Jupyter Notebook
&

Anaconda

Outline

Anaconda

&

Jupyter

Should I use Python 2 or Python 3 for my development activity?

Contents

1. Should I use Python 2 or Python 3 for my development activity?
    1. What are the differences?
    2. Which version should I use?
    3. But wouldn't I want to avoid 2.x? It's an old language with many mistakes, and it took a major version to get them out.
    4. I want to use Python 3, but there's this tiny library I want to use that's Python 2.x only. Do I really have to revert to using Python 2 or give up on using that library?
    5. I decided to write something in 3.x but now someone wants to use it who only has 2.x. What do I do?
    6. Supporting Python 2 and Python 3 in a common code base
    7. Other resources that may help make the choice between Python 2 and Python 3
    8. Footnotes

What are the differences?

Short version: Python 2.x is legacy, Python 3.x is the present and future of the language

Should I use Python 2 or Python 3?

- It is hard to have all the version playing together
- We can solve it by using virtual environments (VirtualEnv)

Virtualenv

Mailing list | Issues | Github | PyPI | User IRC: #pypa Dev IRC: #pypa-dev

Introduction

`virtualenv` is a tool to create isolated Python environments.

https://virtualenv.pypa.io/en/stable/#

Anaconda for Data Science

- Is a distribution of libraries and software specifically build for Data Science

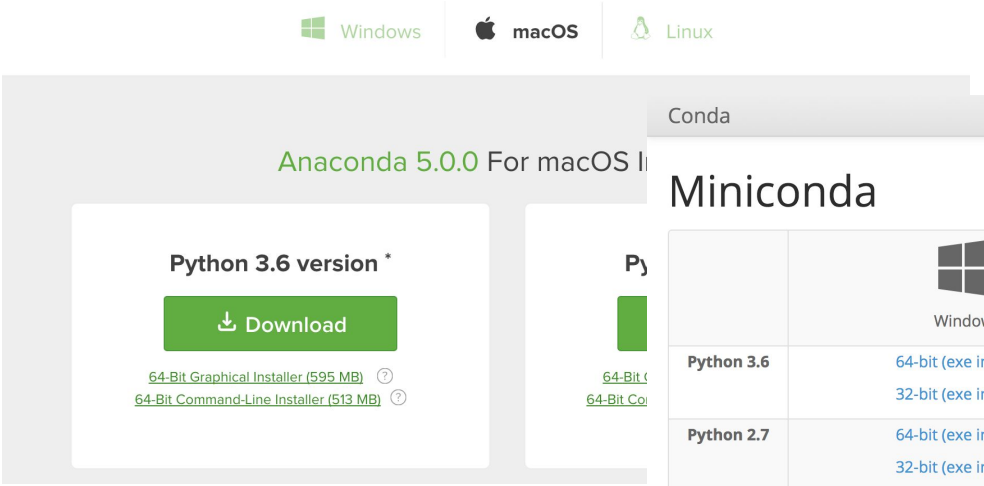
Conda

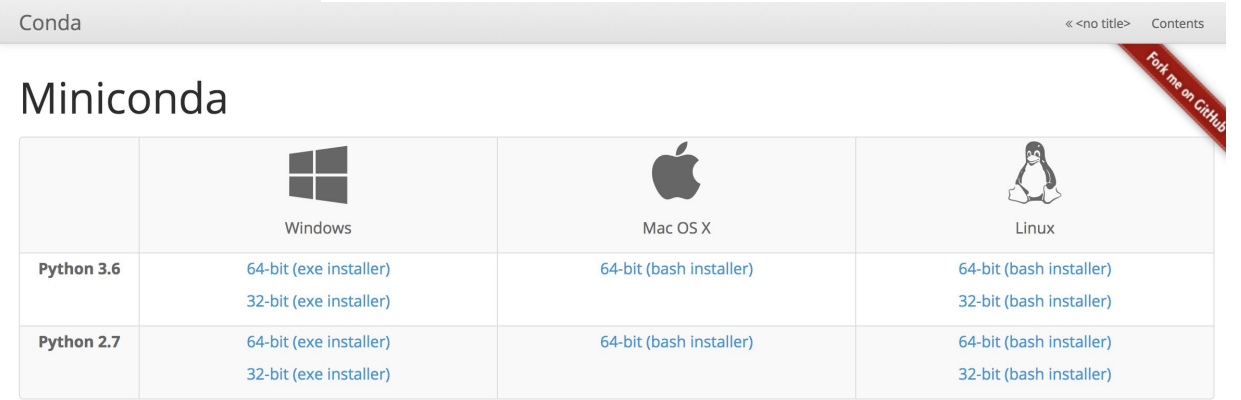
CONDA

Package, dependency and environment management for any language—Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, FORTRAN

Installation

- Anaconda contains over 150 packages and deps ~ 500MB
- As alternative you can use Miniconda (only conda and python)





Post Installation

- Remember to update your PATH: export PATH="/Users/username/anaconda/bin:$PATH"

$ conda upgrade conda

$ conda upgrade --all

Conda as a package manager

- You are probably familiar with pip
- Conda is similar except it focuses around Data Science
- For example in conda you can find Numpy, Scipy and Scikit-learn compiled with MKL library
- Packages are maintained by contributors

Managing Packages

- To install
  - $ conda install numpy scipy pandas
  - $ conda install numpy=1.10
- To remove
  - $ conda remove package_name
- To update
  - $ conda update package_name
- To update all
  - $ conda upgrade --all
- To search
  - $ conda search package_name

Conda as environments manager

- Envs allow you to separate and isolate the packager are you using for different projects
- This issue happening a lot when dealing with Python 2 / 3

Example

```
$ conda create --name tflow python=3

$ source activate tflow

$ conda install numpy pandas matplotlib jupyter

$ conda list
```

More environments actions

- Saving an Environment
  - $ conda env export > environment.yaml
- Loading an Environment
  - $ conda env create -f environment.yaml
- Listing Environments
  - $ conda env list
- Remove an Environment
  - $ conda env remove -n env_name
- Sharing packages for pip
  - $ pip freeze

How do I use Conda?

- Ideally, create a new environment for each project started
- This way you can keep libraries separated

Example

```
$ conda create --name tea_facts python=3

$ conda install numpy pandas matplotlib jupyter

$ conda list
```

Why Python 3

- Jupyter is switching to Python 3 only
- Python 2.7 is being retired
- Python 3.6 has great features such as formatted strings


jupyter
Lua
php
ERLANG
C++
Spark
C#
F#

What are Jupyter Notebooks?

- Web application that allows you to combine, text, code, equation and code
- Notebooks have quickly become ans essential tool for working with data (http://nbviewer.jupyter.org/)
- They supports different languages (kernels) https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

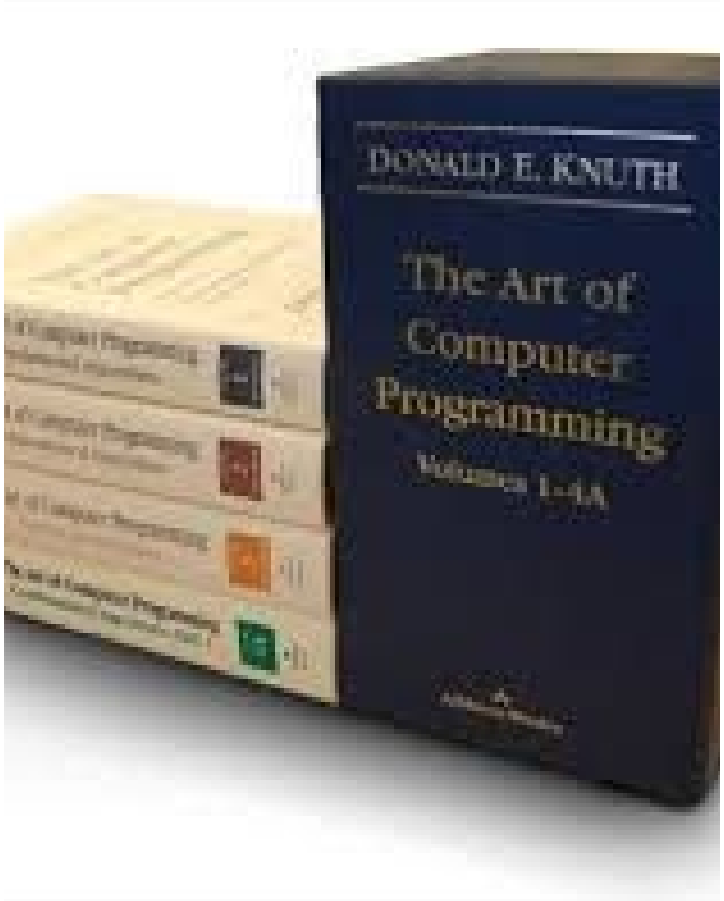
What are Jupyter Notebooks?

- Web application that allows you to combine, text, code, equation and code
- Notebooks have quickly become ans essential tool for working with data (http://nbviewer.jupyter.org/)
- They supports different languages (kernels) https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

Literate Programming

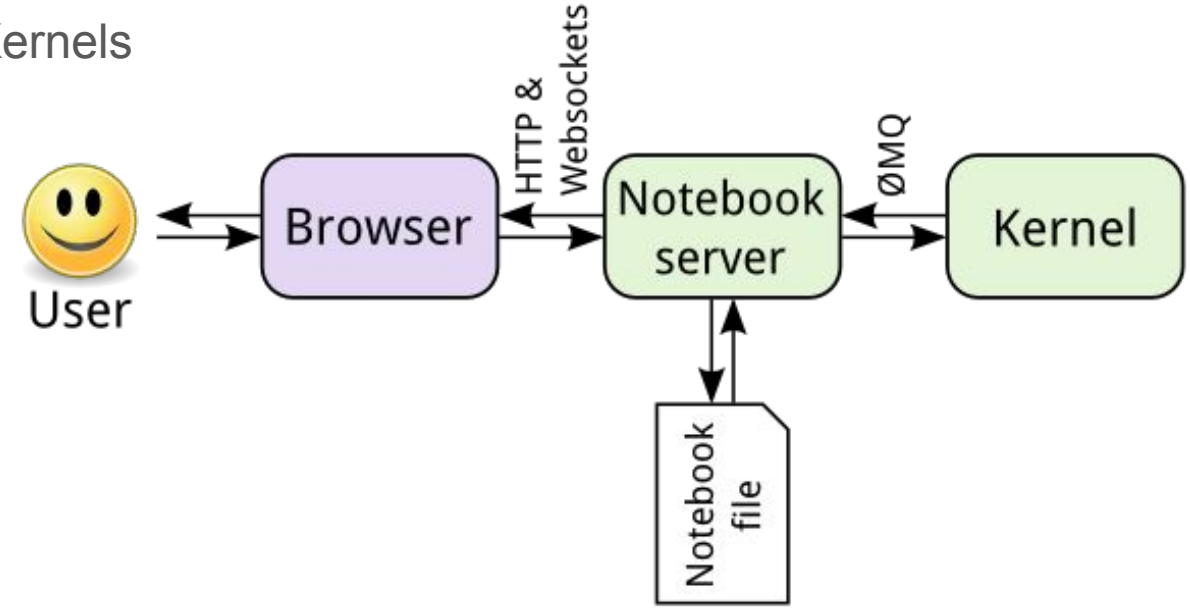
- A form of literate programming proposed donald Knuth in 1984

“ Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer do”



How Notebooks work

- **J**ulia **Py**thon **R** (Jupyter) notebooks grew out of the Ipython project by Fernando Perez
- It supports multiple Kernels
- Can run anywhere



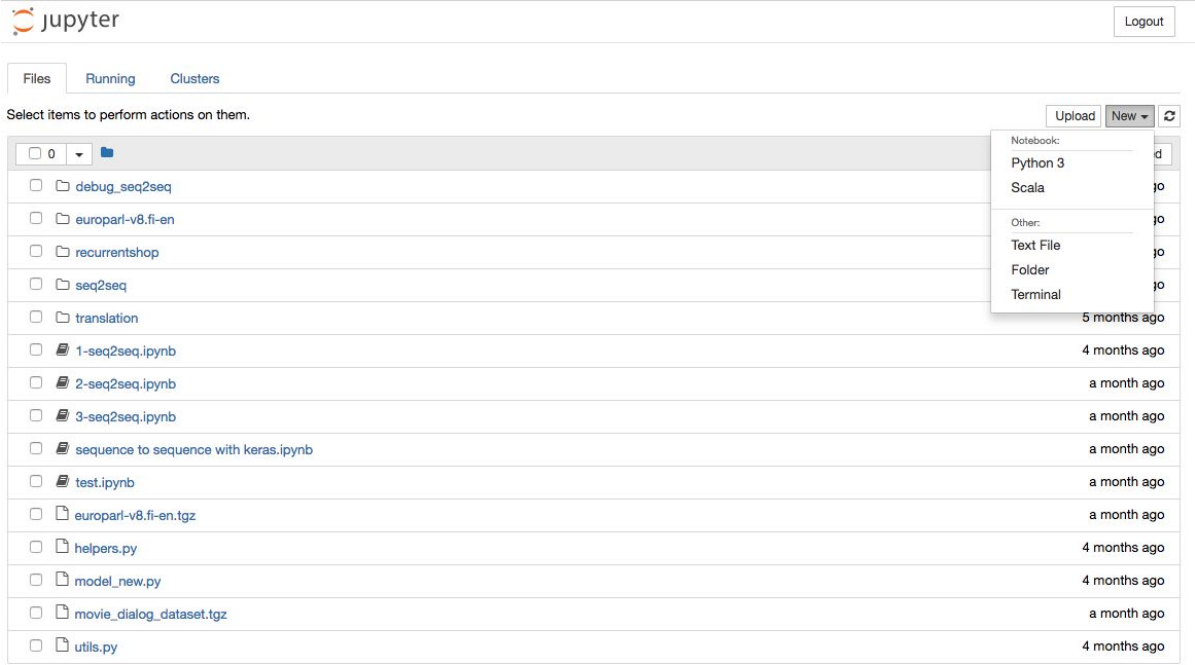
Install and Run Jupyter Notebook

$ conda install jupyter notebook

$ pip install jupyter notebook

To run the notebook server

$ jupyter notebook



jupyter

Logout

Files | Running | Clusters | Conda

15 Conda environments

Action	Name	Default?	Directory
	root		/Users/fabiofumarola/anaconda3
	cuckoo		/Users/fabiofumarola/anaconda3/envs/cuckoo
	demo_brfss		/Users/fabiofumarola/anaconda3/envs/demo_brfss
	dl3		/Users/fabiofumarola/anaconda3/envs/dl3
	dlnd-tf-lab		/Users/fabiofumarola/anaconda3/envs/dlnd-tf-lab
	flappybird		/Users/fabiofumarola/anaconda3/envs/flappybird

Available packages

Search...

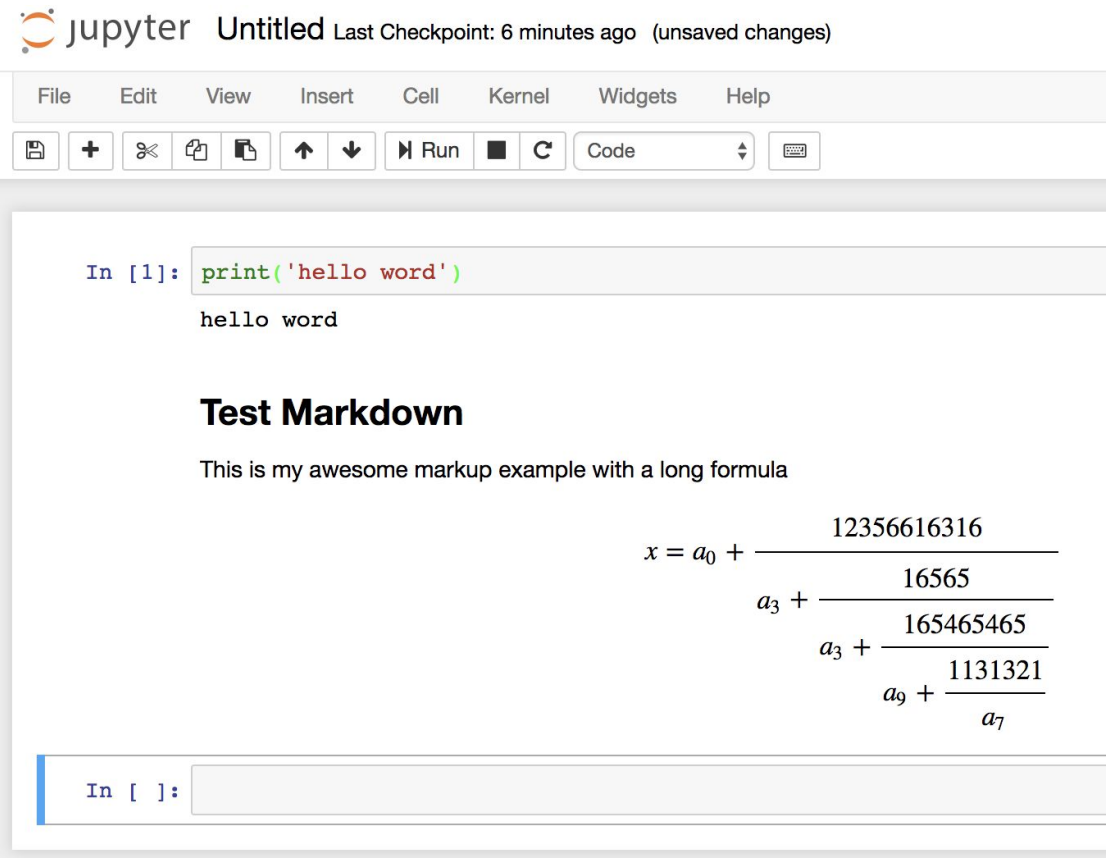
Jupyter nb_conda

124 installed packages in environment "tflow"

Name	Version	Build	Available
appnope	0.1.0	py35_0	
asn1crypto	0.22.0	py35_0	
backports	1.0	py35_1	
backports.functools_lru_cache	1.4	py35_1	
bkcharts	0.2	py35_0	
blas	1.1	openblas	

Notebook Interface

- Create different kind of cells
- Show shortcuts
- Command Palette



Notebook Tutorials

Clone the repository:

- Working with code cells
- Keyboard shortcut
- Notebook Magics

Converting Notebooks

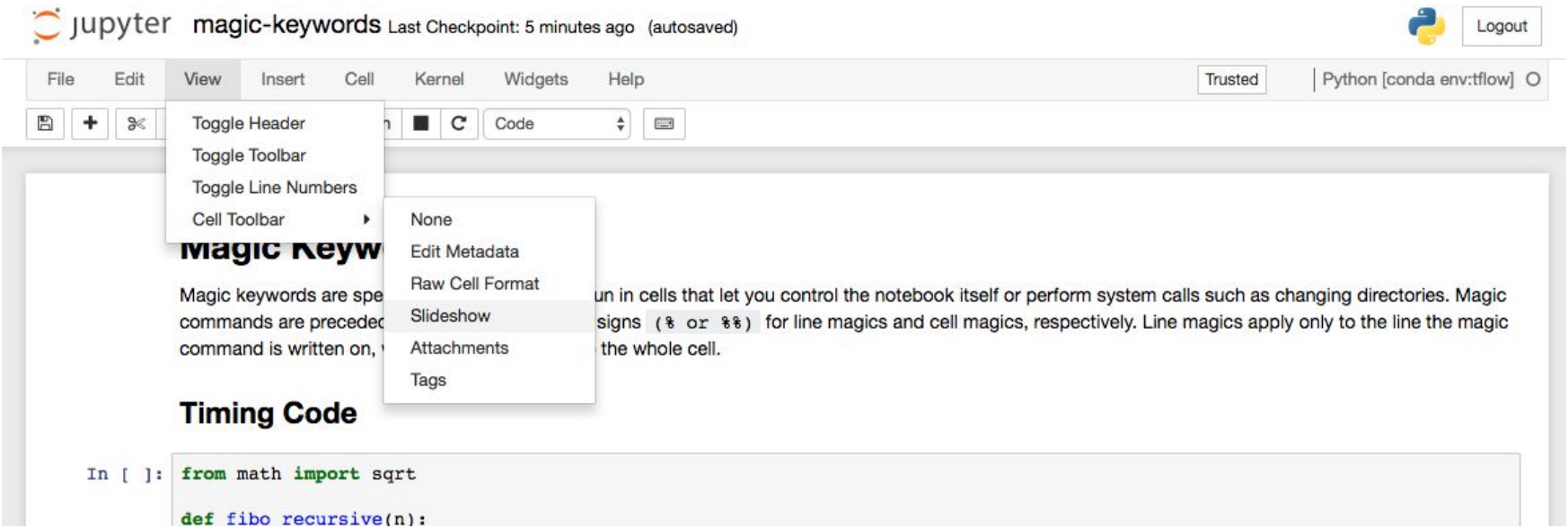
- Notebooks are just a big json file and can be converted into HTML, Markdown, Slideshows, etc.

For example to convert to an HTML file

- $ jupyter nbconvert --to html working-with-cells.ipynb

Creating a Slideshow

The slides are created by designating what cells are slides



Creating a Slideshow

Slide Type Slide

Magic Keywords

Magic keywords are special commands you can run in cells that let you control the notebook itself or perform system calls such as changing directories. Magic commands are preceded with one or two percent signs `(% or %%)` for line magics and cell magics, respectively. Line magics apply only to the line the magic command is written on, while cell magics apply to the whole cell.

Timing Code

]:

Slide Type Sub-Slide

```
from math import sqrt

def fibo_recursive(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    return fibo_recursive(n-1) + fibo_recursive(n - 2)

def fibo(n):
    return ((1+sqrt(5))**n-(1 - sqrt(5))**n)/(2**n*sqrt(5))
```

]:

Slide Type Sub-Slide

```
%timeit fibo_recursive(20)
```

Creating a Slideshow

- To convert the notebook file to slides

```
$ jupyter nbconvert magic-keywords-slide.ipynb --to slides
```

- To convert and run and HTTP Server

```
$ jupyter nbconvert magic-keywords-slide.ipynb --to slides --post serve
```