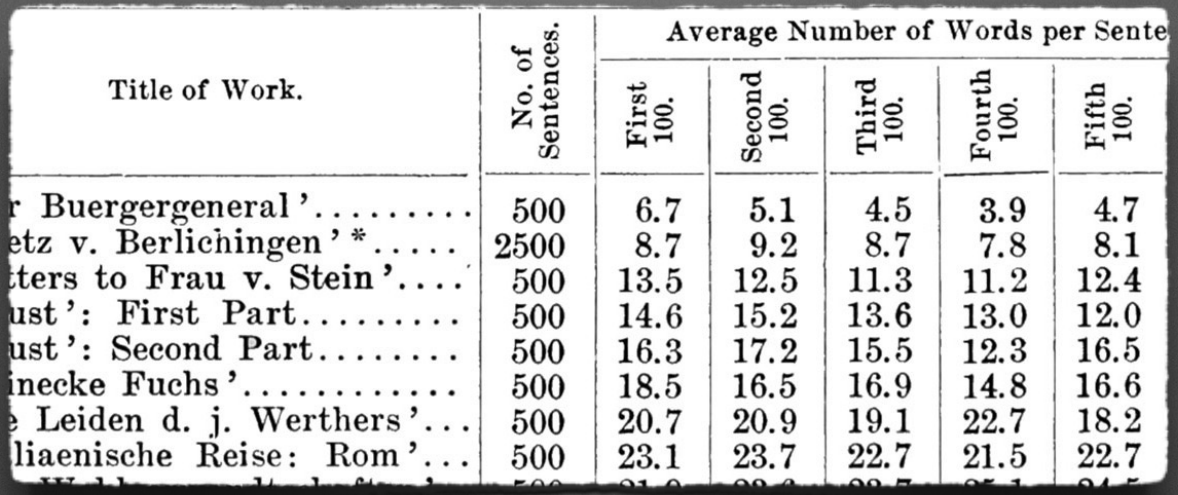


Title of Work.	No. of Sentences.	Average Number of Words per Sente				
		First 100.	Second 100.	Third 100.	Fourth 100.	Fifth 100.
r Buergergeneral'.....	500	6.7	5.1	4.5	3.9	4.7
etz v. Berlichingen'*.....	2500	8.7	9.2	8.7	7.8	8.1
tters to Frau v. Stein'....	500	13.5	12.5	11.3	11.2	12.4
ust': First Part.....	500	14.6	15.2	13.6	13.0	12.0
ust': Second Part.....	500	16.3	17.2	15.5	12.3	16.5
inecke Fuchs'.....	500	18.5	16.5	16.9	14.8	16.6
e Leiden d. j. Werthers'...	500	20.7	20.9	19.1	22.7	18.2
liaenische Reise: Rom'...	500	23.1	23.7	22.7	21.5	22.7

Introduction to Data Analysis with Python

Jordi Vitrià, PhD
Universitat de Barcelona

Introduction to Data Analysis with Python

Syllabus

Data Science Steps & Python Data Science Stack

Data Science Plumbing & Agile Data Science

Why Python?

What is IPython?

Data Science Steps

What do I want? Does it have sense?

Question

What are my data sources? How reliable are they?

Acquire

How do I develop an understanding of the content of my data?

Describe

What are the key relationships in my data?

Discover

How do I develop an understanding of the content of my data?

Analyze

What are the likely future outcomes?

Predict

Are my expectations fulfilled?

Evaluate

Acquire: How do I get my data?
- Web Scraping.
- Data Base queries.
- Access to bulk data stores.

Processing: How I do clean and separate my data?
- Identification: filter data.
- Outliers.
- Imputation: missing value processing.
- Reduction: dimensionality reduction.
- Normalization: duplicates, ranges, format, coordinates, units, etc.
- Feature extraction.

Aggregation: How do I collect and summarize my data?
- Basic Statistics: mean, std, box plots, scatter plots, counts, etc.
- Distribution fitting.
- Feature aggregation.

Enrichment: How do I add more information to my data?
- Feature engineering.
- Search for additional data sources.

Discover: What are the key relationships in my data?
- Clustering (How do I segment the data to find natural groupings?)
- Visualization (Are there unexpected relationships?)

Analyze: How do I model my data?
- Variable selection (How do I determine important variables?)
- Probabilistic modeling (How are my variables related?)

Predict: What are the likely future outcomes?
- Regression (How do I predict the future?)
- Classification (How do I predict a category?)
- Recommendation (How do I predict relevant conditions?)

Evaluate: Are the outcomes generic and robust?
- Statistical Testing.
- Model performance.

Python Data Stack

Scrapy
PyDB
PyMongo
Numpy
Pandas
SciPy
SciKit Learn
MatplotLib
SymPy
IPython
Cython
Bokeh
Blaze
PySpark
StatsModels
Shogun
PyMC
seaborn
PyTables
cvskit
sqlite3
plotly
NetworkX
nltk
gensim
twython

Data Science Plumbing

Agile Development & Model Deployment

Logs from servers, sensors, transactions, etc. Events must be **serialized** in a common format.

Events

Events are collected from different sources and **aggregated** to bulk storage.

Collector

This is a **filesystem** capable of parallel access by concurrent processes.

Bulk Storage

Data processing with scripting languages with **out-of-core** processing capabilities.

Batch Processing

These are multinode stores to **publish data** for consumption by web apps.

Distributed Store

Web application server.

Application Server

Presentation of the data as an **interactive experience** + **story telling**.

Browser

Why Python?

- Python is a modern, open source, **object-oriented programming** language, created by a Dutch programmer, Guido van Rossum, in 1991.
- Officially, it is an **interpreted scripting language**. There are several implementations, being the most used the one implemented in C.
- It offers the power and flexibility of lower level (i.e. compiled) languages, without the steep learning curve, and without most of the associated debugging pitfalls.
- The language is very **clean and readable**, and it is available for almost every modern computing platform.
- It is widely used: Google, DropBox, Deutsche Borse, NASA, etc.

Why Python?

In Data Science, **programming is mostly about discovering solutions by writing code and using a language to express them**.

Python offers a number of advantages to data scientists:

- Powerful and easy to use.
- Anything that can be coded in C, FORTRAN, or Java can be done in Python, almost always in fewer lines of code, and with fewer debugging headaches.
- Its standard library is extremely rich, including modules for string manipulation, regular expressions, file compression, mathematics, scientific programming, profiling and debugging.
- Unnecessary language constructs, such as END statements and brackets are absent, making the code efficient and easy to read.
- Python is object-oriented, which allows data structures to be abstracted in a natural way.

Why Python?

- It is interactive
  - Python may be run interactively on the command line. Commands may entered serially followed by the Return key.
  - This is useful for exploratory programming.
- It is extensible
  - Python is often referred to as a “glue” language, meaning that it is a useful in a mixed-language environment. C or FORTRAN code can be compiled directly into Python programs.
  - Sometimes, it can be slow relative to compiled languages. In many cases this performance deficit is due to a short loop of code that runs thousands or millions of times. Such bottlenecks may be easily removed by coding a function in C or Cython that can be compiled into a Python module.

Why Python?

- There is a vast body of Python third party modules:
  - **NumPy**: Numerical Python (NumPy) is a set of extensions that provides the ability to specify and manipulate array data structures.
  - **SciPy**: An open source library of scientific tools for Python, SciPy supplements the NumPy module. SciPy includes modules for graphics and plotting, optimization, integration, special functions, signal and image processing, genetic algorithms, ODE solvers, and others.
  - **Matplotlib**: Matplotlib is a python 2D plotting library which produces publication-quality figures.
  - **pandas**: A module that provides high-performance, easy-to-use data structures and data analysis tools. In particular, the DataFrame class is useful for spreadsheet-like representation and mannipulation of data.
  - **IPython**: An enhanced Python shell, designed to increase the efficiency and usability of coding, testing and debugging Python.

Why Python?

- Strong points:
  - Viable alternative to SAS/Matlab/R/…
  - Uptake in the financial sector.
  - Upcoming generation of programmers with Python as a first language.
  - Python communities in HPC/Scientific Computing.

- Weak points
  - No (public) great success story.
  - Weak support.

Why Python?

Python in Big Data Workflows

Cloud

Counting/ETL

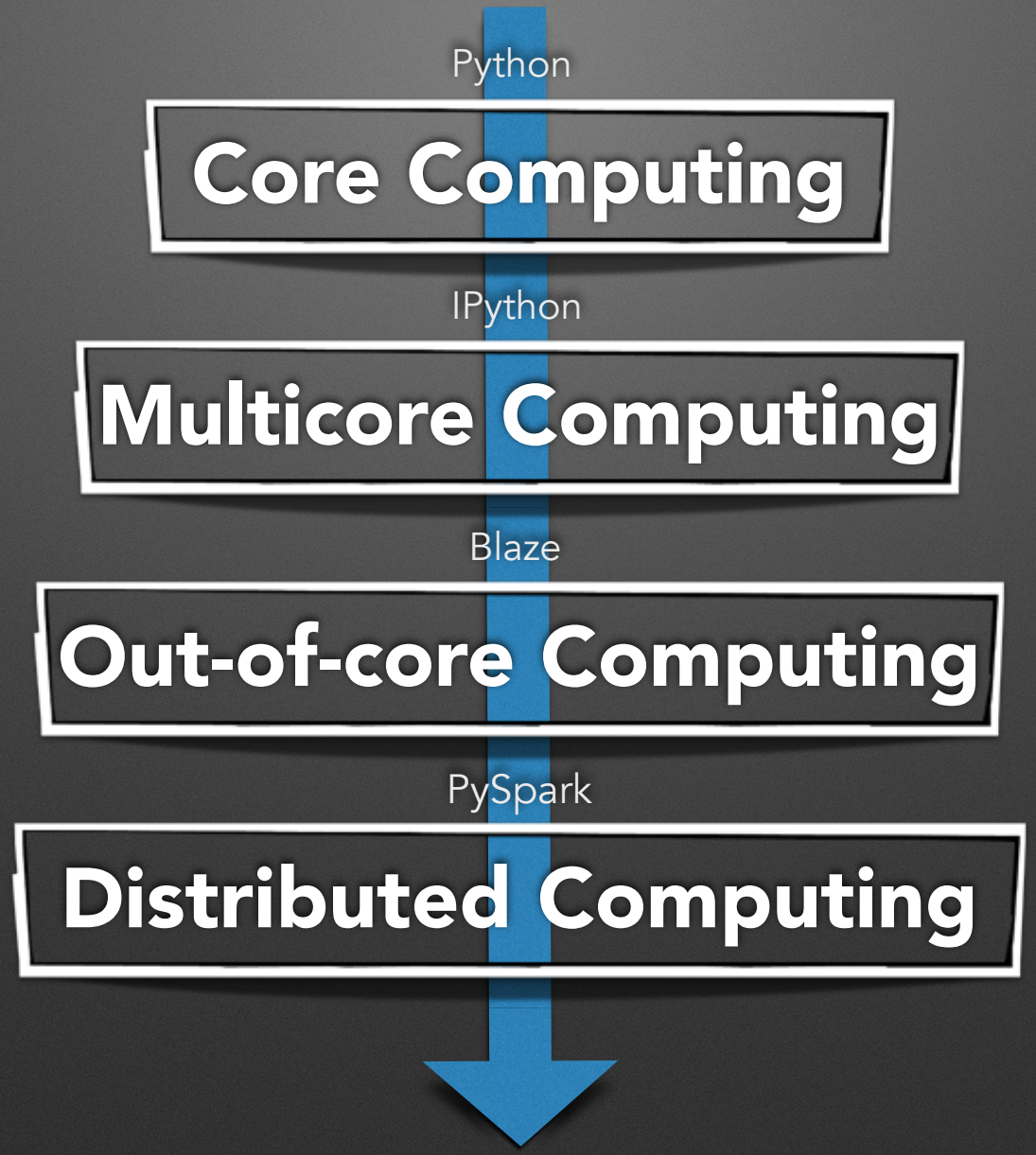
HDF5
Spark
Hadoop
SQL
...

My Machine

Insight/Reporting

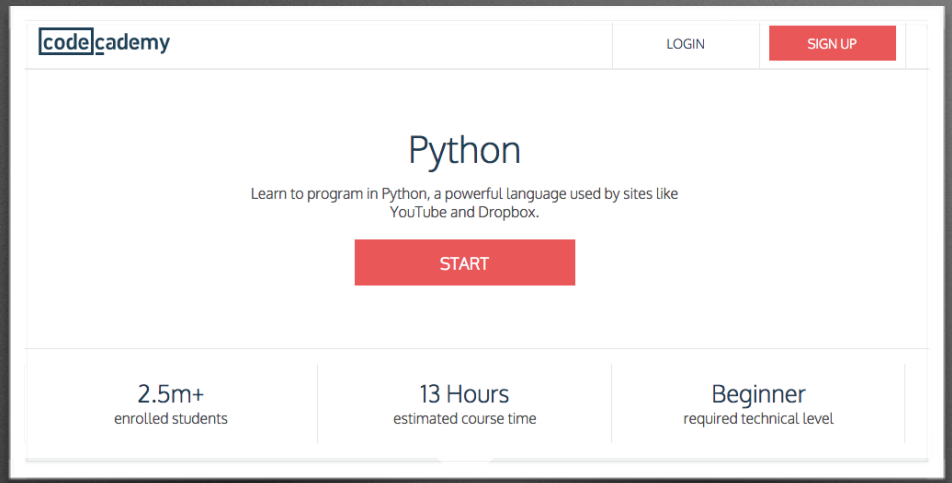
pandas
scikit-learn
Blaze
PySpark
PyMongo
...

Why Python?


Python
Core Computing
IPython
Multicore Computing
Blaze
Out-of-core Computing
PySpark
Distributed Computing

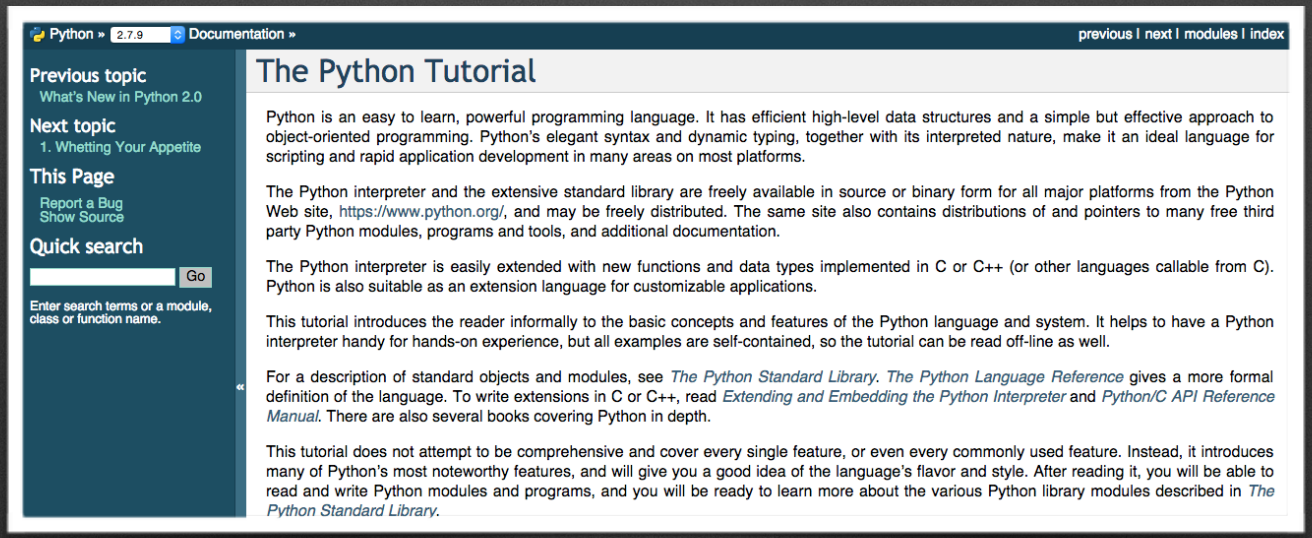
More on Python

www.codecademy.com



http://anandology.com/python-practice-book/

www.python.org



What is IPython?

I is for interactive

In this class, we will use **IPython notebooks** for teaching and programming assignments.

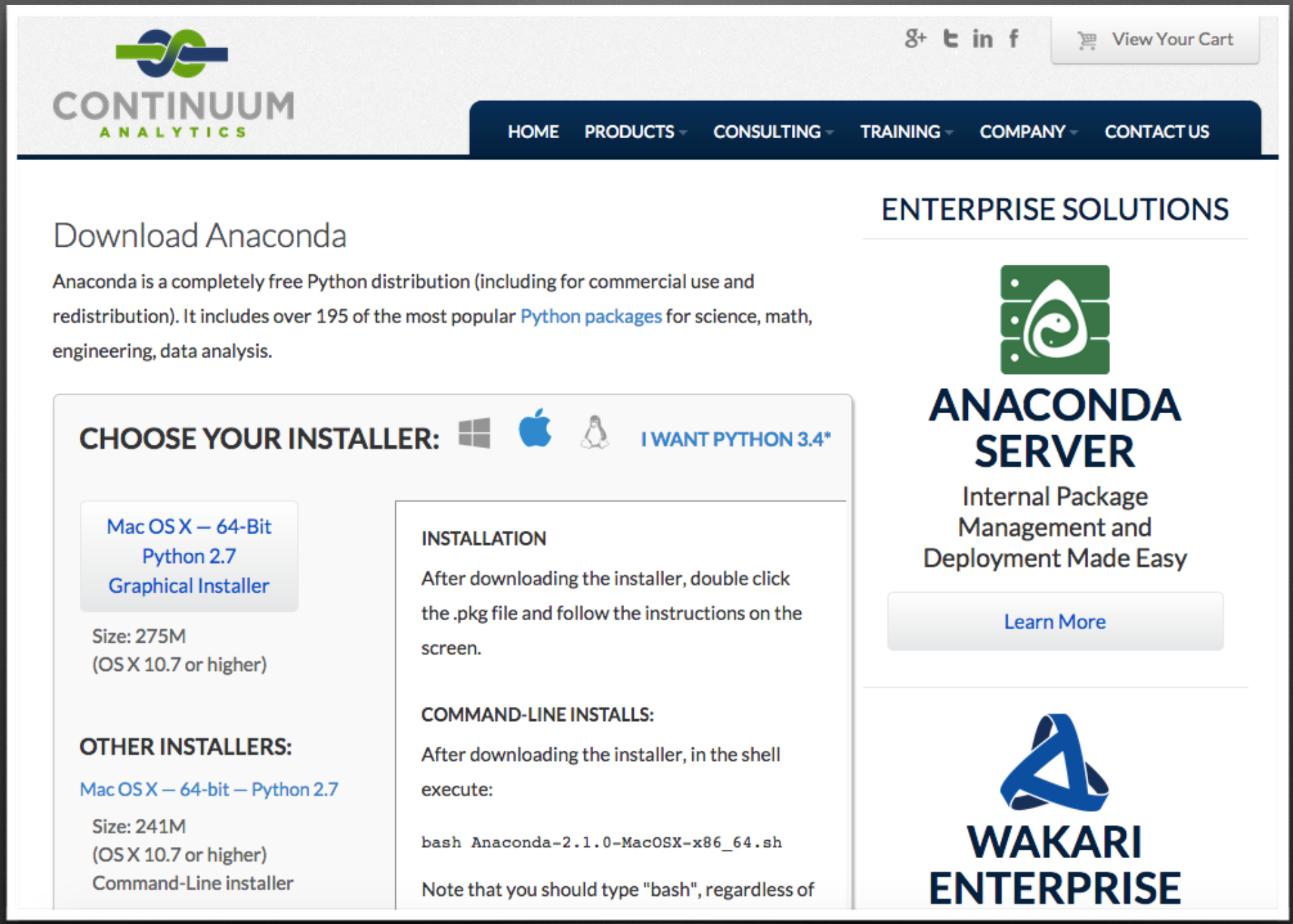
An IPython notebook lets you write and execute Python code in your web browser.

IPython notebooks make it very easy to tinker with code and execute it in bits and pieces; for this reason IPython notebooks are widely used in scientific computing.

IPython

For new users who want to get up and running with minimal effort, we suggest you to download and install Anaconda (http://docs.continuum.io/anaconda/) which provide a setup based on Python 2.7.

Anaconda is a free collection of powerful packages for Python that enables large-scale data management, analysis, and visualization for Business Intelligence, Scientific Analysis, Engineering, Machine Learning, and more.


CONTINUUM
ANALYTICS
View Your Cart
HOME
PRODUCTS
CONSULTING
TRAINING
COMPANY
CONTACT US
Download Anaconda
Anaconda is a completely free Python distribution (including for commercial use and redistribution). It includes over 195 of the most popular Python packages for science, math, engineering, data analysis.
CHOOSE YOUR INSTALLER:
I WANT PYTHON 3.4*
Mac OS X — 64-Bit
Python 2.7
Graphical Installer
Size: 275M
(OS X 10.7 or higher)
OTHER INSTALLERS:
Mac OS X — 64-bit — Python 2.7
Size: 241M
(OS X 10.7 or higher)
Command-Line installer
INSTALLATION
After downloading the installer, double click the .pkg file and follow the instructions on the screen.
COMMAND-LINE INSTALLS:
After downloading the installer, in the shell execute:
bash Anaconda-2.1.0-MacOSX-x86_64.sh
Note that you should type "bash", regardless of
ENTERPRISE SOLUTIONS
ANACONDA SERVER
Internal Package Management and Deployment Made Easy
Learn More
WAKARI ENTERPRISE

IPython

To run IPython,

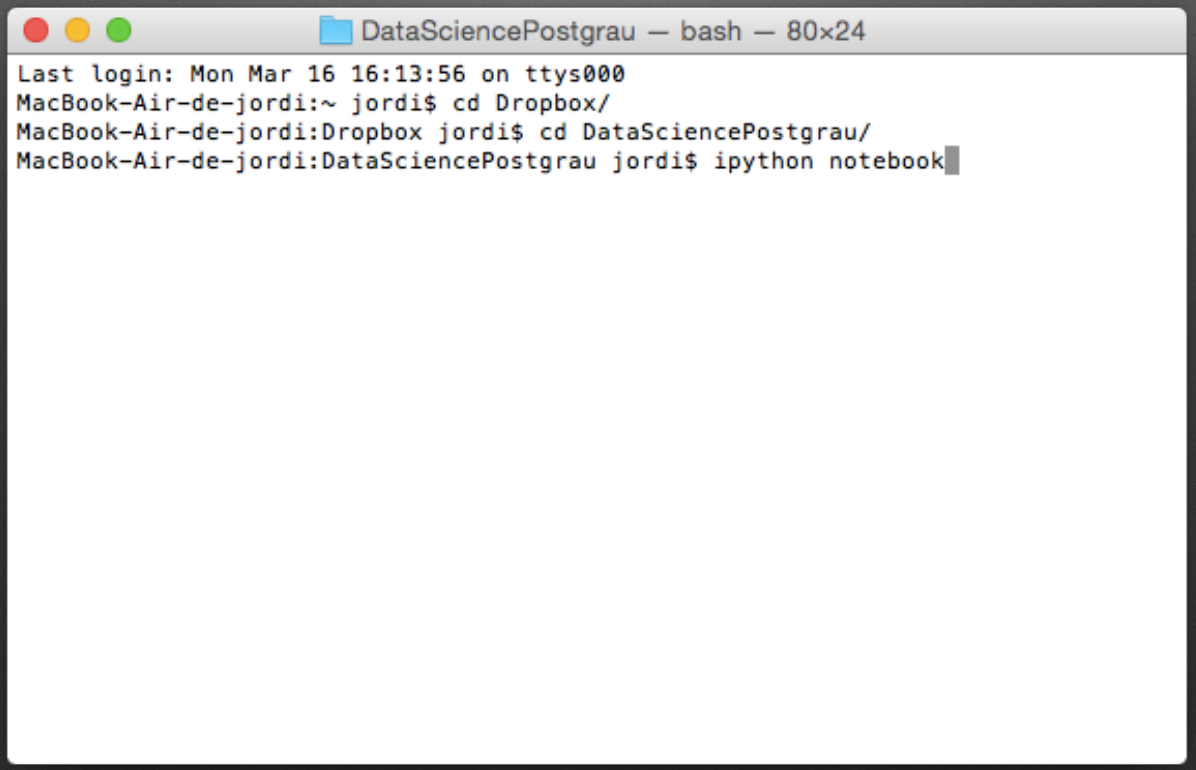
(1) Using the command line, navigate to the directory you want to store notebooks, and

(2) Start running a notebook server from the command line using the following command:

```
ipython notebook
```

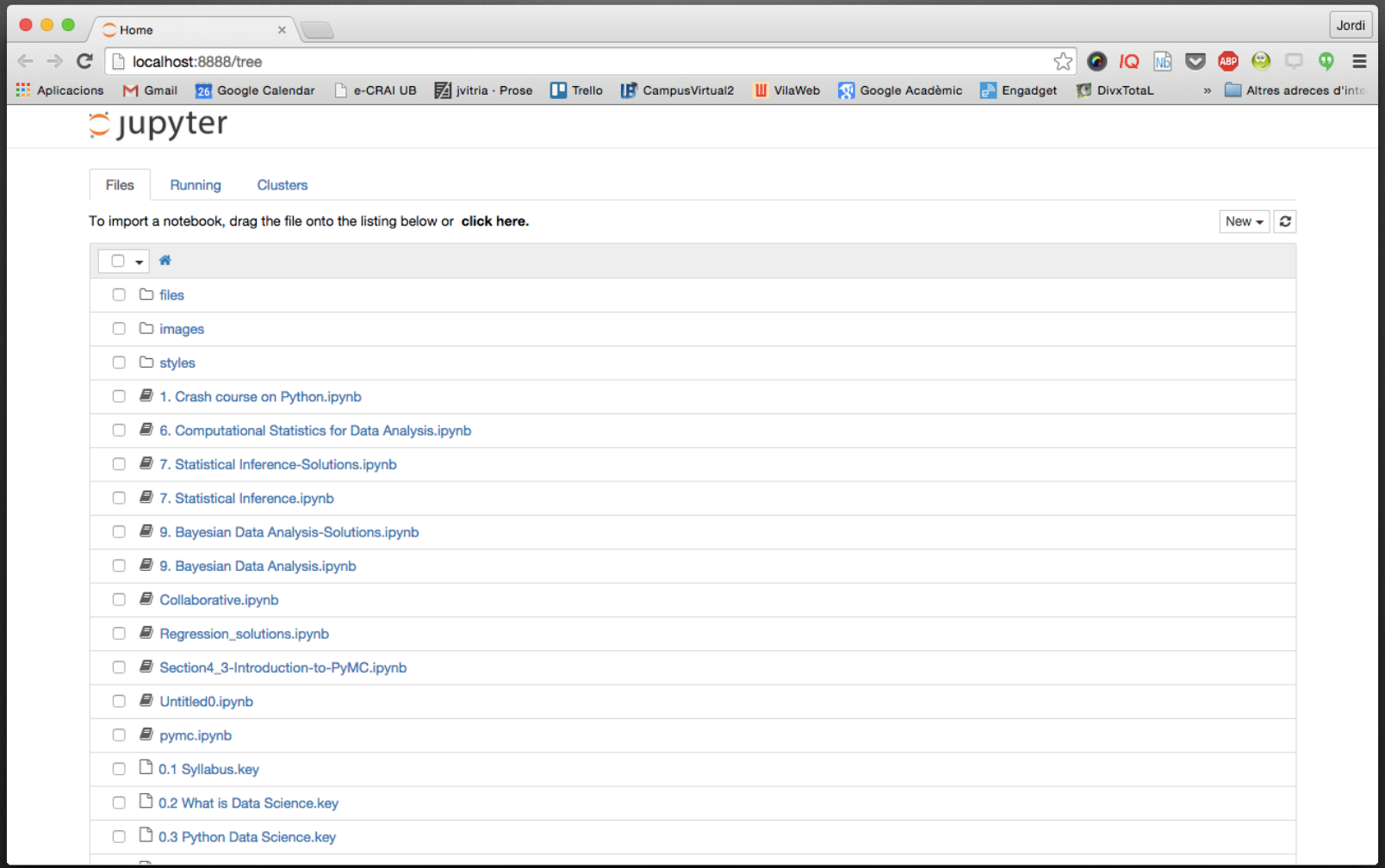
What is IPython?

I is for interactive



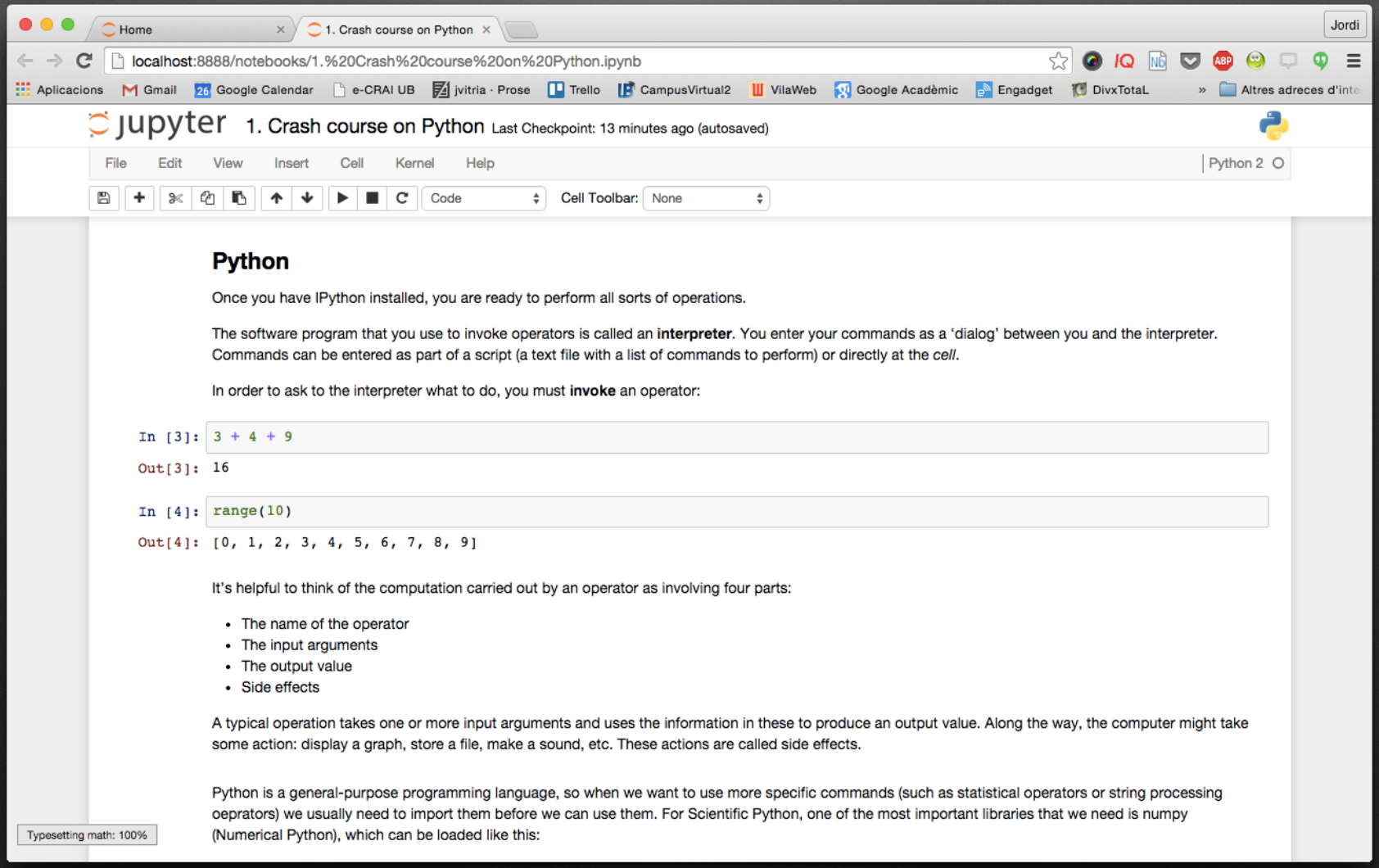
What is IPython?

I is for interactive



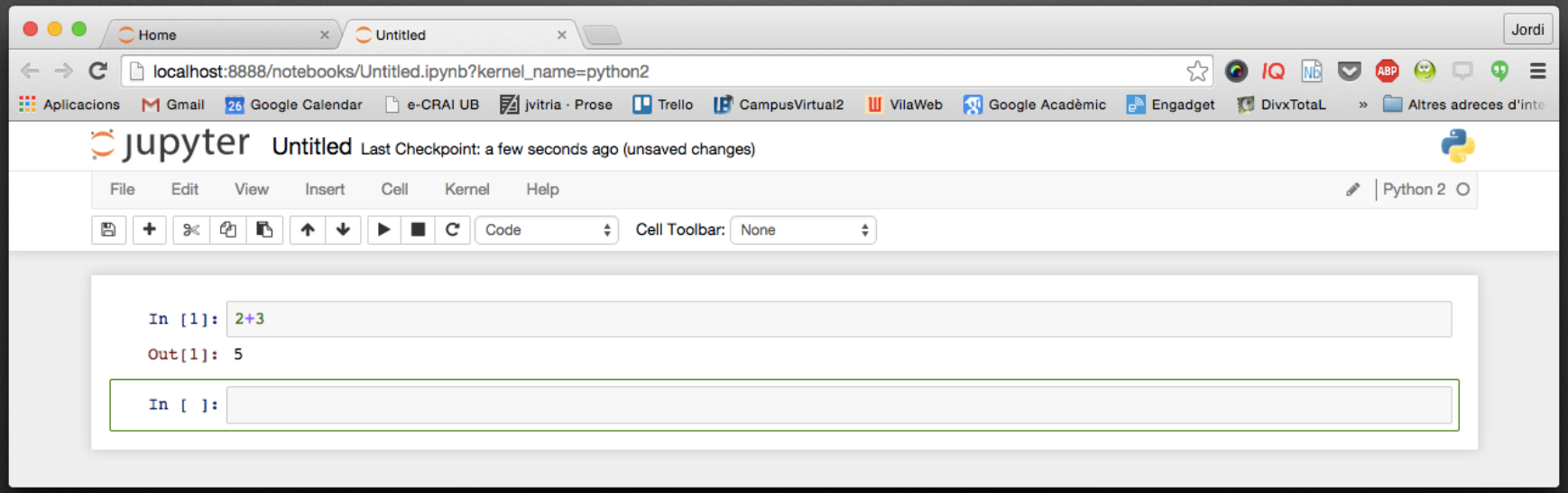
What is IPython?

I is for interactive



What is IPython?

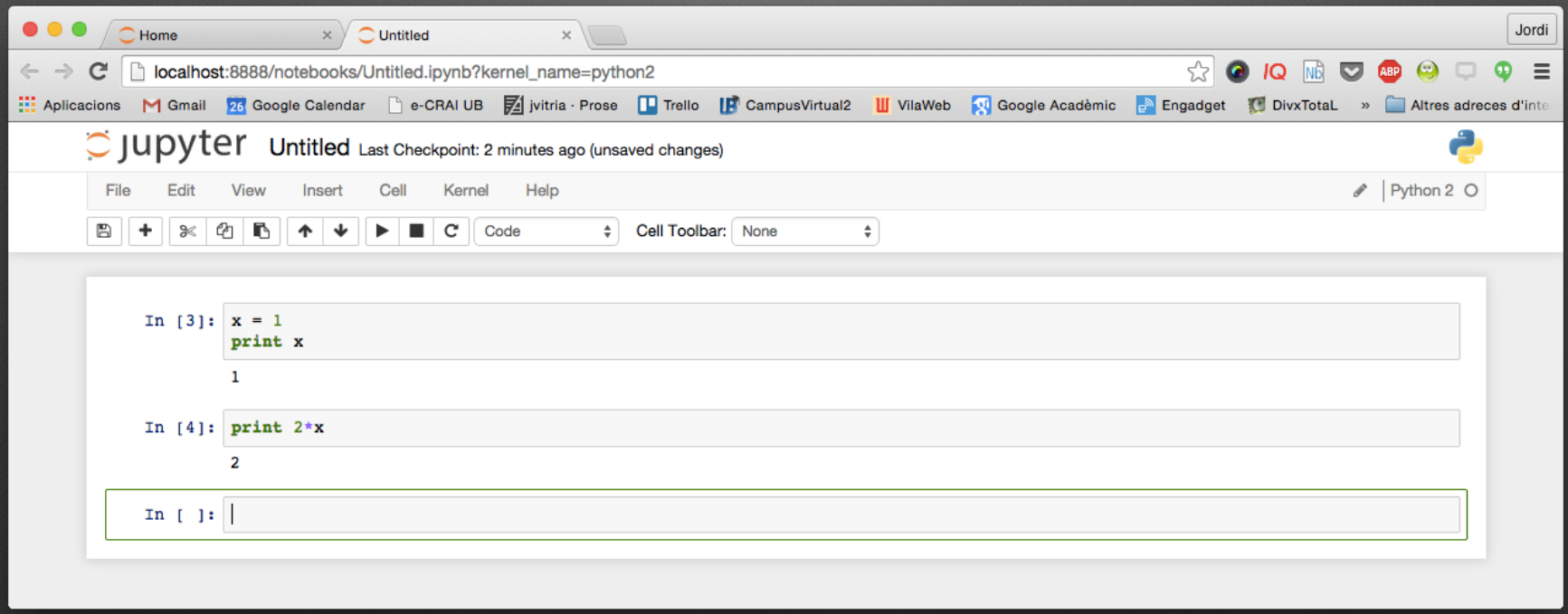
An IPython notebook is made up of a number of cells. Each cell can contain Python code.



You can execute a cell by clicking on it and pressing Shift-Enter. When you do so, the code in the cell will run, and the output of the cell will be displayed beneath the cell.

What is IPython?

Global variables are shared between cells. Executing the second cell thus gives the following result:



By convention, IPython notebooks are expected to be run from top to bottom. Failing to execute some cells or executing cells out of order can result in errors.

What is IPython?

After you have modified an IPython notebook for one of the assignments by modifying or executing some of its cells, remember to save your changes!