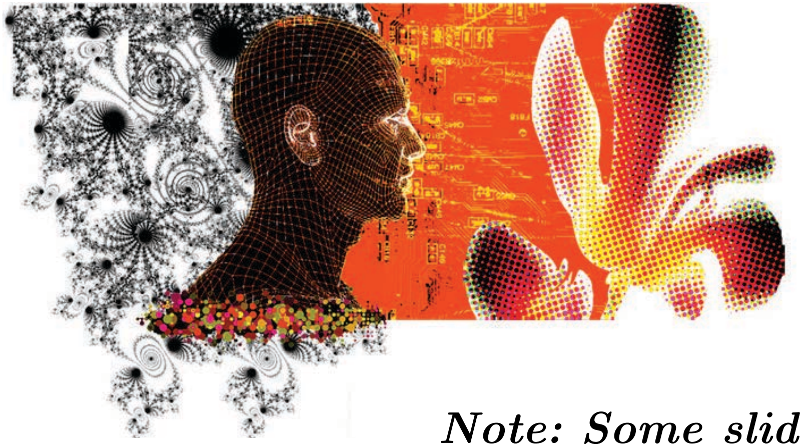
Data Science Training

November 2017

Python

Xavier Bresson

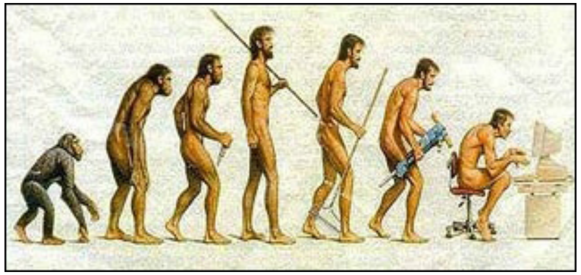
Data Science and AI Research Centre
NTU, Singapore



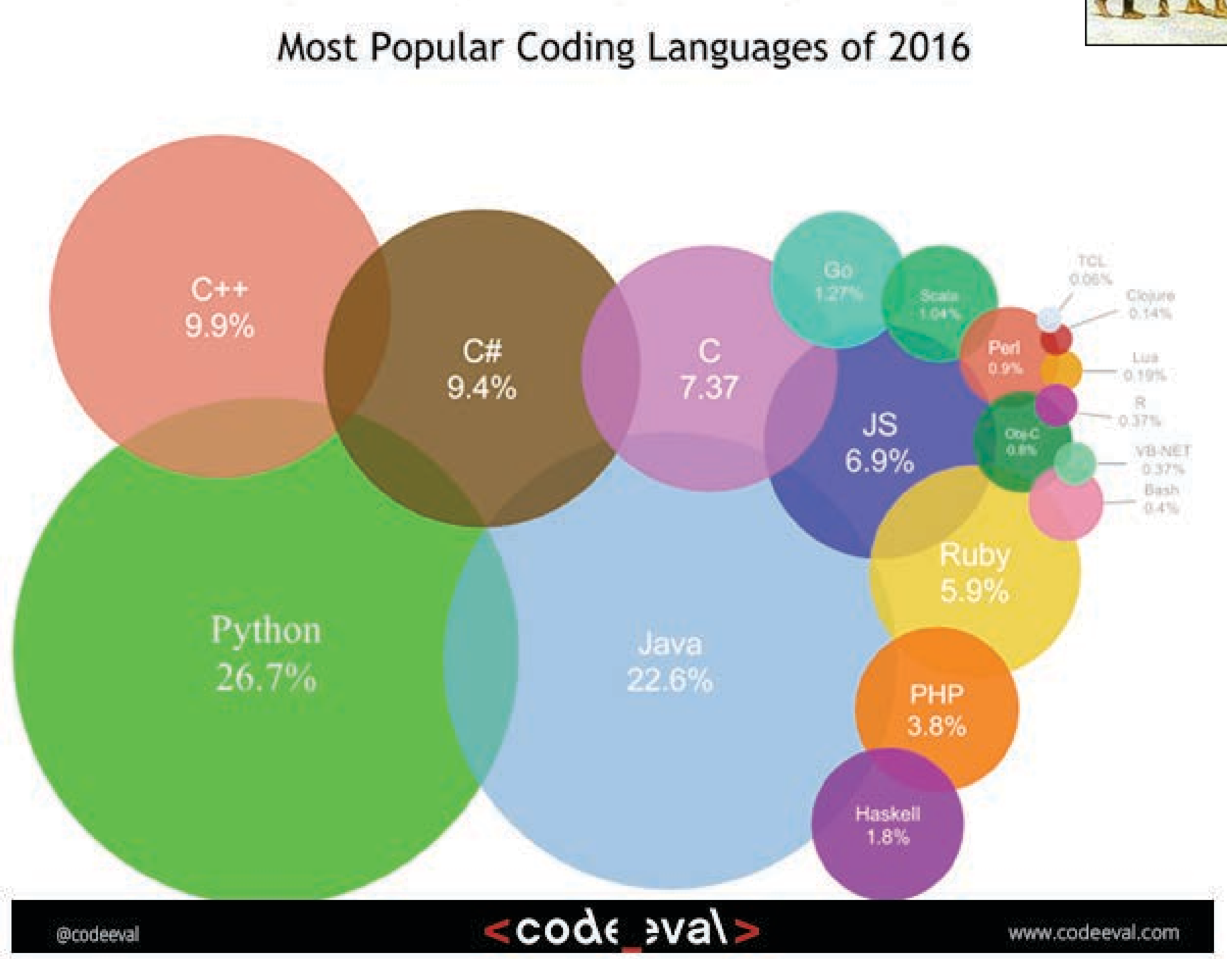
http://data-science-optum17.tk

Note: Some slides are from Defferrard's introduction to Python.

Python

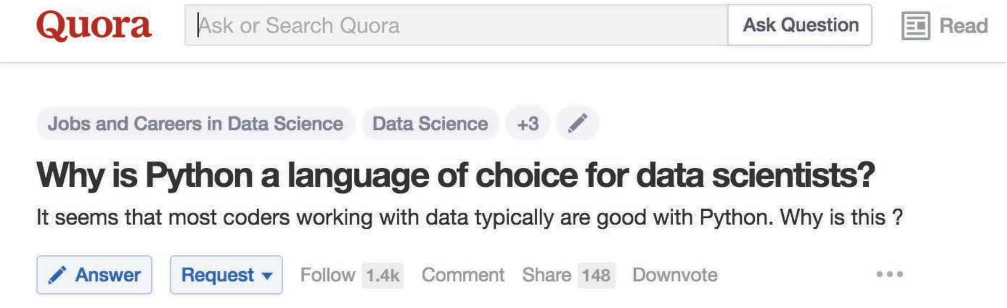


- ➢ **Why Python?**



Python

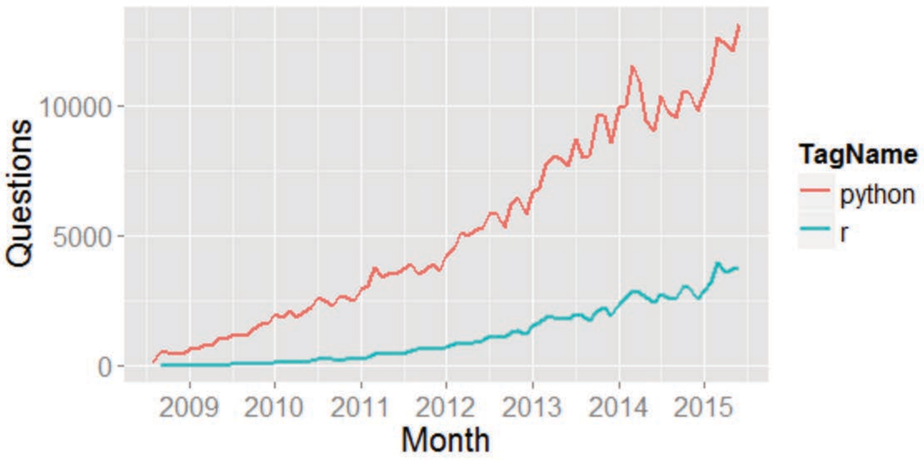
- ➢ Why Python for Data Science?



Why is Python a language of choice for data scientists?

It seems that most coders working with data typically are good with Python. Why is this ?

That's huge. But as Revolution Analytics' Andrie de Vries notes, the number of Stack Overflow questions about Python has grown to triple that of R questions.



- ➢ R, Matlab and Python all suitable for DS.
- ➢ **Matlab and R are specialized:** Matlab for linear algebra projects and R for statistical projects.
- ➢ **Python has a large ecosystem:** scientific computing, statistical tools, easy connection to SQL-type databases, fast data reports, development and production (e.g. Dropbox), etc.

What Computational Needs?

(1) Fast numerical mathematics: BLAS and LAPACK linear algebra libraries (matrix multiplications). Used by Matlab, R, Python.

(2) Easy data import: From cvs, matlab files, sql-type databases, internet (data scraping).

(3) Easy use to legacy codes (no re-coding): Inherite C, matlab, Fortran codes.

(4) Easy presentation of results: Quick output of results, also dynamic updates and html, pdf formats.

(5) Rapid prototyping: 10-min prototype (not easy in c/oop). Matlab, R, Python are good because script-languages.

(6) Use same framework for R&D and production: Python can be used both in development and production, not the case for R and Matlab.

(7) Benefit from parallel and distributed architectures: Cluster computing (multi-threads, MPI, OpenMP), GPU computing (OpenCL, CUDA).

Python Pros: Prototyping

(1) Easy-to-learn language: No need to declare variable types, to free memory, still easy to change type of objects dynamically.

(2) Elegant syntax: Quick to write codes, easy to read. No need to train for weeks to start, then experience integrates all technical details like data structures, etc.

(3) High-level data structures: List, tuple, set, dict & containers

(4) Multi-paradigm: Object-oriented, procedural, functional

(5) Dynamically typing: Helpers for your functions

(6) Automatic memory management: Garbage collector, no need to manage manually memory like c/oop.

(7) Interpreted language: Like Matlab, R, no need to compile, direct execution, good for prototype/development (if not using "for" loops!).

(8) OS independent: Windows, Mac, Linux. Work remotely (cloud) using only a web browser!

(9) Large community: Forums, tutorials – just google your question!

(10) Extensive ecosystem of libraries: scientific computing, statistics, web carwling, etc

(11) Easy to share & install packages via pip: one line command

Python Pros: Production

(1) General purpose language: Historical approach: scientists develop codes in Matlab, R and gave it to engineering team. Python can be directly integrated inside a big system because of it bends easily to other codes.

(2) Encourage code re-use: Python is modulable and packages

(3) Integrated documentation

(4) Open-source: No need to pay for libraries

(5) Many tools for production: Python has developed standard production modules: Unit & integration testing, documentation generation, debugging, performance optimization

Python Cons

(1) Two Python languages: Python2 vs Python3 (not compatible). Historical Python2 will not be supported after 2020. Many people still use Python2, and many projects coded in Python2. Notebook can easily switch python version.

(2) Slow execution (if badly coded): Interpreted language (pre-compiled) so codes with "for" loops for example can be slow. Use

(2a) Specialized libraries for scientific computing: numpy, scipy (exploit vectorized architecture)

(2b) Compile codes with compilers pypy, numba, jython (python compiler for java)

(3) No compiler: Need to run codes to catch errors.

Scientific Python

- Python is a **general programming language**, not initially developed for DS and scientific computing. The scientific community got interested and develop libraries for scientific computing. As Python only uses objects (everything is an object), then they developed number libraries efficiently implemented for array objects.
- **Numerical analysis libraries:**
  - *numpy:* Multidimensional arrays, linear algebra
  - *scipy:* Higher-level data analysis algorithms for optimization, interpolation, signal processing, sparse matrices, decompositions
- **Specialized libraries:**
  - *scikit-learn:* machine learning algorithms
  - *scikit-image:* image processing algorithms
- **Deep Learning libraries**: *tensorflow, pytorch, keras*
- **Statistics library:** *pandas*
- **Symbolic algebra library:** *sympy*
- **Visualization libraries:**
  - *matplotlib:* similar to MATLAB plots
  - *bokeh:* recent interactive visualization library

Data Storage

- In DS, interaction with data is essential. Python have bridges to most data structures:

(1) Flat files

- *CSV:* numpy / pandas
- *Matlab:* scipy
- *JSON:* std lib
- *HDF5:* h5pyBasic relational database storage

(2) Connectors for relational databases

- *SQLite:* std lib
- *PostgreSQL:* psycopg (DB API)
- *MySQL:* mysqlclient
- *Oracle:* cx_Oracle (DB API)
- *Microsoft SQL Server:* pypyodbc (DB API)

(3) NoSQL data stores

- *Redis:* Redis-py
- *MongoDB:* PyMongo (MongoEngine)
- *Hbase:* HappyBase
- *Cassandra:* Datastax

(4) Object-Relational Mapping (ORM)

- *SQLAlchemy, Peewee, Pony*

Python IDE: Jupyter Notebook

- **Languages with integrated development environment (IDE):** Matlab, R, Java, C/oop.
- **Python IDE:** Jupyter is a html-based notebook environment. The interface runs on any internet browser, but the codes run either on the laptop or remotely s.a. on the cloud. Huge benefit!
- **OS independent:** Windows, Mac, Linux
- **Future of scientific computing:** All-in-one approach: results, paper, figures, codes, data, highly interactive, easy improvement of techniques through collaboration
- **Most adapted language for prototyping/data exploration:** Easy conversion to Python modules when mature for production
- **Large community of developers:** w/ *github* to share codes and *nbviewer* to share notebook results
- **Cons:** Not yet able to explore data values like Matlab with its workspace
- **Alternate scientific IDEs:** Spyder, Rodeo
- **Other IDEs:** IDLE, PyCharm

Install Python Yourself

- ***Windows:*** Anaconda
- ***Mac:*** Anaconda or homebrew/pip
- ***Linux:*** Package manager

⇒ See **below**

- ***Cloud IDE:*** Amazon Web Services (AWS)

Live Session

- ***Notebook:*** Jupyter
- ***Basics of Scientific Python:*** numpy, scipy, scikit-learn, matplotlib

⇒ Run **code01.ipynb**

Questions?

Data Science Training

November 2017

Setup Python on Your Computer

Xavier Bresson

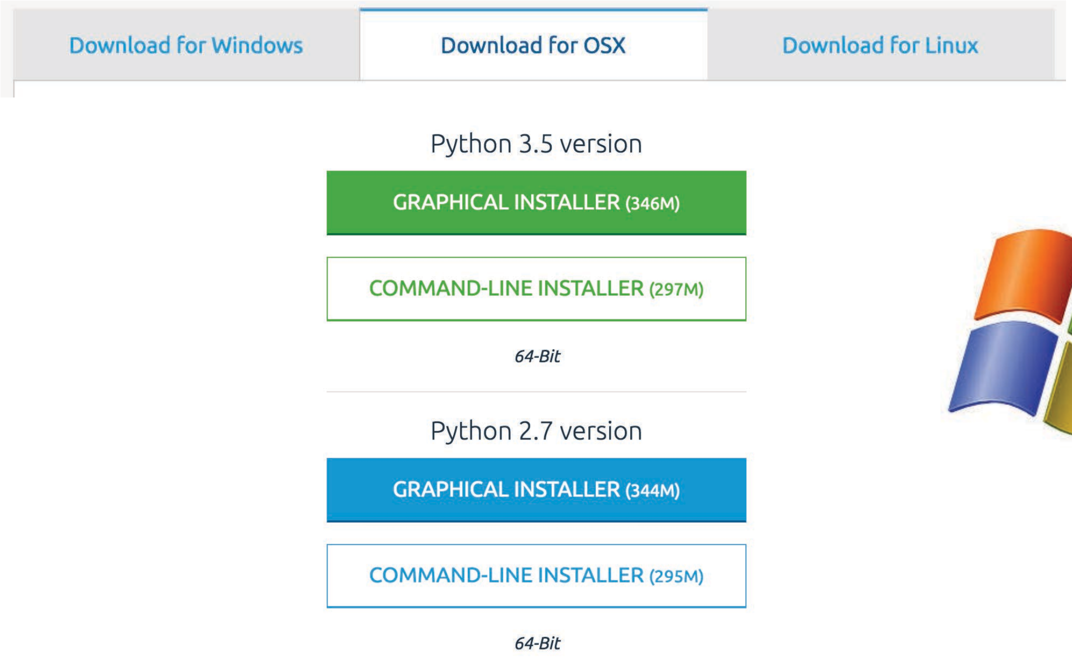
Data Science and AI Research Centre
NTU, Singapore

http://data-science-training-xb.com

Anaconda

- **Anaconda:**
  - Pre-installed 100 Python libraries for all OS (Windows, Mac, Linux)
  - Anaconda2 for Python 2.x and Anaconda3 for Python 3.x

- **Download it:** https://www.continuum.io/downloads



Download for Windows | Download for OSX | Download for Linux

Python 3.5 version

GRAPHICAL INSTALLER (346M)

COMMAND-LINE INSTALLER (297M)

64-Bit

Python 2.7 version

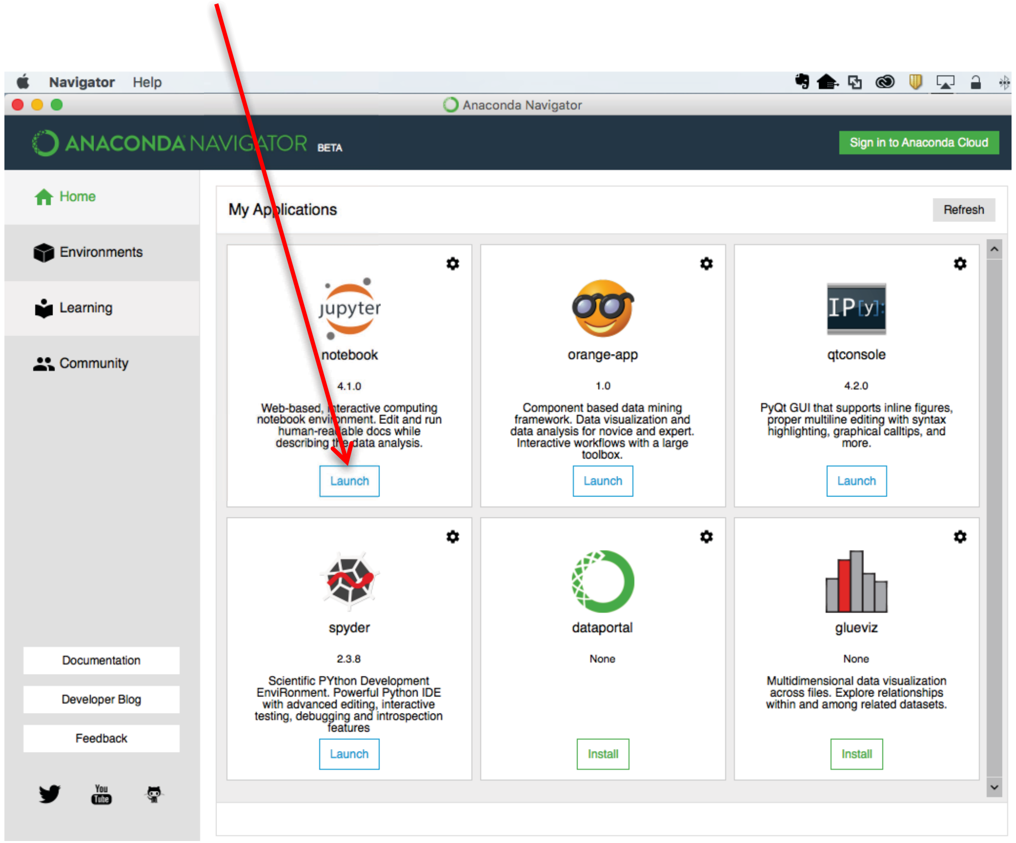
GRAPHICAL INSTALLER (344M)

COMMAND-LINE INSTALLER (295M)

64-Bit

Anaconda

- Open Python Notebook:



notebook
4.1.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

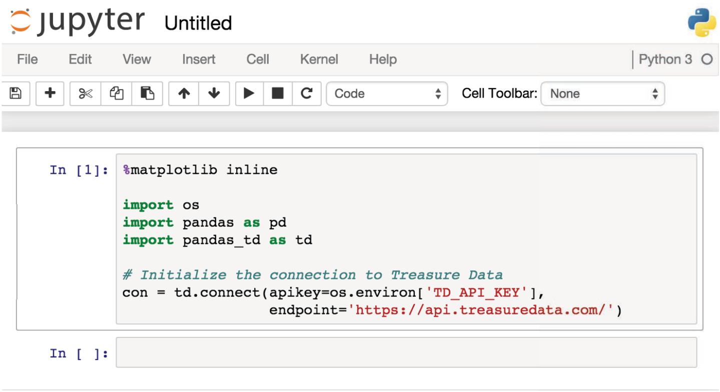
orange-app
1.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Launch

qtconsole
4.2.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch

spyder
2.3.8
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch

dataportal
None
Install

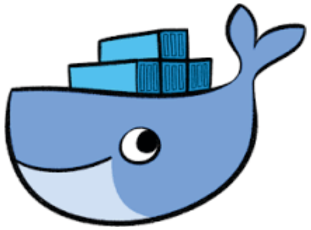
glueviz
None
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install



```
In [1]: %matplotlib inline

        import os
        import pandas as pd
        import pandas_td as td

        # Initialize the connection to Treasure Data
        con = td.connect(apikey=os.environ['TD_API_KEY'],
                         endpoint='https://api.treasuredata.com/')

In [ ]:
```

Docker

- **Docker:**
  - Pre-installed systems for all OS (Windows, Mac, Linux)
  - Highly flexible to setup any library.



Docker for Windows

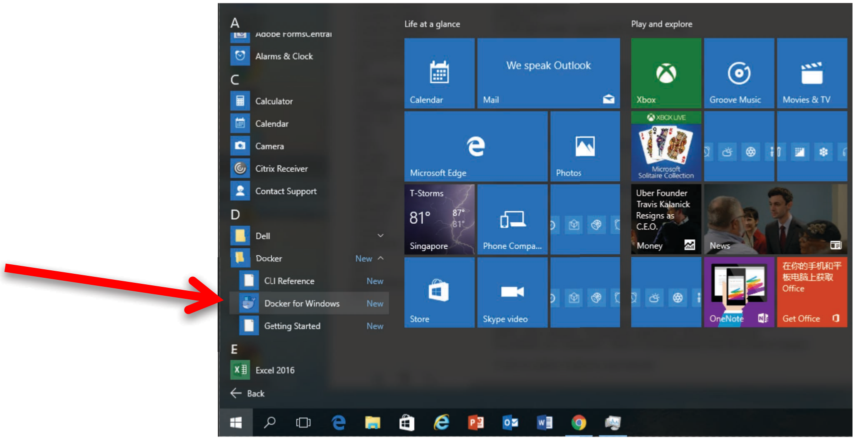
- **Windows 10:** Latest version of Docker
  https://download.docker.com/win/stable/InstallDocker.msi

- **Windows <10:** Install Docker Toolbox
  https://www.docker.com/products/docker-toolbox

- **Note:** If Hyper-V feature is not enabled
  ⇒ Docker will ask you to enable it automatically (it takes a few minutes to restart windows).
  ⇒ Enable Hyper-V Virtualization in BIOS (for Dell, restart windows and press "F2" when the first screen appears).

Docker for Windows

- **Open App:**



- **Open Terminal** with **cmd.exe**, and run an instance:

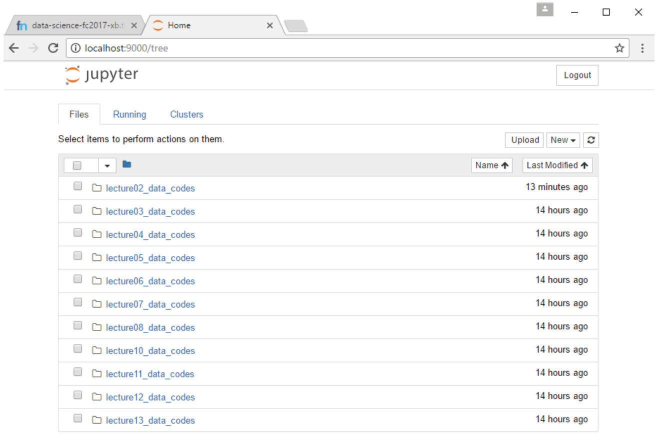
  docker run -d -p 9000:8888 docker_ID/container_ID

  Note: check running instances: **docker images**

  check containers: **docker ps -a**

- **Open Internet browser:** Type

  http://localhost:9000



Select items to perform actions on them.

Name	Last Modified
lecture02_data_codes	13 minutes ago
lecture03_data_codes	14 hours ago
lecture04_data_codes	14 hours ago
lecture05_data_codes	14 hours ago
lecture06_data_codes	14 hours ago
lecture07_data_codes	14 hours ago
lecture08_data_codes	14 hours ago
lecture10_data_codes	14 hours ago
lecture11_data_codes	14 hours ago
lecture12_data_codes	14 hours ago
lecture13_data_codes	14 hours ago

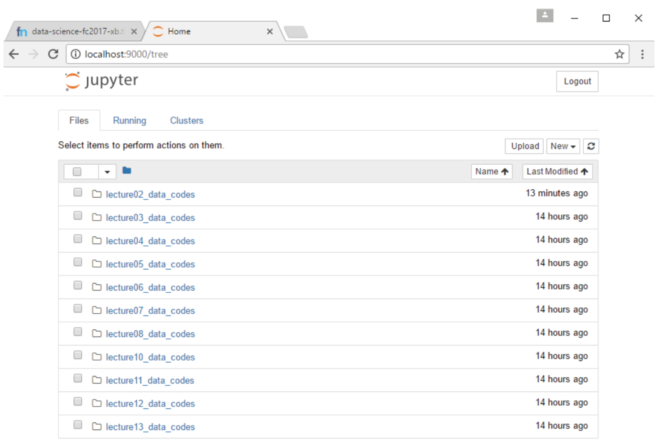
Docker for Mac, Linux

- **Mac:** Latest version of Docker
  https://www.docker.com

- **Linux:** Latest version of Docker
  sudo apt-get install docker.io

- **Open Terminal,** and run an instance:
  ```
  docker run -d -p 9000:8888 docker_ID/container_ID
  ```
  Note: check running instances: `docker images`
  check containers: `docker ps -a`

- **Open Internet browser:** Type
  http://0.0.0.0:9000

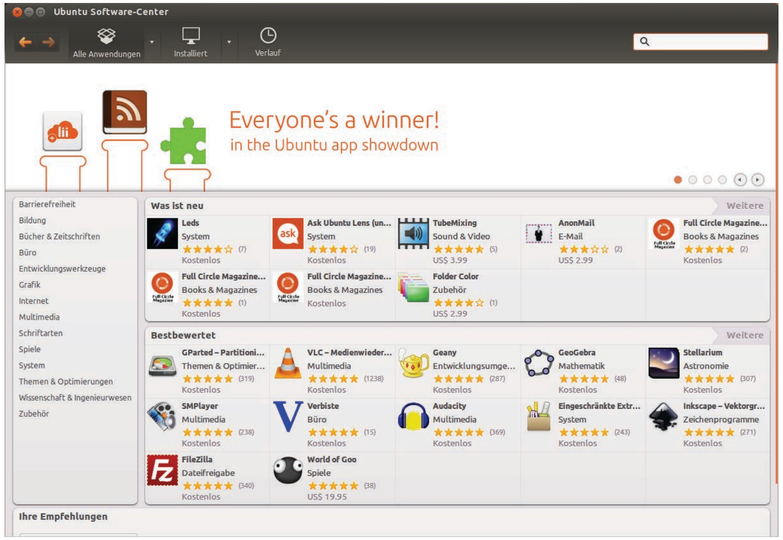


Select items to perform actions on them.

Name	Last Modified
lecture02_data_codes	13 minutes ago
lecture03_data_codes	14 hours ago
lecture04_data_codes	14 hours ago
lecture05_data_codes	14 hours ago
lecture06_data_codes	14 hours ago
lecture07_data_codes	14 hours ago
lecture08_data_codes	14 hours ago
lecture10_data_codes	14 hours ago
lecture11_data_codes	14 hours ago
lecture12_data_codes	14 hours ago
lecture13_data_codes	14 hours ago

Linux

(1) Use Linux package manager to install Python:



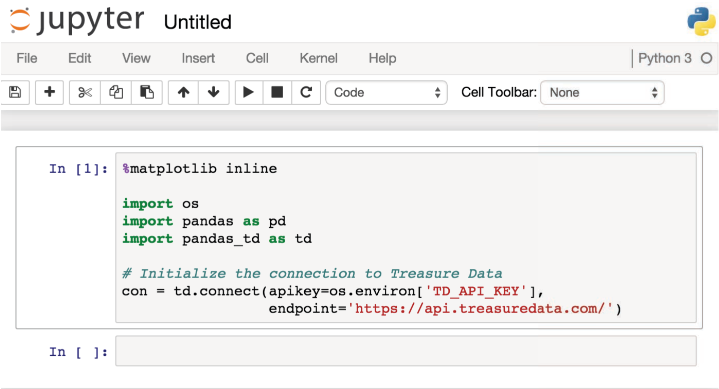
Ubuntu Software-Center
Alle Anwendungen
Installiert
Verlauf
Everyone's a winner!
in the Ubuntu app showdown
Barrierefreiheit
Bildung
Bücher & Zeitschriften
Büro
Entwicklungswerkzeuge
Grafik
Internet
Multimedia
Schriftarten
Spiele
System
Themen & Optimierungen
Wissenschaft & Ingenieurwesen
Zubehör
Was ist neu
Weitere
Leds System (7) Kostenlos
Ask Ubuntu Lens (un... System (19) Kostenlos
TubeMixing Sound & Video (5) US$ 3.99
AnonMail E-Mail (2) US$ 2.99
Full Circle Magazine... Books & Magazines (2) Kostenlos
Full Circle Magazine... Books & Magazines (1) Kostenlos
Full Circle Magazine... Books & Magazines Kostenlos
Folder Color Zubehör (1) US$ 2.99
Bestbewertet
Weitere
GParted – Partitioni... Themen & Optimier... (319) Kostenlos
VLC – Medienwieder... Multimedia (1238) Kostenlos
Geany Entwicklungsumge... (287) Kostenlos
GeoGebra Mathematik (48) Kostenlos
Stellarium Astronomie (307) Kostenlos
SMPlayer Multimedia (238) Kostenlos
Verbiste Büro (15) Kostenlos
Audacity Multimedia (369) Kostenlos
Eingeschränkte Extr... System (243) Kostenlos
Inkscape – Vektorgr... Zeichenprogramme (271) Kostenlos
FileZilla Dateifreigabe (340) Kostenlos
World of Goo Spiele (38) US$ 19.95
Ihre Empfehlungen

(2) Setup virtual environment:

```
mkdir ~/pyvenv/my_virtual_env

pyvenv ~/pyvenv/my_virtual_env

. ~/pyvenv/my_virtual_env/bin/activate

pip (or pip3) install numpy scipy scikit-learn
matplotlib jupyter ipython
```

(3) Run Python Notebook:

```
jupyter notebook
```



```
In [1]: %matplotlib inline

        import os
        import pandas as pd
        import pandas_td as td

        # Initialize the connection to Treasure Data
        con = td.connect(apikey=os.environ['TD_API_KEY'],
                         endpoint='https://api.treasuredata.com/')

In [ ]:
```

Mac

(1) Homebrew:

Open terminal, type:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/
Homebrew/install/master/install)"
```

(2) Pip:

Open terminal, type:

```
brew install git
git config --global user.name "your name"
git config --global user.email your.name@me.com
```

Mac

(3) Pyenv:

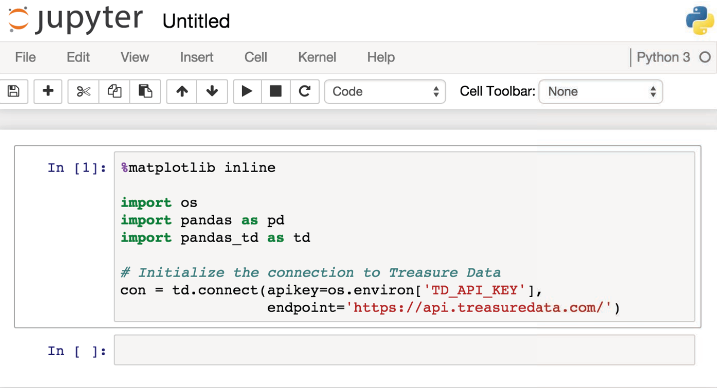
Open terminal, type:

(3a) Installation:

```
git clone https://github.com/yyuu/pyenv.git ~/.pyenv
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bash_profile
echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bash_profile
echo 'eval "$(pyenv init -)"' >> ~/.bash_profile
echo 'eval "$(pyenv virtualenv-init -)"' >> ~/.bash_profile
pyenv install 3.4.5
pyenv install 2.7.12
pyenv rehash
```

(3b) Create a new virtual env:

```
pyenv shell 3.4.5
pyenv virtualenv my_new_venv
# Activate
pyenv activate my_new_venv
# Install libraries:
pip install numpy scipy scikit-learn matplotlib jupyter ipython
# Run jupyter
jupyter notebook
```



```
In [1]: %matplotlib inline

        import os
        import pandas as pd
        import pandas_td as td

        # Initialize the connection to Treasure Data
        con = td.connect(apikey=os.environ['TD_API_KEY'],
                         endpoint='https://api.treasuredata.com/')

In [ ]:
```

Questions?