# LEARNING ipython

#ipython

# Table of Contents

# About

# Chapter 1: Getting started with ipython

## Remarks

IPython is a Read-Evaluate-Print Loop shell for interactive Python development. It supports interactive visualizations using GUI toolkits, and provides a kernel for Jupyter. It can also be embedded into other projects.

There are other similar REPL shells for Python, for example, ptpython and bpython.

## Versions

| Version | Release Date |
|---|---|
| 5.0.0 | 2016-07-07 |
| 4.2.0 | 2016-04-20 |
| 4.1.0 | 2016-02-02 |
| 4.0.0 | 2015-08-12 |
| 3.2.0 | 2015-06-21 |
| 3.1.0 | 2015-04-03 |
| 3.0.0 | 2015-02-27 |
| 2.4.0 | 2015-01-30 |
| 2.3.0 | 2014-10-01 |
| 2.2.0 | 2014-08-06 |
| 2.1.0 | 2014-05-21 |
| 2.0.0 | 2014-04-02 |
| 0.13 | 2012-06-30 |
| 0.12 | 2011-12-19 |
| 0.11 | 2011-07-31 |
| 0.10 | 2010-09-01 |
| 0.9 | 2008-09-13 |

## Examples

### Installation and Usage

Like the built-in `python` interactive shell, IPython is a REPL (Read-Evaluate-Print Loop) shell, with a variety of features that make it more pleasant to use for day-to-day Python development than the built-in REPL shell.

## Installation

To install it:

```
pip install ipython
```

Or, via Anaconda:

```
# To install into the active environment:
$ conda install ipython

# Or, to create a new environment with IPython installed:
$ conda create -n <env_name> ipython
```

Or, via Enthought Canopy:

```
$ enpkg ipython
```

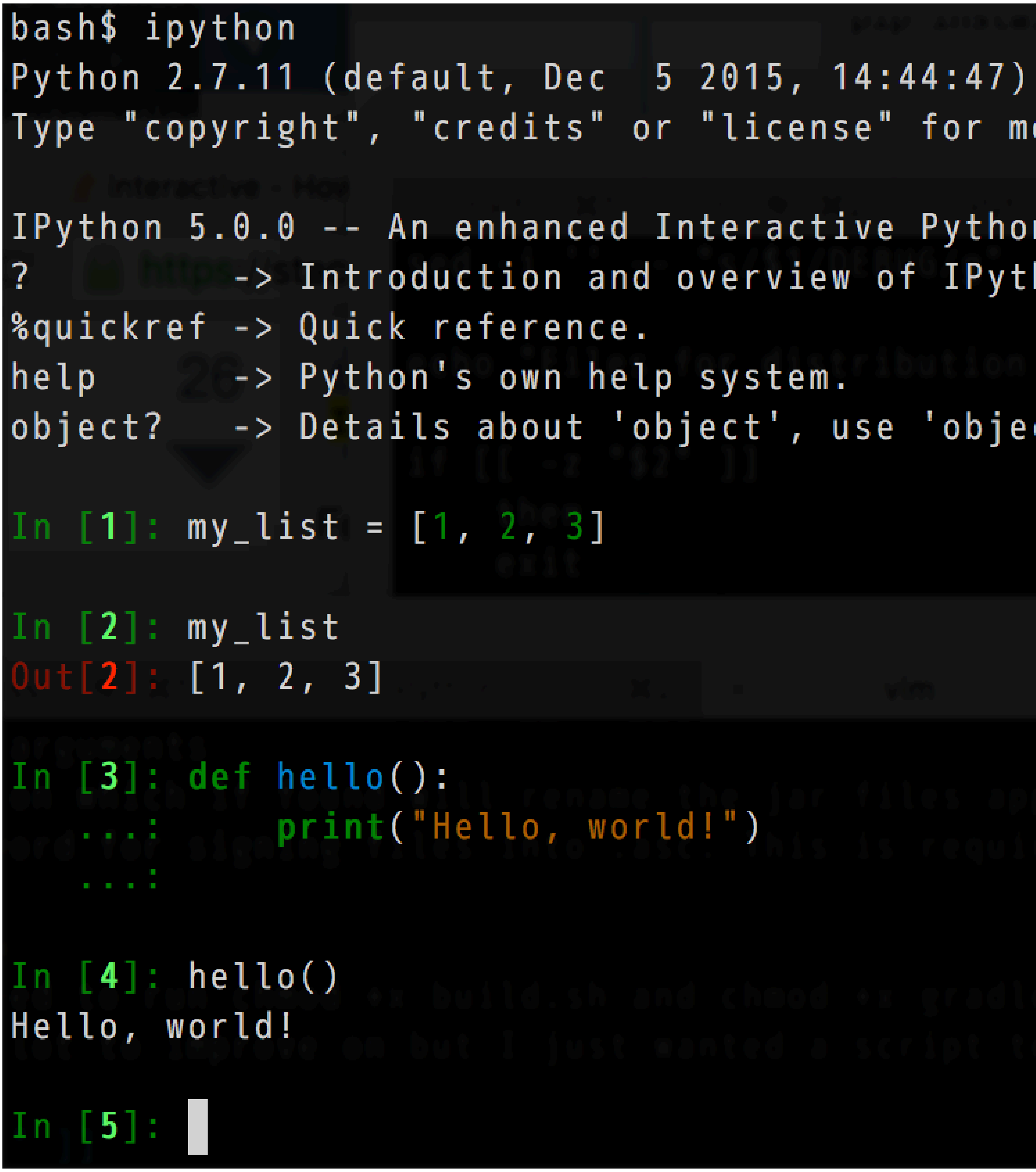
## Usage

After installation, run it using your default Python (2 or 3) using:

```
ipython
```

Or to use Python 3:

```
ipython3
```



```
bash$ ipython
Python 2.7.11 (default, Dec  5 2015, 14:44:47)
Type "copyright", "credits" or "license" for mo

IPython 5.0.0 -- An enhanced Interactive Pytho
?         -> Introduction and overview of IPyt
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'obje

In [1]: my_list = [1, 2, 3]

In [2]: my_list
Out[2]: [1, 2, 3]

In [3]: def hello():
   ...:     print("Hello, world!")
   ...:

In [4]: hello()
Hello, world!

In [5]:
```

## Getting Help

```
?
```

This gives you an introduction and overview of IPython's features.

```
object?
```

This lists all methods and fields of the object and its documentation (if it exists).

```
object??
```

Same as above, provides even more detail about the object, in particular will try to find and display the source code for the object.

```
object.<TAB Key>
```

`TAB`-completion that lists and iterates through available fields/methods of an object. Due to the dynamic nature of Python not all methods can be discovered this way. Also private methods (starting with _) will be hidden by default, insert a _ and press `TAB` again to display them.

```
%quickref
```

This displays a quick-reference for the IPython shell.

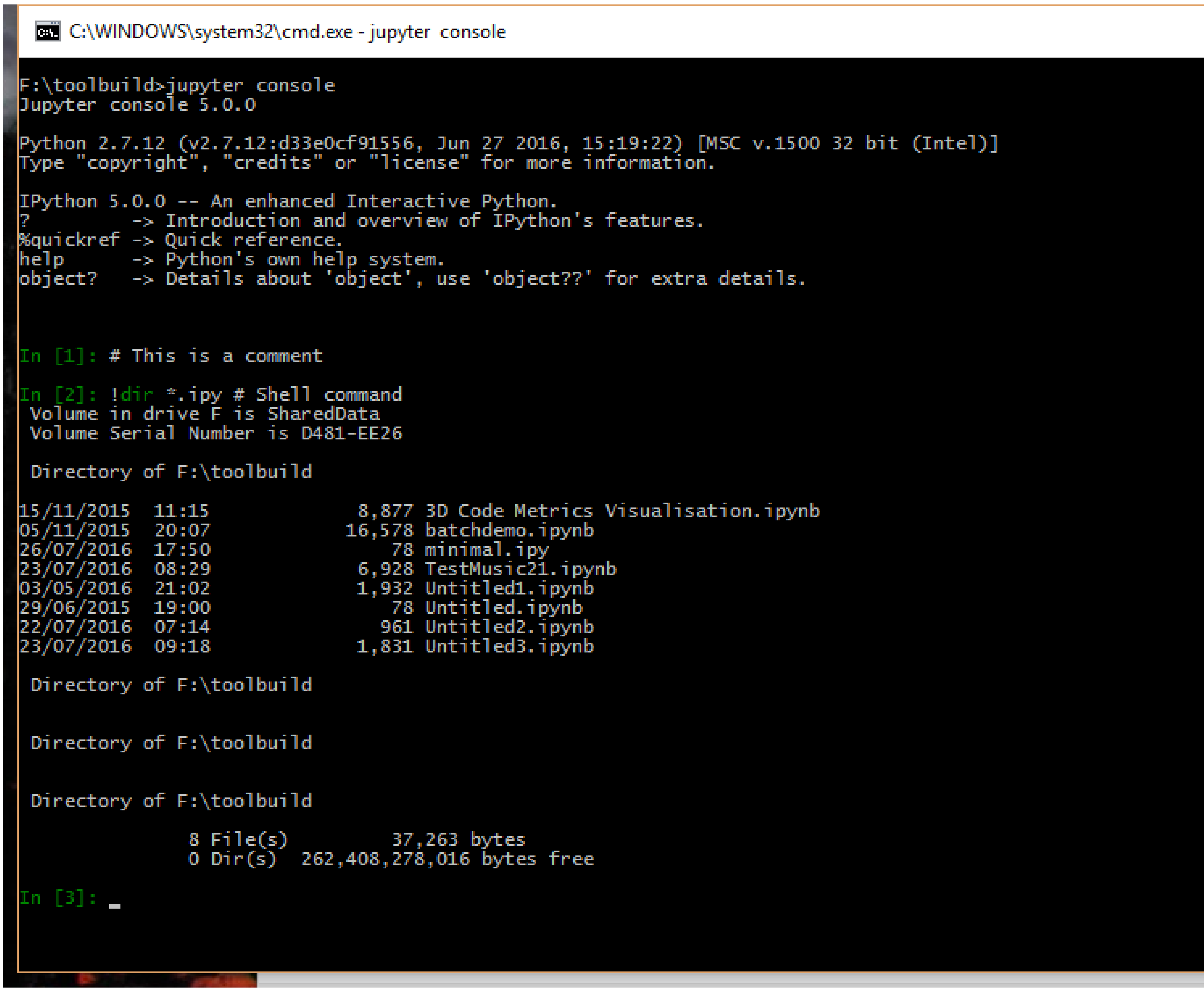
## IPython vs Jupyter

IPython has two parts to it: A command line interface that replaces the default `python` REPL and a way to run Python through the web browser as a graphical user interface.

With the latest developments the browser part has been split into the Jupyter project that enables multiple programming languages to use the graphical interface. It is still possible to use IPython as a Python kernel for this.

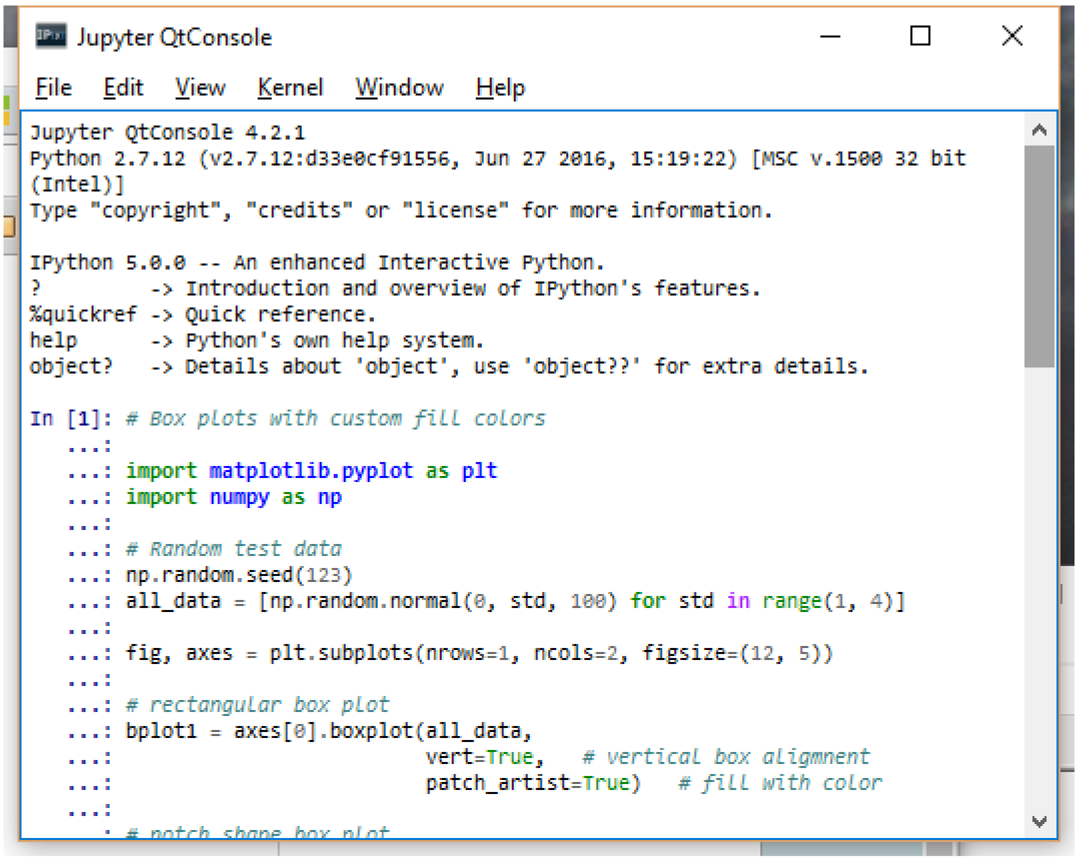
Up to date setup instructions for Jupyter can be found in the official install docs.

`ipython`, or `jupyter console`, when invoked from the command line without any *other* parameters will enter an interactive terminal session as below:



```
F:\toolbuild>jupyter console
Jupyter console 5.0.0

Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)]
Type "copyright", "credits" or "license" for more information.

IPython 5.0.0 -- 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]: # This is a comment

In [2]: !dir *.ipy # Shell command
 Volume in drive F is SharedData
 Volume Serial Number is D481-EE26

 Directory of F:\toolbuild

15/11/2015  11:15             8,877 3D Code Metrics Visualisation.ipynb
05/11/2015  20:07            16,578 batchdemo.ipynb
26/07/2016  17:50                78 minimal.ipy
23/07/2016  08:29             6,928 TestMusic21.ipynb
03/05/2016  21:02             1,932 Untitled1.ipynb
29/06/2015  19:00                78 Untitled.ipynb
22/07/2016  07:14               961 Untitled2.ipynb
23/07/2016  09:18             1,831 Untitled3.ipynb

 Directory of F:\toolbuild


 Directory of F:\toolbuild


 Directory of F:\toolbuild

               8 File(s)         37,263 bytes
               0 Dir(s)  262,408,278,016 bytes free

In [3]: _
```

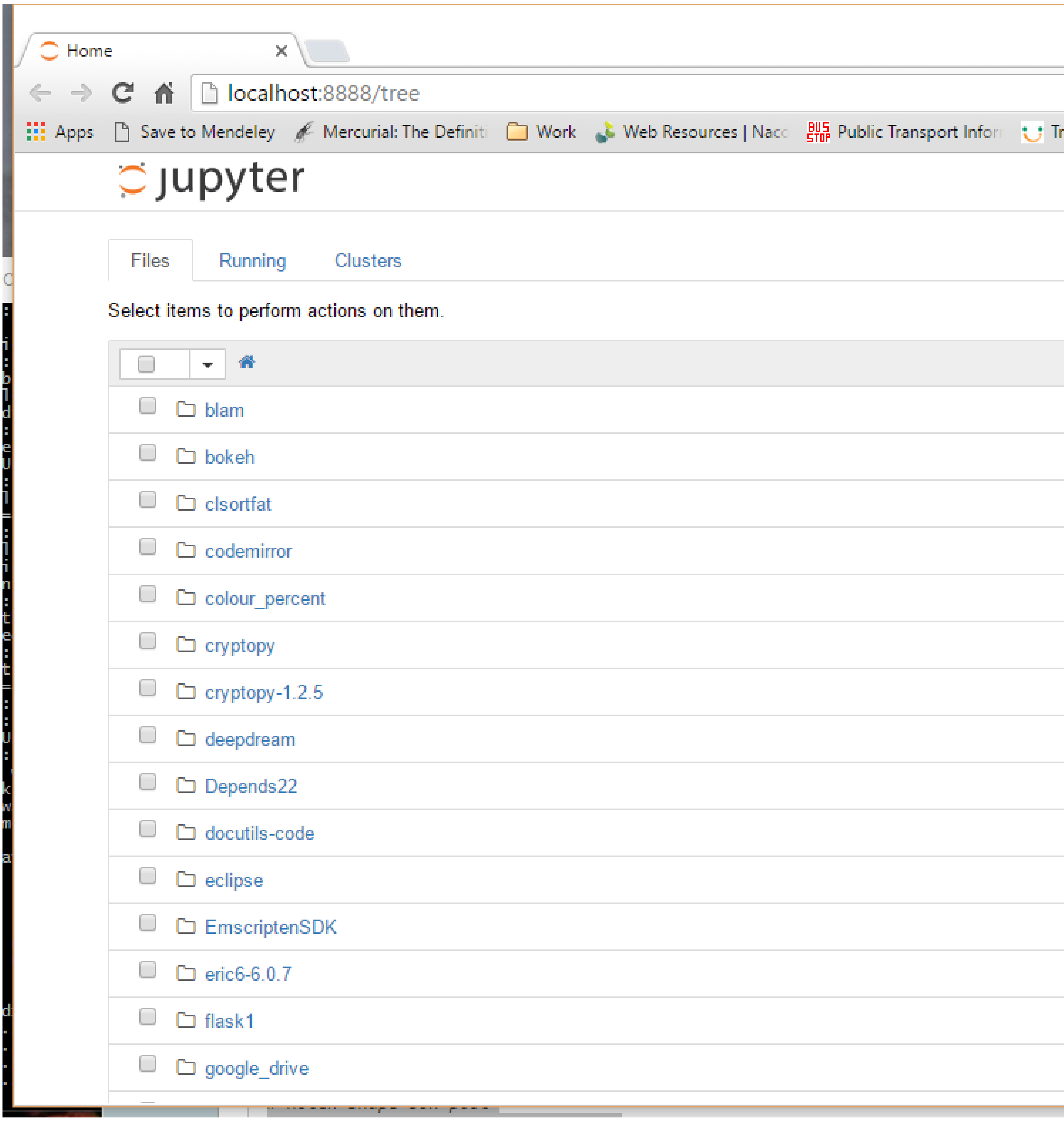
`jupyter qtconsole`, *or* `ipython qtconsole` *before version 5*, will start a multi-tabbed QT based console:



```
Jupyter QtConsole 4.2.1
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit
(Intel)]
Type "copyright", "credits" or "license" for more information.

IPython 5.0.0 -- 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]: # Box plots with custom fill colors
   ...:
   ...: import matplotlib.pyplot as plt
   ...: import numpy as np
   ...:
   ...: # Random test data
   ...: np.random.seed(123)
   ...: all_data = [np.random.normal(0, std, 100) for std in range(1, 4)]
   ...:
   ...: fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 5))
   ...:
   ...: # rectangular box plot
   ...: bplot1 = axes[0].boxplot(all_data,
   ...:                          vert=True,    # vertical box alignment
   ...:                          patch_artist=True)   # fill with color
   ...:
   ...: # notch shape box plot
```

`jupyter notebook`, *or* `ipython notebook` *before version 5*, will start a server and by default open up a web page, at `http://localhost:8888/tree`, with the "Home" view of the current directory. This allows you to open existing notebooks or new kernels *in several languages, depending on which you have installed*; each will be opened in a new browser tab.

Notebooks allow you to mix markdown, including MathJax, code from the kernel of your choice, plots and graphs, images and even videos.


Home

localhost:8888/tree

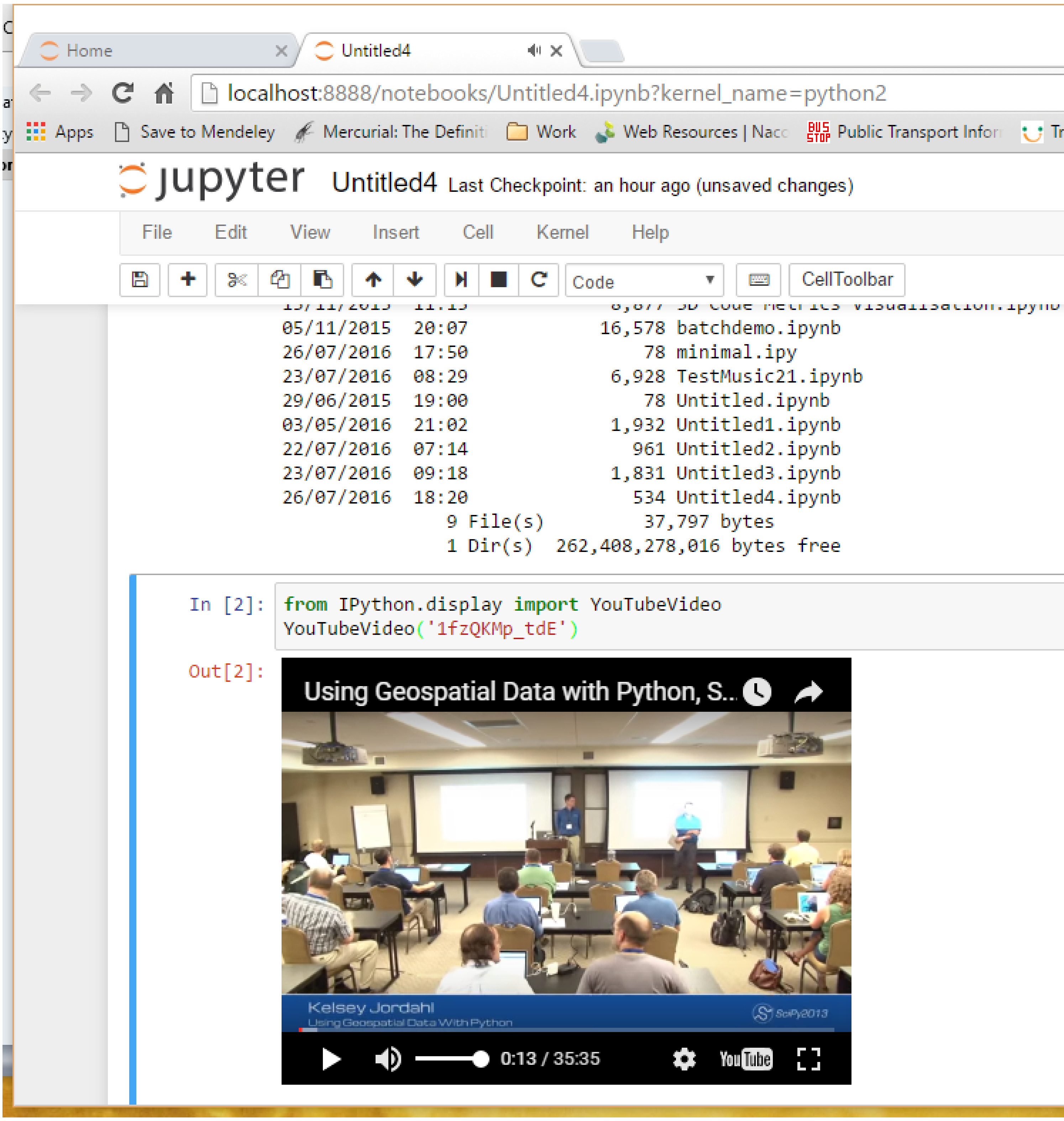
Apps | Save to Mendeley | Mercurial: The Definit | Work | Web Resources | Nacc | Public Transport Infor | Tr

jupyter

Files | Running | Clusters

Select items to perform actions on them.

- blam
- bokeh
- clsortfat
- codemirror
- colour_percent
- cryptopy
- cryptopy-1.2.5
- deepdream
- Depends22
- docutils-code
- eclipse
- EmscriptenSDK
- eric6-6.0.7
- flask1
- google_drive



```
05/11/2015  20:07            16,578 batchdemo.ipynb
26/07/2016  17:50                78 minimal.ipy
23/07/2016  08:29             6,928 TestMusic21.ipynb
29/06/2015  19:00                78 Untitled.ipynb
03/05/2016  21:02             1,932 Untitled1.ipynb
22/07/2016  07:14               961 Untitled2.ipynb
23/07/2016  09:18             1,831 Untitled3.ipynb
26/07/2016  18:20               534 Untitled4.ipynb
               9 File(s)         37,797 bytes
               1 Dir(s)  262,408,278,016 bytes free
```

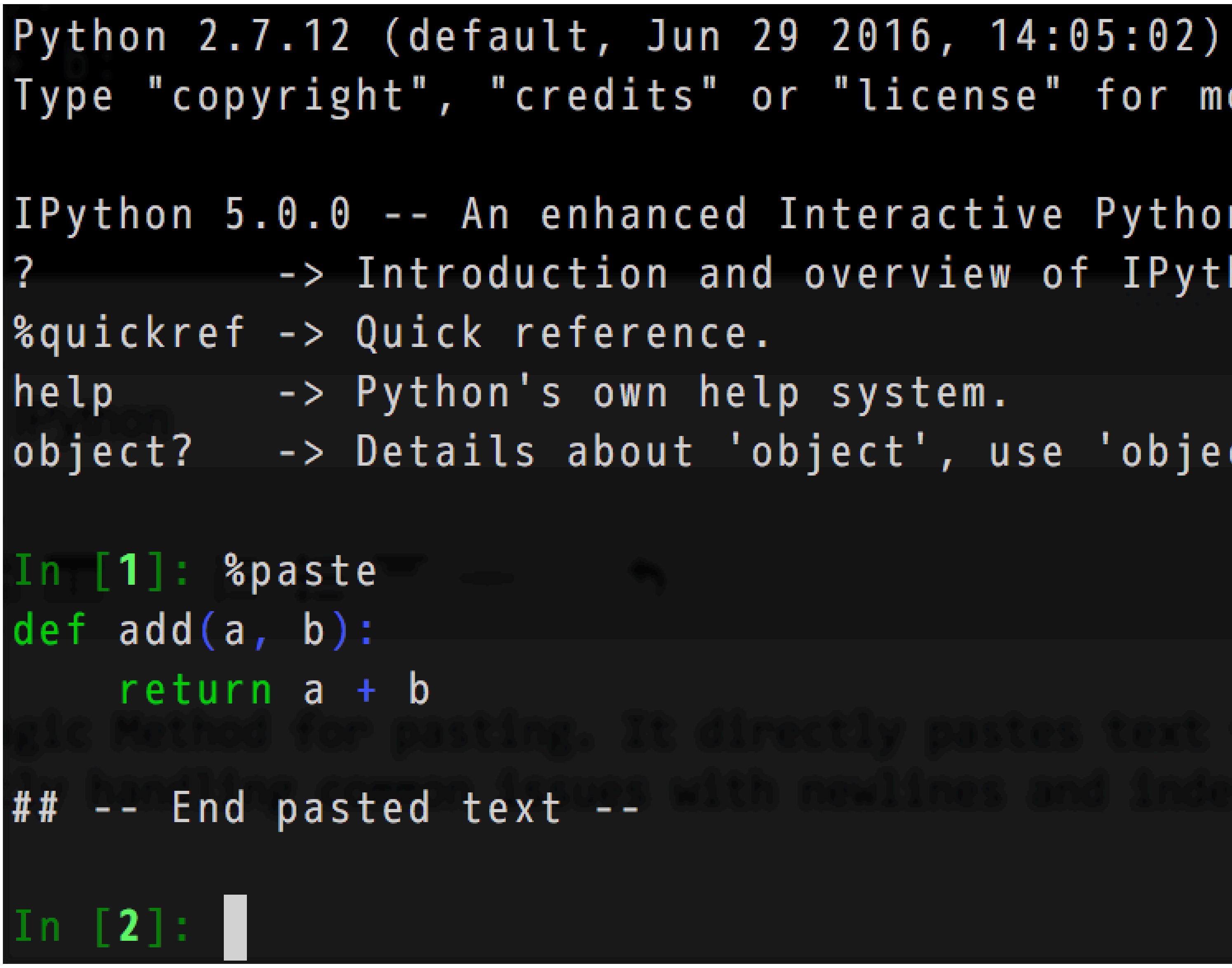
```
In [2]: from IPython.display import YouTubeVideo
        YouTubeVideo('1fzQKMp_tdE')
```

Out[2]:

Using Geospatial Data with Python, S...
Kelsey Jordahl
Using Geospatial Data With Python
SciPy2013
0:13 / 35:35

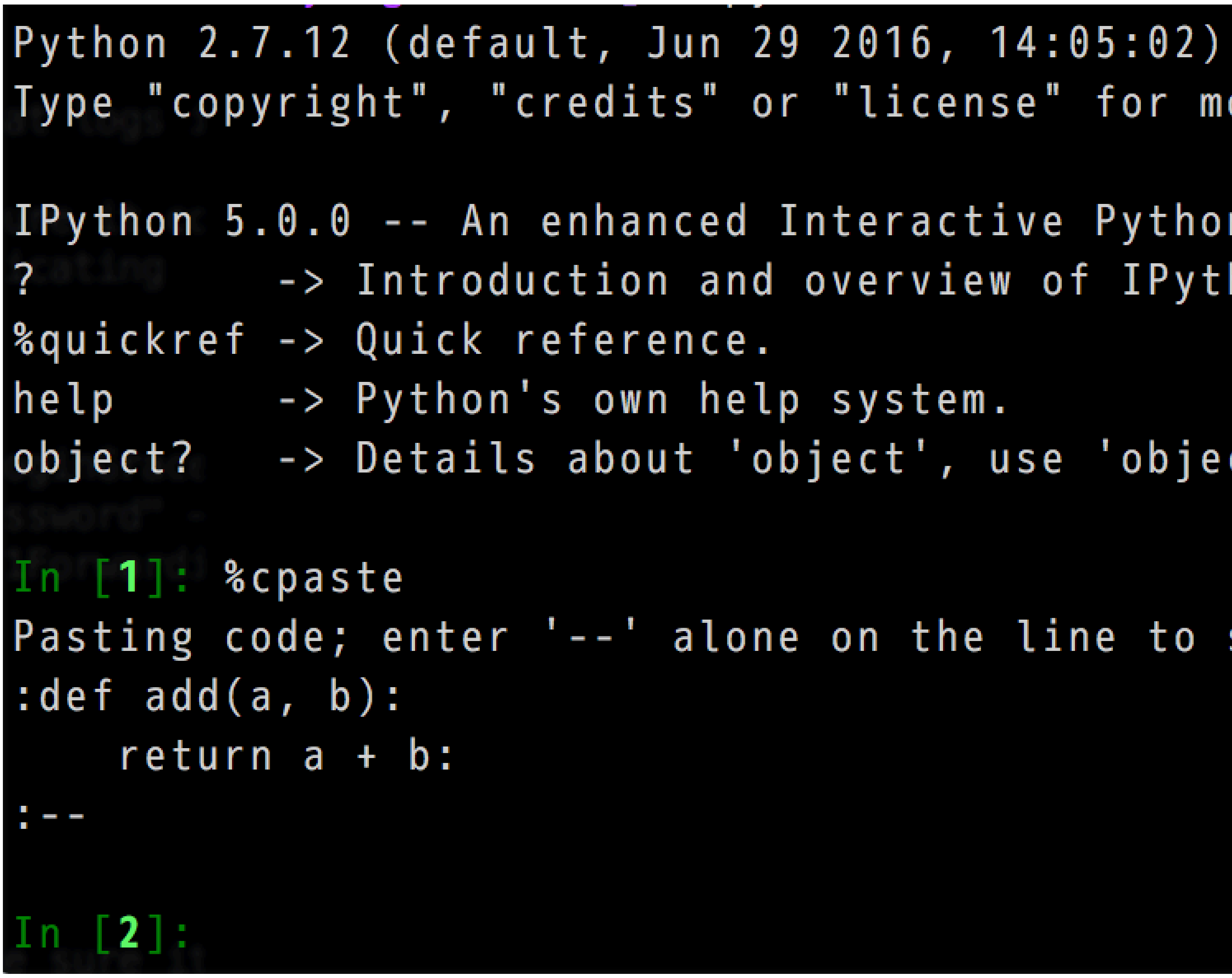
## Pasting into IPython

```
%paste
```


```
Python 2.7.12 (default, Jun 29 2016, 14:05:02)
Type "copyright", "credits" or "license" for m

IPython 5.0.0 -- An enhanced Interactive Pytho
?         -> Introduction and overview of IPyt
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'obje

In [1]: %paste
def add(a, b):
    return a + b

## -- End pasted text --

In [2]:
```

This is the primary Magic Method for pasting. It directly pastes text from the system clipboard, intelligently handling common issues with newlines and indentation.

```
%cpaste
```



```
Python 2.7.12 (default, Jun 29 2016, 14:05:02)
Type "copyright", "credits" or "license" for m

IPython 5.0.0 -- An enhanced Interactive Pytho
?         -> Introduction and overview of IPyt
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'obje

In [1]: %cpaste
Pasting code; enter '--' alone on the line to
:def add(a, b):
    return a + b:
:--

In [2]:
```

If you are using IPython via SSH, use `%cpaste` instead, as it does not need to access the remote system clipboard.

Since IPython 5.0.0, the improved prompt toolkit should directly handle pasting multi-line code without the need for `%paste` or `%cpaste`.

## Store variables on IPython

`%storemagic` stores variables and macros on IPython's database. To automatically restore stored variables at startup add this to `ipython_config.py`:

```
c.StoreMagic.autorestore = True
```

Example:

```
In [1]: l = ['hello',10,'world']
In [2]: %store l
In [3]: exit
```

```
(IPython session is closed and started again...)

ville@badger:~$ ipython
In [1]: l
Out[1]: ['hello', 10, 'world']
```

Note:

It should be noted that if you change the value of a variable, you need to %store it again if you want to persist the new value.

Note also that the variables will need to be pickleable; most basic python types can be safely %store'd.

Read Getting started with ipython online: https://riptutorial.com/ipython/topic/3400/getting-started-with-ipython

# Chapter 2: IPython shortcuts, tips and tricks

## Remarks

IPython (and the Jupyter Notebook) defines a special meaning for the underscore. It always contains the the most recent output. This comes in useful when processing data in multiple steps. This is demonstrated in the example above. To clean up the text, it is run through a few regular expressions and then normalized before being split. Outside of IPython, each result would have to be stored in a new variable or the steps nested. In IPython, often we are exploring and keeping track of variables or reproducing a long series of nested calls is tedious. So this is where the underscore appears.

There is one gotcha. If you assign a value to the underscore in the global scope, it causes unexpected behavior. For example:

```
address = ('http://example.com', 80)
(_, port) = address
```

Here I am only interested in the second element in the tuple, the port number. So I follow convention and assign the first element to the underscore to indicate it is a throwaway. However, now the value of the underscore is `http://example.com`. And if I were to run more code:

```
1+4
```

The expected value of the underscore would be 5. However, it is not. The value is still the domain from the tuple. When you assign to the underscore in the global scope, it not only clobbers the value, but it also stops storing the most recent output. This is not the case if you assign to the underscore inside a function or loop.

## Examples

### The special use of the underscore in IPython

```
from urllib.request import urlopen
from collections import Counter
import re

conn = urlopen('http://textfiles.com/100/dodontae.hum')
lines = conn.readlines()
conn.close()

# readlines() returns byte strings
data = ''.join([line.decode('utf-8') for line in lines])

# replace non-letters with a space
re.sub('[^A-Za-z]', ' ', data)

# condense successive whitespace into a single space
# the underscore retrieves the most recent output
re.sub('\s+', ' ', _)

# normalize the text by lowercasing and removing leading and trailing whitespace
_.lower().strip()

# split into words on space
words = _.split(' ')

from collections import Counter
word_count = Counter()

for word in words:
    word_count[word[0]] += 1

word_count.most_common()
```

Read IPython shortcuts, tips and tricks online: https://riptutorial.com/ipython/topic/10536/ipython-shortcuts--tips-and-tricks

# Chapter 3: Magics

## Introduction

IPython extends Python with a set of commands called *magics*. These are special functions whose names start with a `%` that are recognized by the IPython shell. Magics whose name begins with just one `%` take as argument the rest of the line and are called *line magics*. Magics that begin with a double percent sign `%%` take a multi-line argument and are called *cell magics*.

## Examples

### The %timeit and `%time` magics

The `%timeit` magic runs the given code many times, then returns the speed of the fastest result.

```
In [1]: %timeit sum(range(100000))
100 loops, best of 3: 2.91 ms per loop
```

The `%%timeit` cell magic can be used to time blocks of code.

```
In [2]: %%timeit
   ...: a = 0
   ...: for i in range(100000):
   ...:     a += i
   ...:
100 loops, best of 3: 9.67 ms per loop
```

The `%time` magic times a single run of a function, similar to the Unix `time` command. Unlike `%timeit`, `%time` also shows the result.

```
In [3]: %time sum(range(100000))
CPU times: user 2.68 ms, sys: 3 µs, total: 2.68 ms
Wall time: 2.69 ms
Out[3]: 4999950000
```

### Built-in line and cell magics

Magics whose name begins with just one `%` take as argument the rest of the line and are called *line magics*. Magics that begin with a double percent sign `%%` take a multi-line argument and are called *cell magics*.

A commonly used magic is `%timeit`, a wrapper around the Python's `timeit.timeit` function, for measuring the execution time of a piece of code.

```
In [35]: ra = [random.randint(0,1000) for r in range(1000)]
In [35]: %timeit sorted(ra)
1000 loops, best of 3: 507 µs per loop
```

An example of cell magic is `%%bash` (equivalent to `%%script bash`) for running the input as bash code

```
In [49]: %%bash
    ...: i=3
    ...: while [ $i -ge 0 ]
    ...: do
    ...: echo $i
    ...: i=$(($i-1))
    ...: done
    ...:
3
2
1
0
```

Note that the end of a cell is marked by an empty line.

To view all built-in magics use `%lsmagic`

```
In [51]: %lsmagic
Out[51]:
Available line magics:
%alias  %alias_magic  %autocall  %autoindent  %automagic  %bookmark  %cd  %cls
%colors  %config  %copy  %cpaste  %ddir  %debug  %dhist  %dirs  %doctest_mode
%echo  %ed  %edit  %env  %gui  %hist  %history  %killbgscripts  %ldir  %load
%load_ext  %loadpy  %logoff  %logon  %logstart  %logstate  %logstop  %ls  %lsmagic
%macro  %magic  %matplotlib  %mkdir  %notebook  %page  %paste  %pastebin  %pdb
%pdef  %pdoc  %pfile  %pinfo  %pinfo2  %popd  %pprint  %precision  %profile
%prun  %psearch  %psource  %pushd  %pwd  %pycat  %pylab  %quickref  %recall
%rehashx  %reload_ext  %ren  %rep  %rerun  %reset  %reset_selective  %rmdir  %run
%save  %sc  %set_env  %store  %sx  %system  %tb  %time  %timeit  %unalias
%unload_ext  %who  %who_ls  %whos  %xdel  %xmode

Available cell magics:
%%!  %%HTML  %%SVG  %%bash  %%capture  %%cmd  %%debug  %%file  %%html
%%javascript  %%js  %%latex  %%perl  %%prun  %%pypy  %%python  %%python2  %%python3
%%ruby  %%script  %%sh  %%svg  %%sx  %%system  %%time  %%timeit  %%writefile

Automagic is ON, % prefix IS NOT needed for line magics.
```

Note that by default 'automagic' is on, and therefore line magics can be called without `%` prefix (but cell functions still need a `%%` prefix).

Read Magics online: https://riptutorial.com/ipython/topic/5258/magics

# Credits

| S. No | Chapters | Contributors |
|---|---|---|
| 1 | Getting started with ipython | bastibe, Community, enrico.bacis, Fermi paradox, Jordan Georgiev, karel, k-nut, m00am, Matt, Steve Barnes, Thomas K, Tim Givois, Will |
| 2 | IPython shortcuts, tips and tricks | Douglas Starnes |
| 3 | Magics | asmeurer, user2314737 |