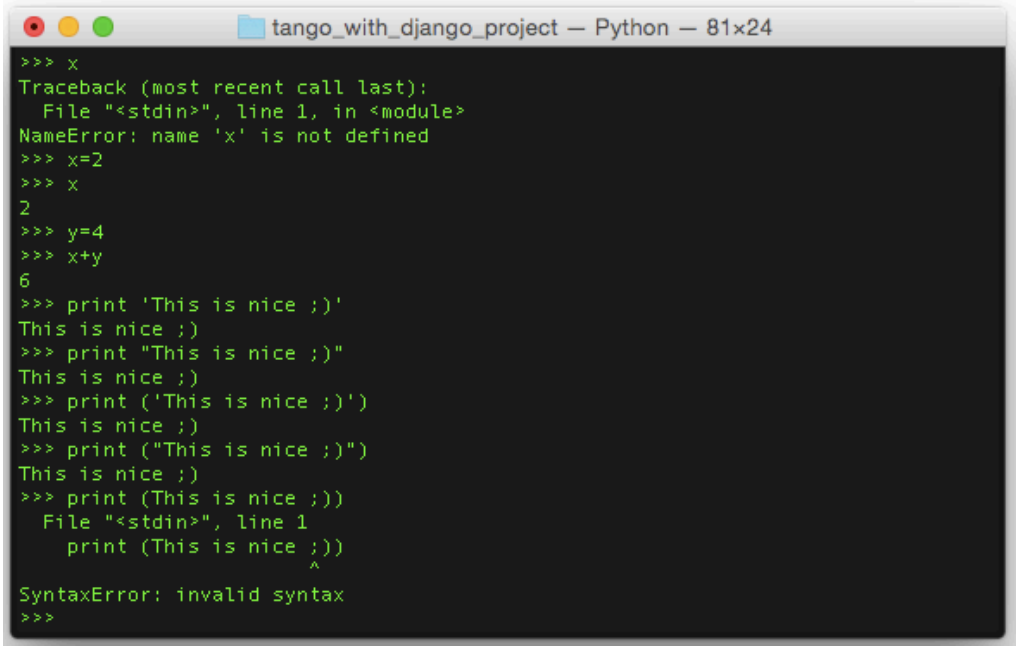
    - Canopy – Python based environment providing core scientific analytic and scientific Python packages, for scientific development and visualization:
      https://store.enthought.com/

    - IEP – Interactive Editor for Python, is an interactive cross-platform Python IDE that is simple and efficient. IEP consists of an **editor**, a **shell**, and a set of **tools** to help the developer
      http://www.iep-project.org/

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

Python for Data Analysis and Scientific Computing

- 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 it's packages and modules



Python for Data Analysis and Scientific Computing

- Using the shell (linux and OSX users):
  - Using 'pip': a tool for installing Python packages from the Python Package Index
    - In your terminal type:

      $ sudo easy_install pip

      $ sudo pip install --upgrade pip
    - Then you can install python and some packages:

      $ pip install python

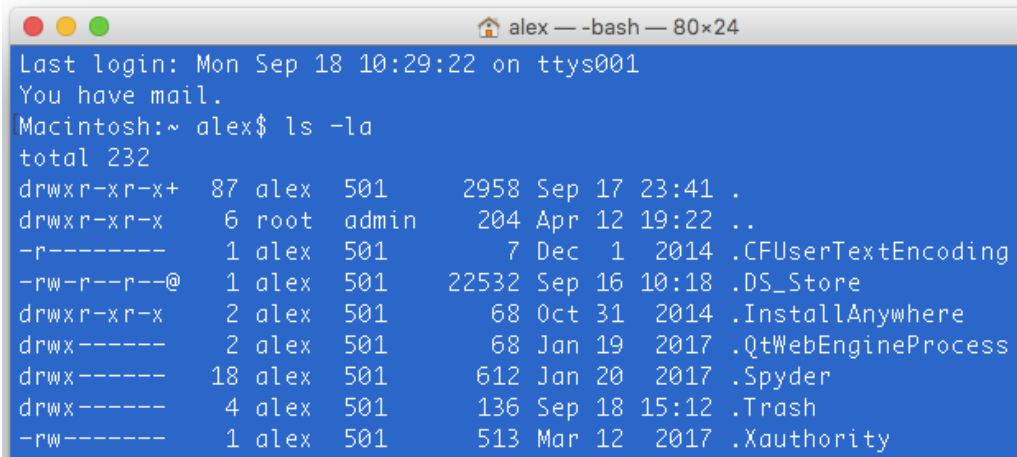
      $ pip install numpy

      $ pip install scipy

      $ pip install matplotlib

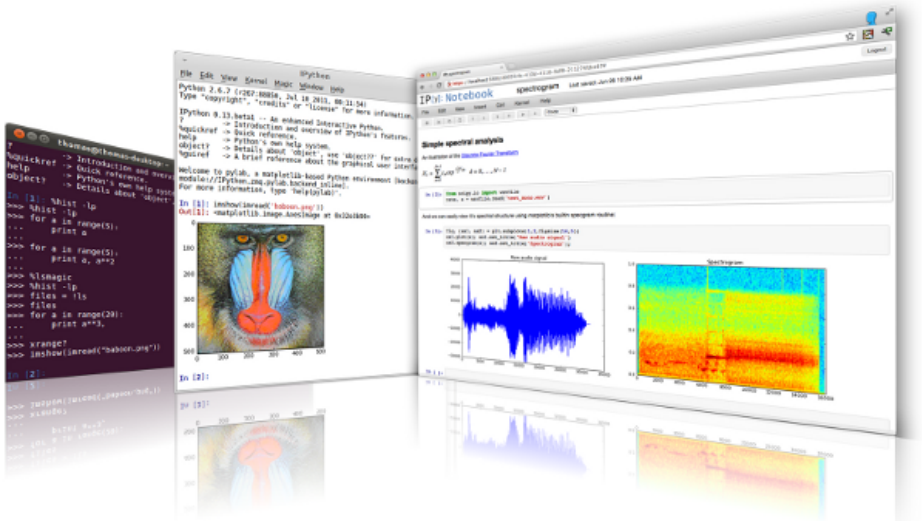
      $ pip install django

** a side note on permissions and ownership:*



Python for Data Analysis and Scientific Computing

- Working with IPyton
  - IPython
    - Is a GUI environment for interactive Python use with shells (terminal- and Qt-based) and these key features:
      - Interpreters that can be embedded into users' custom projects
      - It has a build-in web-based interactive notebook environment with complete set of shell features and support for embedded figures, animations and multimedia
      - Use of GUI toolkits and support for interactive data visualization
      - A high-performance library for parallel computing portable to multicore systems, clusters and cloud computing



Python for Data Analysis and Scientific Computing

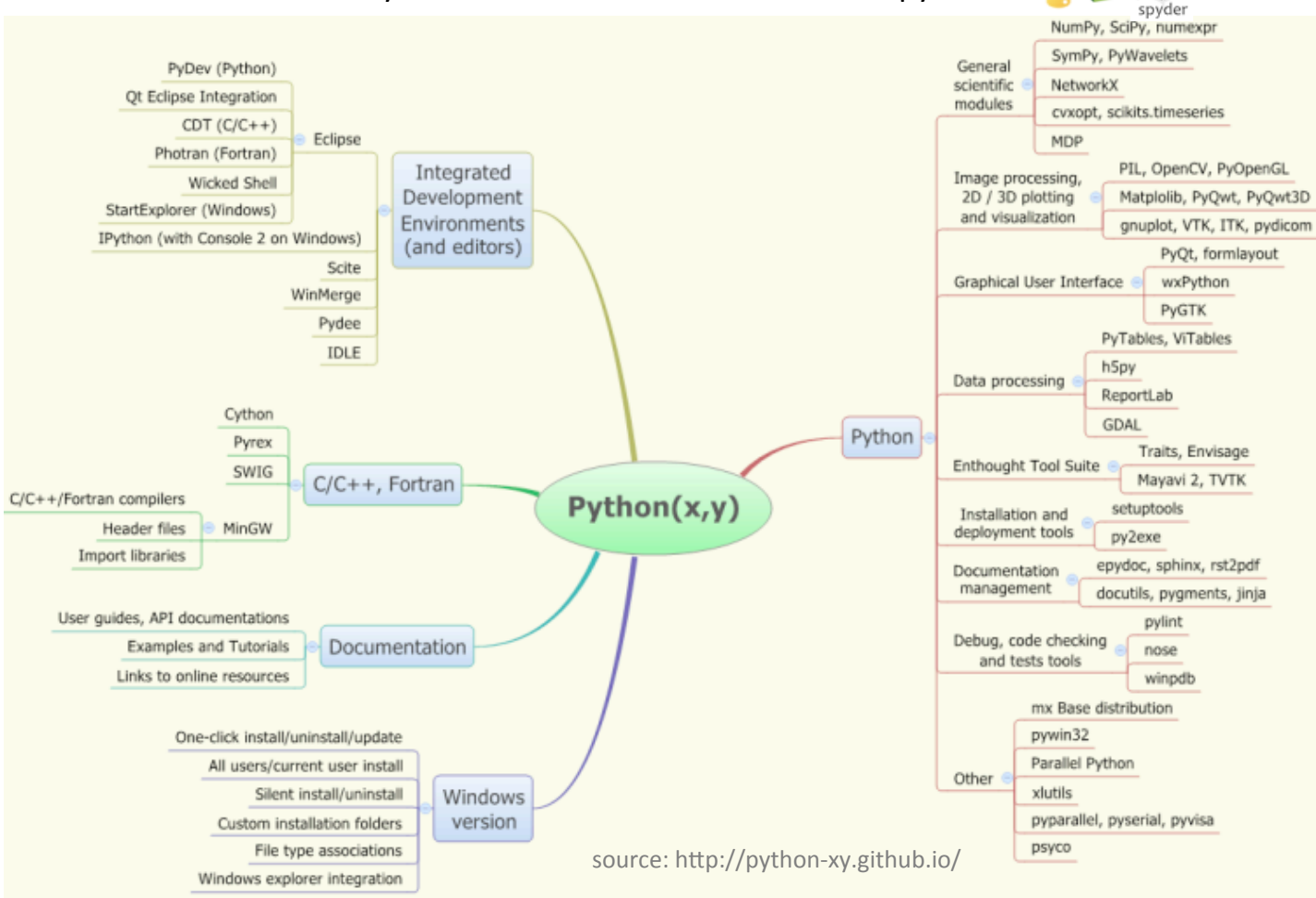
- Working with IPyton and the editor
  - IPython and the editor
    - History, persistent across sessions
    - Caching of output results during a session
    - Extensible tab completion, with support by default for completion of python variables and keywords, filenames and function keywords
    - Comprehensive system of 'magic' commands for controlling the environment and performing many tasks related either to IPython or the operating system
    - A rich configuration system with easy switching between different setup
    - System shell access with user-extensible alias setup
    - Session logging and reloading
    - Code is easy to embed and use in other Python programs and GUIs
    - Embedded debugger

Python for Data Analysis and Scientific Computing

- Working with alternative environments:
  - Python(x,y) – a free scientific-oriented Python Distribution based on Qt and Spyder –

It offers:

- object-oriented prog.
- MATLAB-like syntax
- parallel computing
- 2D & 3D plotting
- any level coding



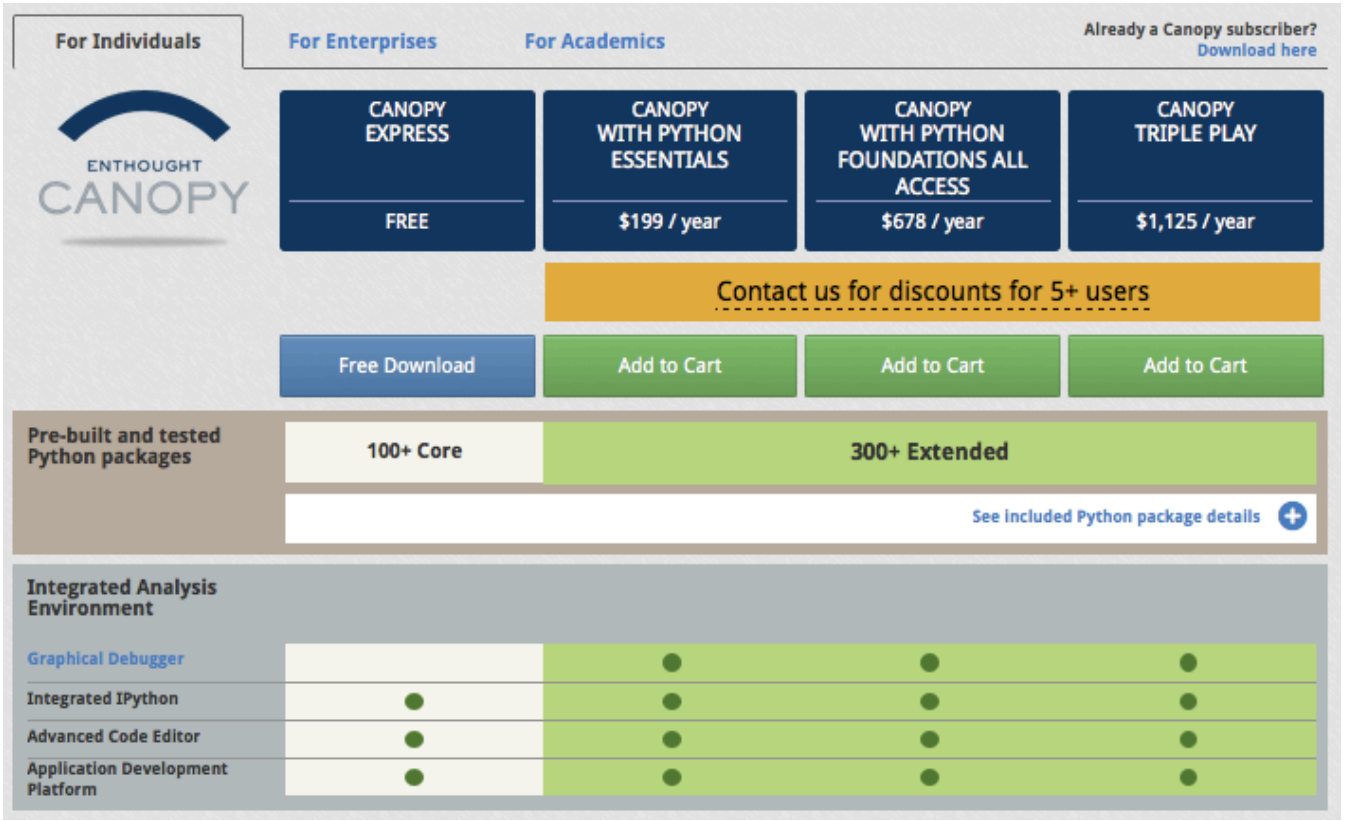
source: http://python-xy.github.io/

Python for Data Analysis and Scientific Computing

- 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

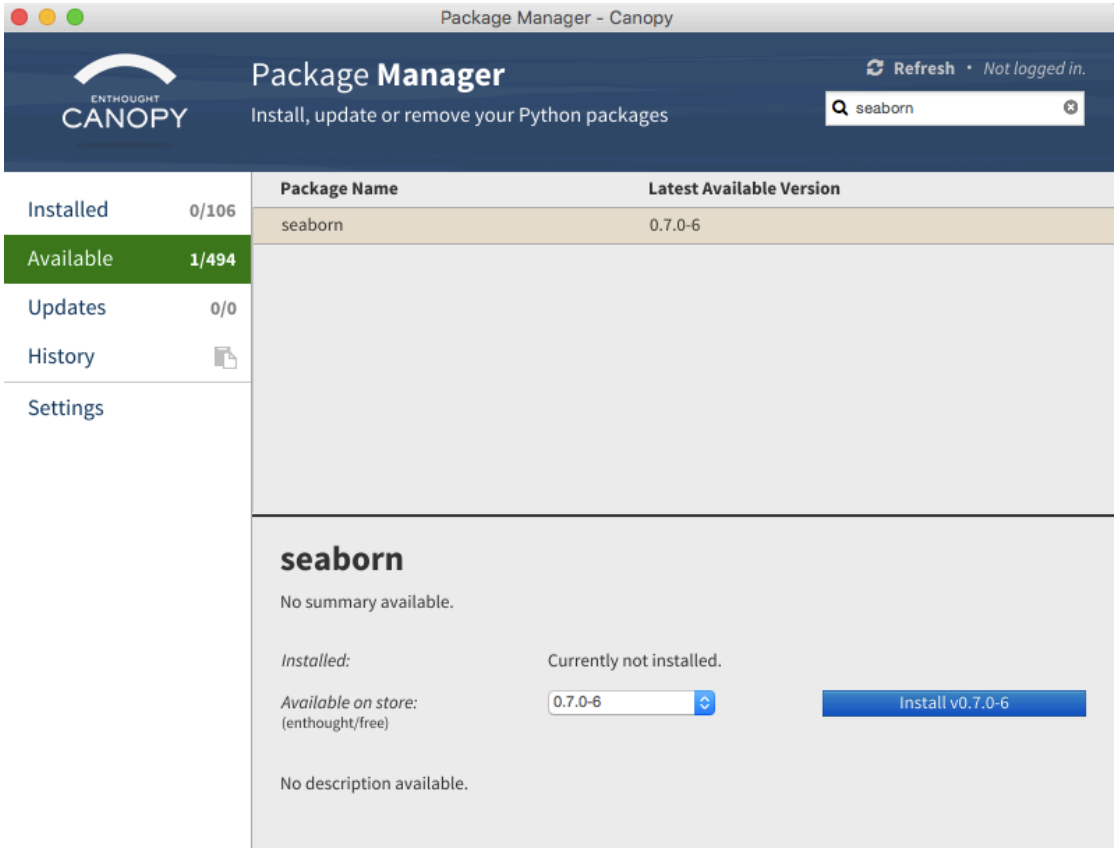


Python for Data Analysis and Scientific Computing

- 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

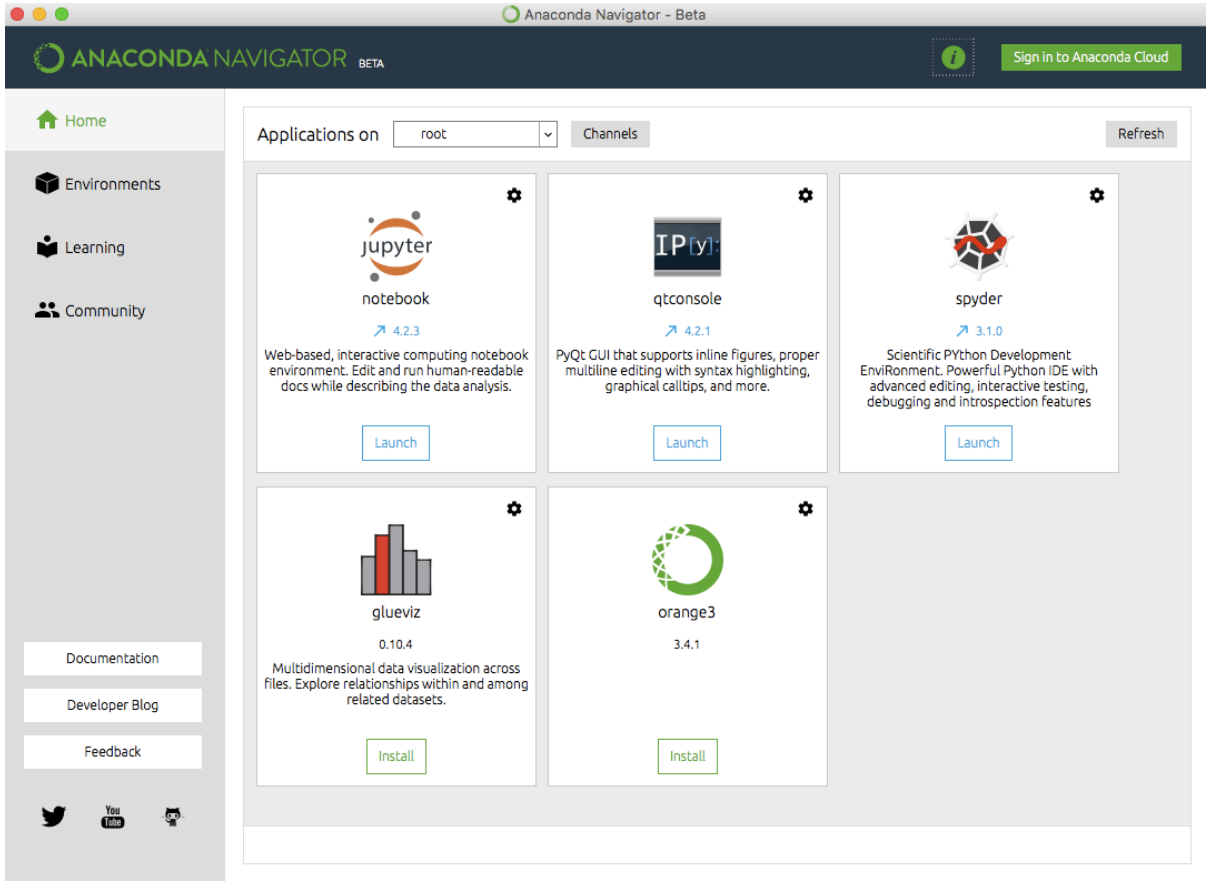


Python for Data Analysis and Scientific Computing

- Working with alternative environments:
  - Anaconda Navigator

Available are:

- Jupyter Notebook
- IP[y] qtconsole
- Spyder
- glueviz



Python for Data Analysis and Scientific Computing

- Working with alternative environments:
  - Anaconda Navigator

Available are:

- Jupyter Notebook
- IP[y] qtconsole
- Spyder
- glueviz

- **Jupyter**: is also available in Canopy: it is a web-based interactive computing notebook environment. Easy to read and describe the data analysis

- **IP[y] qtconsole**: is a PyQt based GUI working with various figures and plots among other editing features

- **Spyder**: is for scientific python development with interactive testing env and debugging features. It can be installed independently in some platforms such as Linux

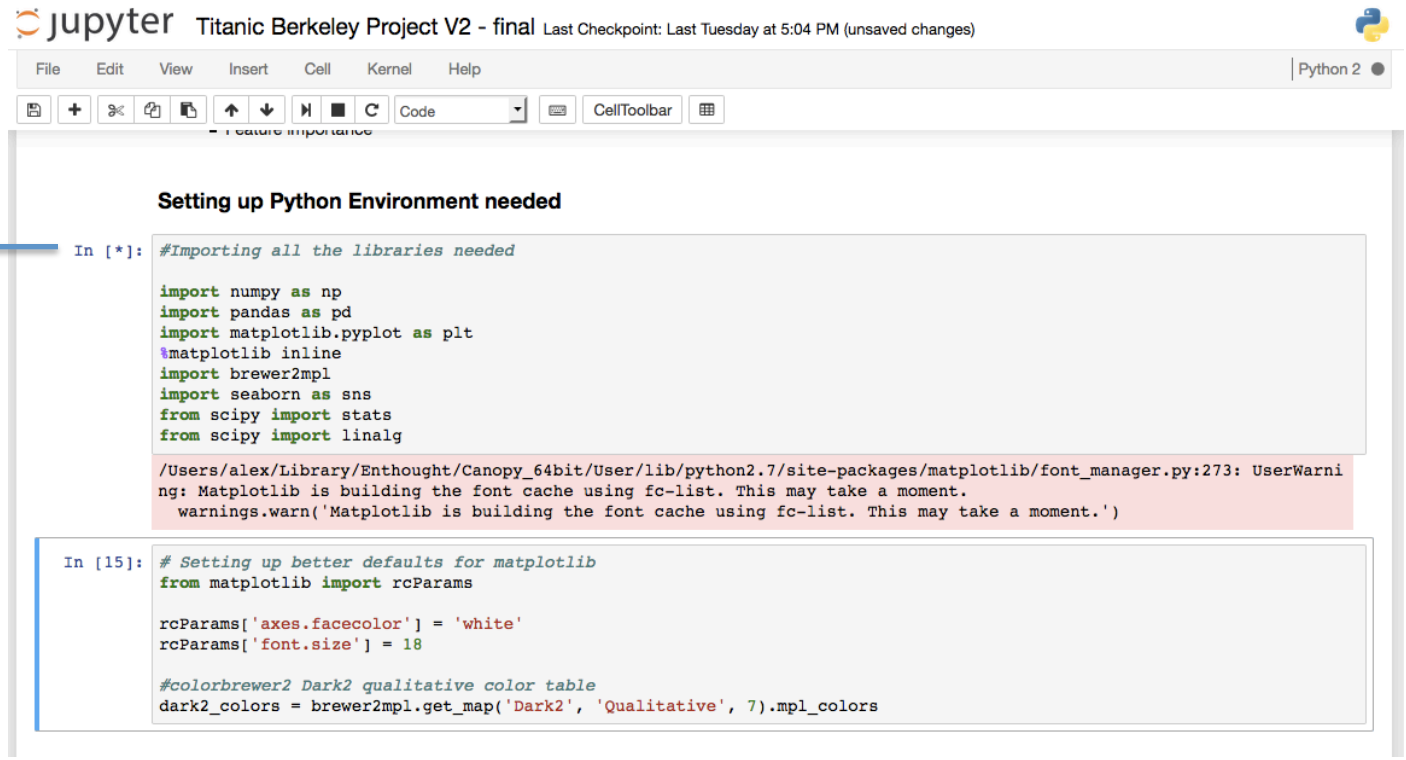
- **glueviz**: n-dimensional data visualization using multiple files. View the relationships among multiple datasets

Python for Data Analysis and Scientific Computing

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

Using the browser

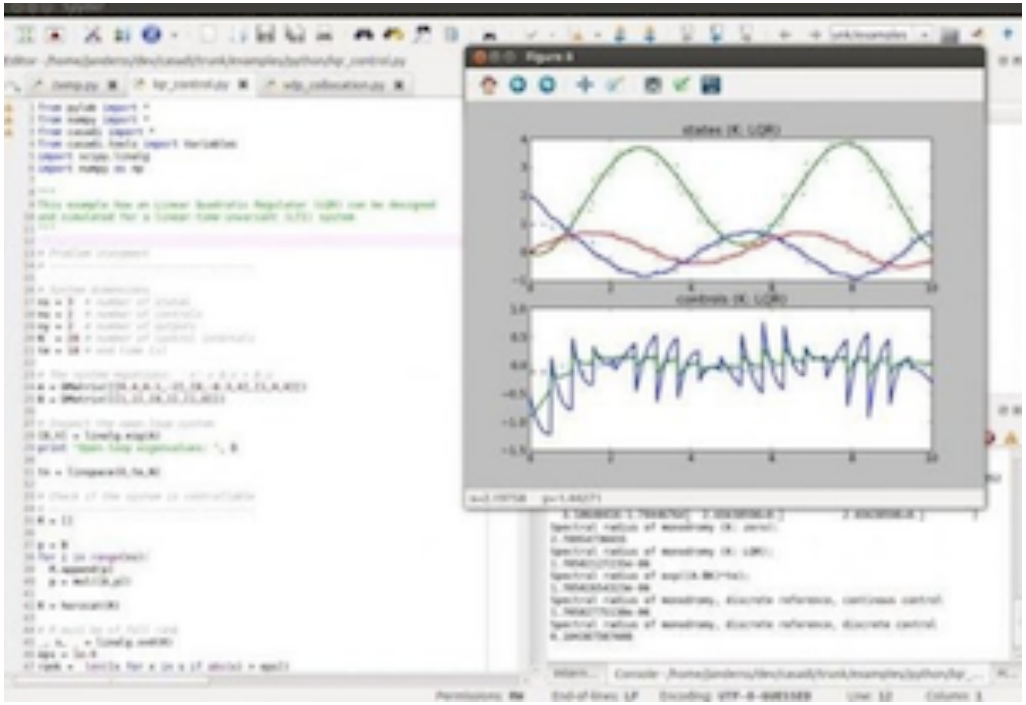
- [*] the line is in a process of executing
- Execute your code sequentially using Shift+Enter



source: https://store.enthought.com/

Python for Data Analysis and Scientific Computing

- Working with alternative environments:
  - Spider …
    - ***Spider*** is a free Open-source

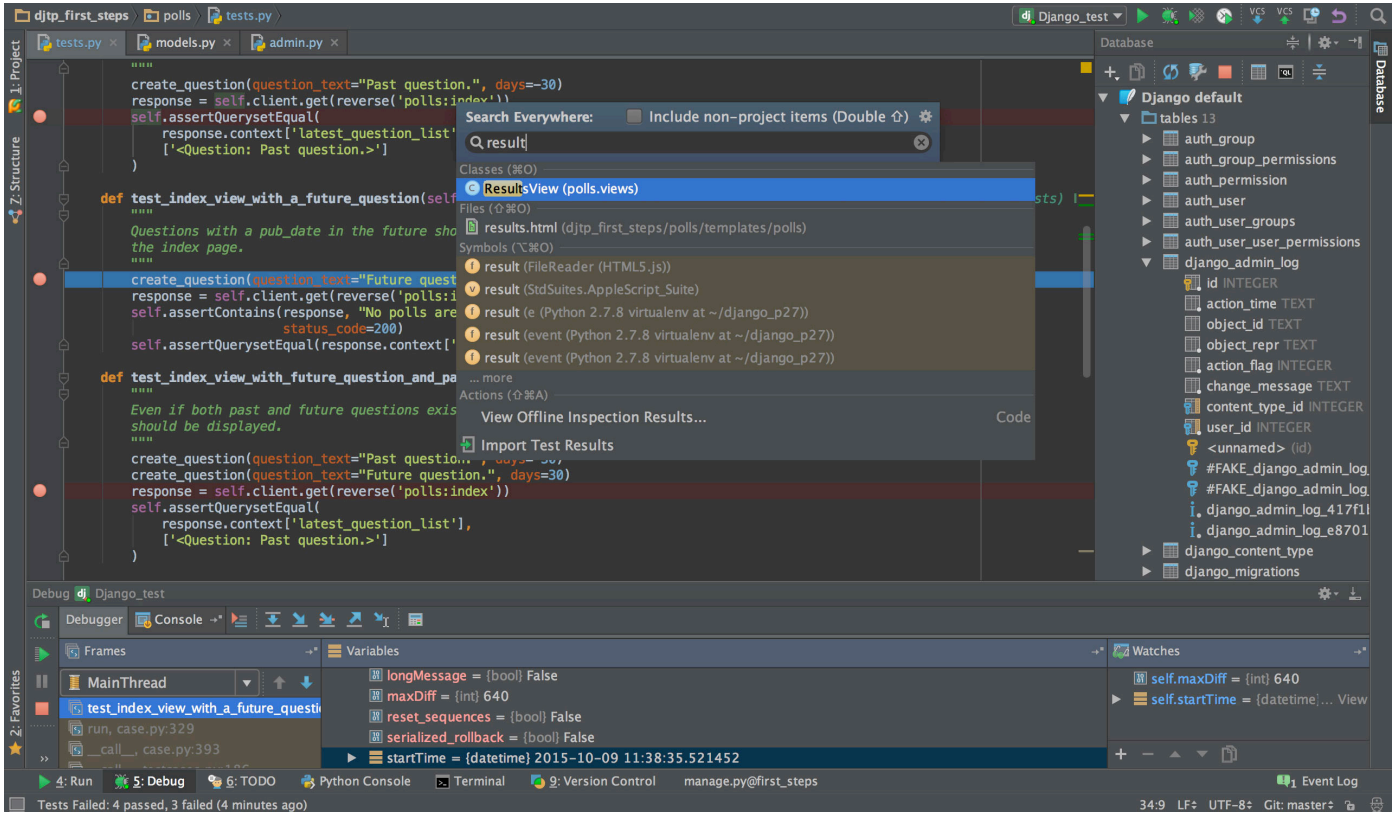


Python for Data Analysis and Scientific Computing

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

It offers:

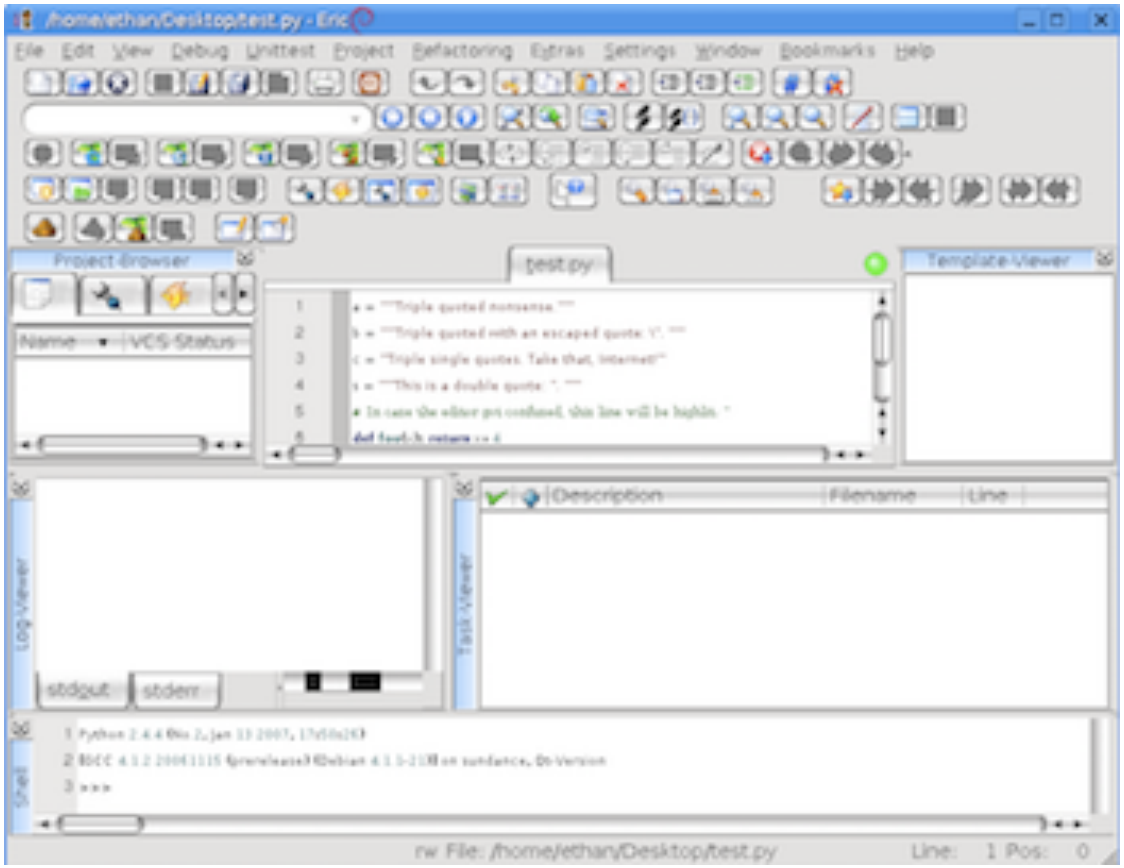
- object-oriented prog.
- save time by auto completing the routines
- on-the-fly error checking
- focus on bigger picture
- many features
- many packages



source: https://www.jetbrains.com/pycharm/

Python for Data Analysis and Scientific Computing

- Working with alternative environments:
  - Eric …
    - ***Eric*** is a free Open-source



Python for Data Analysis and Scientific Computing

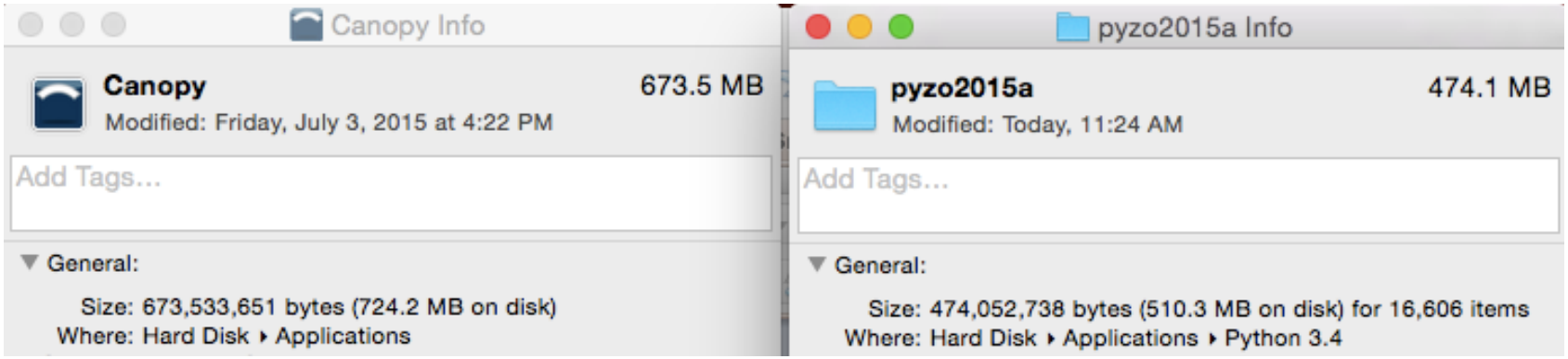
- Working with alternative environments:
  - Pyzo … what?

PYZO: PYTHON TO THE PEOPLE

    - ***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
    - Pizo 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

Python for Data Analysis and Scientific Computing

- Working with alternative environments:
  - Pyzo takes less space than Canopy (without Canopy debugger installed)



Python for Data Analysis and Scientific Computing

- Python packages for data analysis:

  - **NumPy**: provides fast precompiled functions for numerical routines, large n-dimensional arrays, and matrices, as well as high-level mathematical functions to operate on these arrays

  - **SciPy**: provides scientific, technical, and deep analytical computing functionality and contains modules for statistics, optimization, linear algebra, integration, interpolation, and plenty more

  - **Matplotlib**: is a plotting library used by NumPy that provides the ability to embed plots into applications using GUI toolkits like wxPython, Qt, or GTK+

  - **Pandas**: offers data structures and manipulation and analysis of numerical tables

Python for Data Analysis and Scientific Computing

- Python packages for data analysis:

  - **Scikit-learn**: contains various algorithms for classification, regression and clustering such as: naive Bayes, SVM, logistic regression, random forests, gradient boosting, k-means and DBSCAN. It is designed to use the scientific libraries NumPy and SciPy

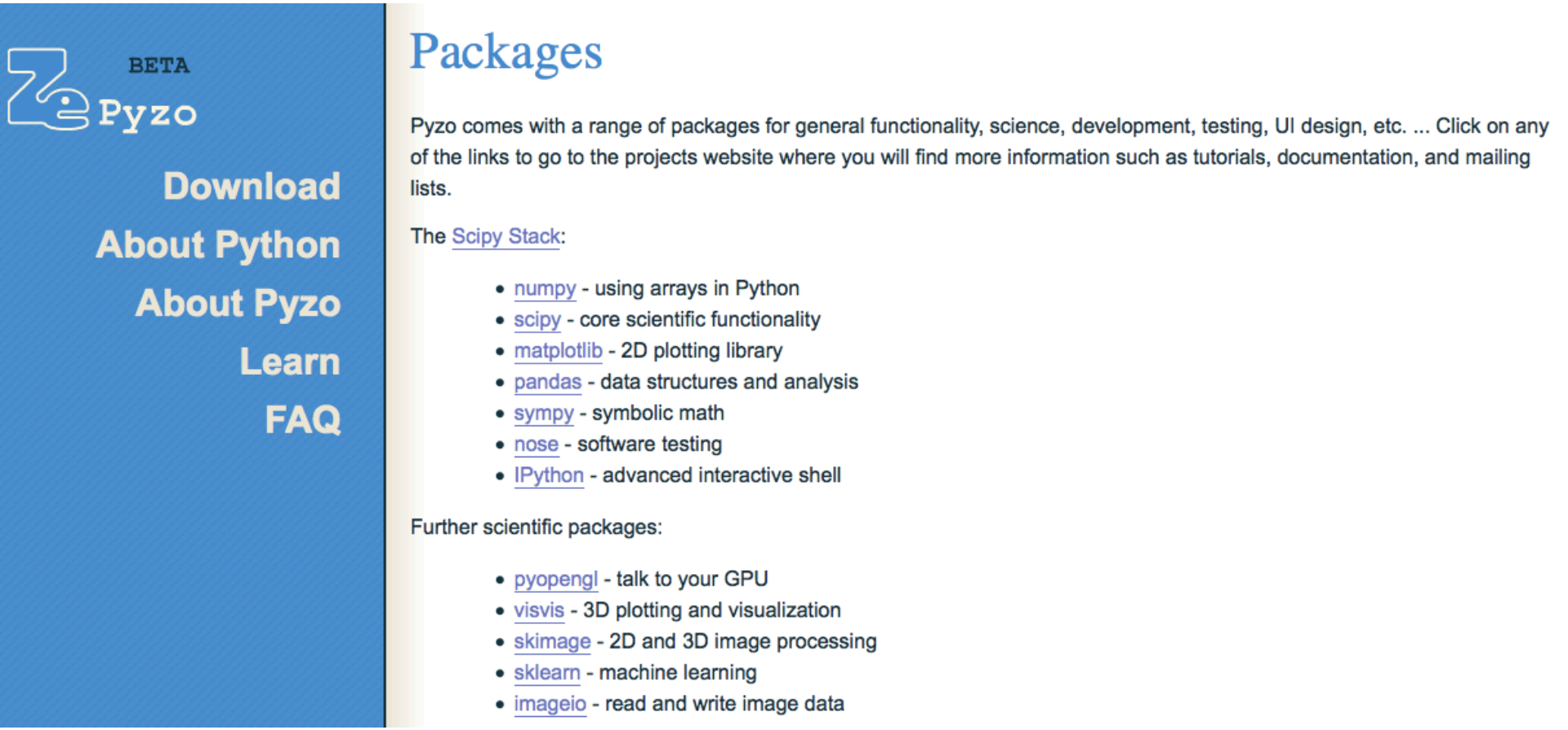
  - **Theano**: with Theano you can define, evaluate, and optimize mathematical expressions applied to n-dimensional arrays

  - **Statsmodels**: provides statistical models, statistical tests, plotting functions, and descriptive statistics, while working with different types of data

  - **Other**: Mlpy, NLTK, Keras, PyBrain, Hebel, nolearn, Orange

Python for Data Analysis and Scientific Computing

- Working with Pyzo



source: http://www.pyzo.org/packages.html#packages

Python for Data Analysis and Scientific Computing

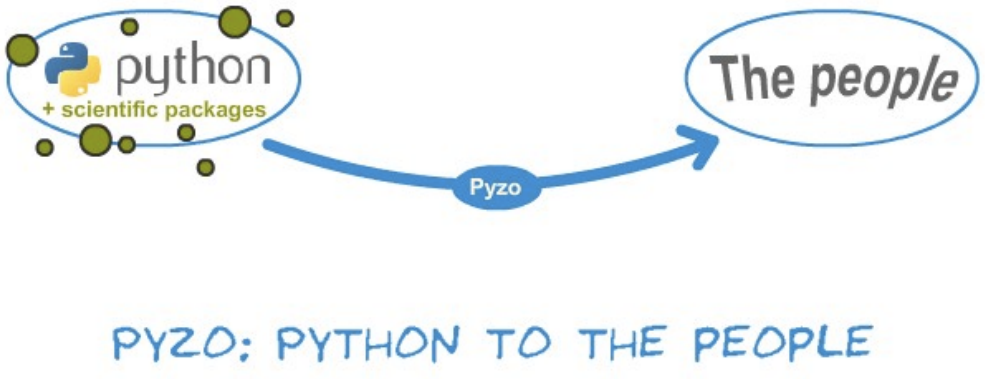
- Working with Pyzo
  - Useful magic commands:

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



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

Python for Data Analysis and Scientific Computing

- How to check your packages? .. (In your interactive editor - I use Pyzo)

```
In [3]: import pip
In [4]: pip
Out[4]: <module 'pip' from '/Applications/Python 3.4/pyzo2015a/lib/python3.4/site-packages/pip/__init__.py'>
In [5]: pip list
Out[5]:
You are using pip version 6.0.8, however version 7.0.1 is available.
…. snip snip snip …
Django 1.8.1
…. snip snip snip …
matplotlib 1.4.2
…. snip snip snip …
numpy 1.9.1
…. snip snip snip …
scipy 0.15.1          ->          one can use: "scipy.__version__" to check the package version (scipy in this case)
```

or

```
In [5]: pip.get_installed_distributions()
Out[5]:
…. snip snip snip …
Django 1.8.1 (/Applications/Python 3.4/pyzo2015a/lib/python3.4/site-packages),
…. snip snip snip …
matplotlib 1.4.2 (/Applications/Python 3.4/pyzo2015a/lib/python3.4/site-packages),
…. snip snip snip …
numpy 1.9.1 (/Applications/Python 3.4/pyzo2015a/lib/python3.4/site-packages),
…. snip snip snip …
scipy 0.15.1 (/Applications/Python 3.4/pyzo2015a/lib/python3.4/site-packages),
```

- Using:

```
In [6]: help()
```

Python for Data Analysis and Scientific Computing

- Python modules, packages and scientific blocks
  - A module is a Python file with .py extension containing definitions and statements
  - Modules implement a set of functions and can be *imported* into other modules using the *import* command:

Example:

Lets create our first Python program called – *lecture1.py*:

```
# Let's make a simple addition until certain criteria is not met
def add(a):       # the criteria to be met is a
        x, y = 2, 4
                while x < a:
                print(x,y),
                x = x + y
```

Possible usage:

```
# importing a module called 'lecture1'
import lecture1          -> must be in the same working folder
lecture1.add(24)
```

Tips:

1. *to check what's loaded in memory simply look at the workspace*
2. *if you enter an infinite loop hit 'ctrl-c' to break the loop and stop the execution*
3. *try debugging your program using breakpoints*

Python for Data Analysis and Scientific Computing

- Python modules, packages and scientific packages
  - A *module* in Python is a source file that can expose classes, functions and global variables
  - A *package* in Python is a directory of one or several Python modules (source .py files)

Example:

```
# 1. this is how we import a package:
import math
        Usage:
        math.acos(0.3454453)

# 2. this is how we import a module from a package:
import acos from math … ( or: from math import acos)
        Usage:
        acos(0.3454453)
```

Tips:

1. *Observe what's loaded in workspace in the two cases above*
2. *you can delete a particular item from workspace by using **'del(item)'***
3. *to clear the workspace use **'reset'***
4. *The built-in function **'dir()'** is used to find what's currently defined and shown in workspace*

Python for Data Analysis and Scientific Computing

- Python modules, packages and scientific blocks
  - **'sys'**: includes information about the modules available to us. It includes built-in and imported modules
  - **Imported modules:** usually Python programs are a combination of several modules imported with a main application

    They can be seen in the following way:

```
import sys, textwrap
names = sorted(sys.modules.keys())
name_text = ' || '.join(names)
print(textwrap.fill(name_text))
```

  - **Built-in modules:** the interpreter can be built and compiled with some C modules, that do not appear in the imported modules list since they were included in the package

    They can be seen in the following way:

```
import sys
for module_name in sys.builtin_module_names:
    print(module_name)
```

  - **The search path:** the search path for modules is managed as a list saved in **sys.path.** The default path list consist of the script directory used to start the application and the current working directory

```
sys.path
```

Python for Data Analysis and Scientific Computing

Magic commands in **IPython**:

List of important **Magic** commands in **IPython**:

%automagic	magic [-brief,]
ls	lsmagic
pwd	ed, edit
cd	load
mv	run
dir	time
mkdir	writefile
rm, rmdir	env
who, whos	system
alias, unalias	config
hist, history	save
dhist	logoff logon logstart logstate logstop

Python for Data Analysis and Scientific Computing

HW assignment 1:

Make sure you have chosen, installed and configured your Python-NumPy-SciPy environment