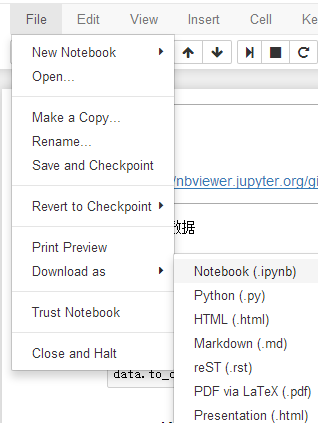
Jupyter Notebook简介



jupyter notebook转换成其他文件格式

1 打开anaconda的shell

2 cd 到 .ipynb 文件所在文件夹

3 最简单转换：默认为html格式

jupyter nbconvert py35\_rpy2.ipynb

4 可以转换成其他格式，如pdf（需安装pandoc、miktex，但仍有问题）

jupyter nbconvert py35\_rpy2.ipynb --to pdf

<http://hyry.dip.jp/tech/slice/slice.html/35>

IPython

输入reset 再y：可删除所有variable explorer中的变量

若输入reset out 再y：可删除所有之前输出结果的记录

若输入reset in 再y：可删除所有之前输入的记录

%matplotlib inline （先加这句，可在笔记中显示图像）

close\_px['AAPL'].plot()

|  |  |
| --- | --- |
| **魔术命令** | **说明** |
| %quickref | 显示IPython的快速参考 |
| %magic | 显示所有魔术命令的说明 |
| %debug | 从最新的异常跟踪底部进入交互式调试器 |
| %hist | 打印命令的输入历史 |
| %pdb | 在异常发生后自动进入调试器 |
| %paste | 执行剪贴板中的python代码 |
| %reset | 删除interactive命名空间中的全部变量及名称 |
| %run script.py | 在IPython中执行一个Python脚本文件 |
| %timeit stetement | 多次执行statement以计算系统平均执行时间 |
| %who %who\_ls %whos | 显示interactive命名空间中定义的变量、信息级别和冗余度 |
| %xdel variable | 删除variable，并尝试清除其在IPython中的对象上的一切引用 |

IPython Notebook使用浏览器作为界面，向后台的IPython服务器发送请求，并显示结果。在浏览器的界面中使用单元(Cell)保存各种信息。Cell有多种类型，经常使用的有表示格式化文本的[Markdown](http://daringfireball.net/projects/markdown/)单元，和表示代码的Code单元。

每个代码单元都有一个输出区域，在Code单元中输入代码，按 Shift-Enter 将运行此代码，代码中最后一个表达式的值将输出区域显示。如果希望屏蔽输出，可以在最后一条语句之后添加一个分号：”;”。此外，代码中还可以使用print语句在输出区域中显示信息。

在Markdown单元中还可以直接使用Html和Javascript。

**一、数学公式**

在Markdown单元中可以使用LaTeX表示数学公式，例如

0c27cf2af9a8.png

。数学公式的显示使用MathJax，缺省情况下，MathJax从网络上下载，如果希望离线使用它，需要在IPython Notebook中输入如下代码，把MathJax安装到本地磁盘中：

**from** **IPython.external.mathjax** **import** install\_mathjax
install\_mathjax()

Code单元的输出也可以显示为数学公式，例如在单元中输入如下代码，将显示为数学公式：

**from** **IPython.display** **import** Latex
Latex(r"$\sqrt{x^2+y^2}$")

cd97487620fb.png

SymPy的表达式也可以显示为LaTex，例如：

%load\_ext sympyprinting
**from** **sympy** **import** \*
x, y = symbols("x,y")
sqrt(x\*\*2+y\*\*2)

589f872b4927.png

以%开头的为IPython的命令(Magic Command)，这里通过%load\_ext命令载入sympyprinting扩展插件，载入此插件之后，所有的SymPy表达式都显示为数学公式。

**二、各种显示**

IPython.display模块中提供了许多显示Python返回值的类，例如下面的代码用Image类显示”python.png”图片，缺省路径为Notebook文件所在的目录：

**from** **IPython.display** **import** Image
Image(filename="python.png")

Image还可以用来显示表示图像的字符串。例如下面的代码通过cv2的imencode()将NumPy数组转换为一个表示PNG图像数据的数组，然后将此数组转换为字符串之后通过Image()将显示为图像：

**import** **cv2**
**import** **numpy** **as** **np**
**from** **IPython.display** **import** Image
img = np.random.randint(0,255,(250,250,3))
cv2.blur(img, (11,11), img)
r, dat = cv2.imencode(".png",img)
Image(dat.tostring())



此外，还可以通过HTML和Javascript将Python代码的输出显示为Html，或者作为Javascript运行。

**from** **IPython.display** **import** Javascript
Javascript("alert('ok')")

将在浏览器中运行Javascript代码。

**三、Magic命令**

**%pylab：**载入numpy和pylab，并且将这两个模块中的名字载入到全局名字空间中

**%load：**从文件或者网址载入代码到一个新的单元中

**%prun：**用于代码的执行性能分析，可以作为行命令和单元命令使用

**%load\_ext：**载入IPython的扩展模块，通过它可以载入更多的Magic命令

IPython中Magic命令有两种执行方式，以%开始的命令被称为行命令，它只对单行有效，以%%开头的为单元命令，它放在单元的第一行，对整个单元有效。

例如**timeit**命令可以快速测试代码的执行效率，它可以作为行命令或者单元命令。

%timeit 1 + 1
%timeit 1.0 + 1.0
%timeit "1" + "1"

10000000 loops, best of 3: 52 ns per loop
10000000 loops, best of 3: 53.4 ns per loop
10000000 loops, best of 3: 50.9 ns per loop

%%timeit
s = 0
**for** i **in** xrange(100):
s += i

100000 loops, best of 3: 11 us per loop

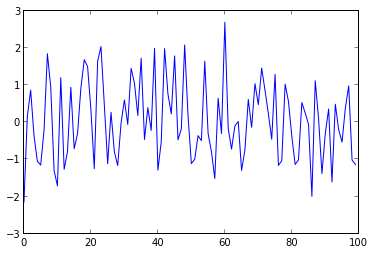
每个Magic命令都可以指定参数，可以输入timeit?查看其帮助文档。下面让我们看看一些常用的Magic命令。

**%pylab**命令将载入numpy和pylab，并且将这两个模块中的名字载入到全局名字空间中。缺省参数时，它使用matplotlib的缺省界面库显示图表，如果带**inline**参数则将图表作为图像插入到Notebook中。使用界面库显示图像时可以使用交互工具，而将图表直接插入到Notebook中则有利于编写文档。

下面的例子，plot和random是从pylab和numpy中载入的。

%pylab inline
plot(random.randn(100));

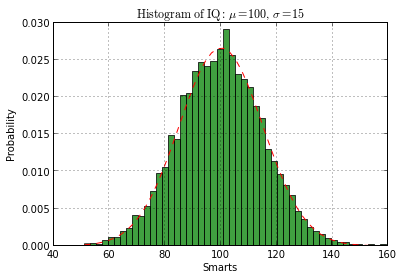
Welcome to pylab, a matplotlib-based Python environment [backend: module://IPython.zmq.pylab.backend\_inline].
For more information, type 'help(pylab)'.



**%load**可以从文件或者网址载入代码到一个新的单元中，例如下面载入某个matplotlib的示例程序，并执行：

%load http://matplotlib.org/mpl\_examples/pylab\_examples/histogram\_demo.py

#!/usr/bin/env python
**import** **numpy** **as** **np**
**import** **matplotlib.mlab** **as** **mlab**
**import** **matplotlib.pyplot** **as** **plt**
mu, sigma = 100, 15
x = mu + sigma\*np.random.randn(10000)
# the histogram of the data
n, bins, patches = plt.hist(x, 50, normed=1, facecolor='green', alpha=0.75)
# add a 'best fit' line
y = mlab.normpdf( bins, mu, sigma)
l = plt.plot(bins, y, 'r--', linewidth=1)
plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title(r'$\mathrm{Histogram\ of\ IQ:}\ \mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()



**%prun**用于代码的执行性能分析，可以作为行命令和单元命令使用。下面的程序分析numpy.linalg.det()的性能：

%%prun
**for** i **in** xrange(100):
linalg.det(random.rand(10,10))

其输出如下：

3402 function calls in 0.096 seconds
Ordered by: internal time
ncalls tottime percall cumtime percall filename:lineno(function)
100 0.032 0.000 0.091 0.001 linalg.py:1560(slogdet)
300 0.022 0.000 0.022 0.000 {method 'reduce' of 'numpy.ufunc' objects}
200 0.011 0.000 0.012 0.000 numeric.py:167(asarray)
100 0.006 0.000 0.006 0.000 linalg.py:84(\_realType)
100 0.005 0.000 0.005 0.000 linalg.py:151(\_assertRank2)
...

**%load\_ext**载入IPython的扩展模块，通过它可以载入更多的Magic命令。下面我们载入**cythonmagic**模块，并使用**%%cython**命令编译一个高效的频率统计函数count()。

%load\_ext cythonmagic

%pylab inline：导入pylab(mathplotlib)库，而且显示图片时可以嵌在网页中的.

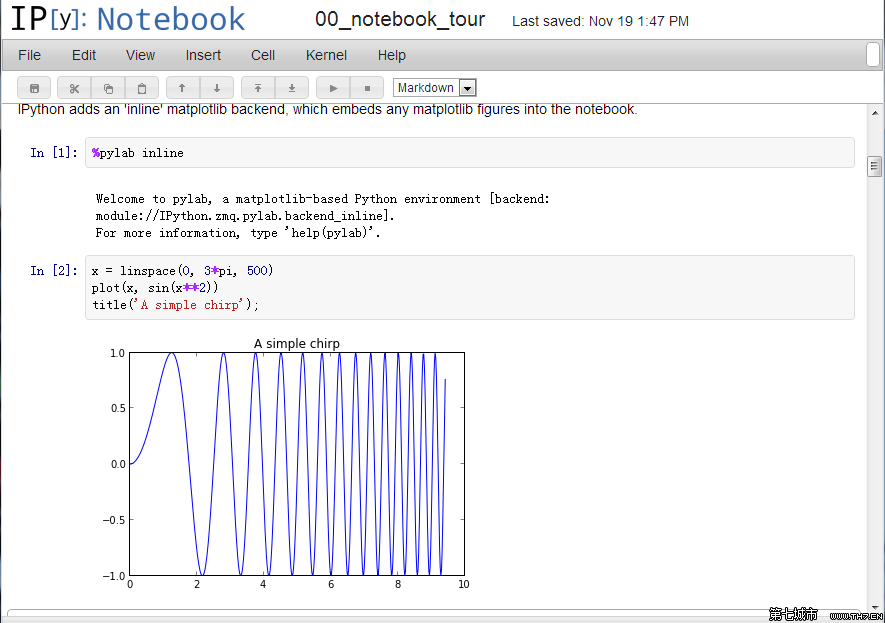
<http://www.360doc.com/content/14/0426/19/9482_372445625.shtml>

浏览器.ipython使用了websocket,它是不支持IE的.一般Chrome就好了

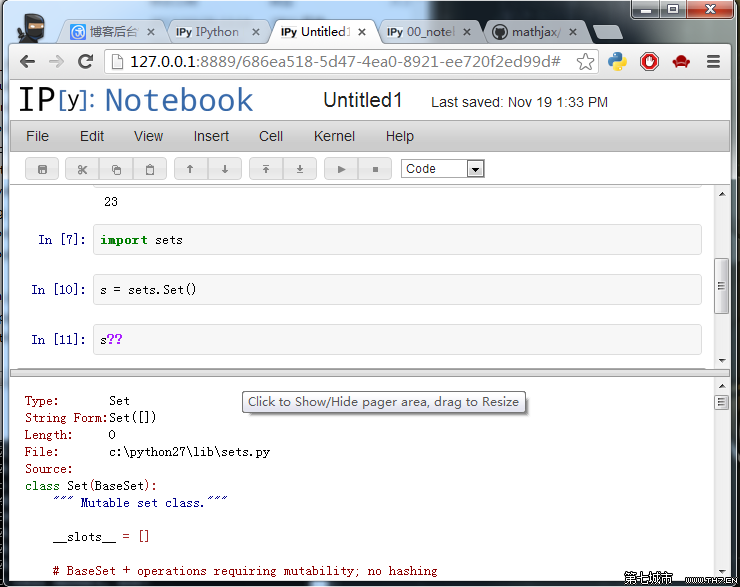
在markdown中是支持代码着色的

 ipython notebook --pylab inline ： 导入pylab(mathplotlib)库，而且显示图片时可以嵌在网页中的.

%pylab inline ： 当然也可以在网页中输入 %pylab inline启动这个功能的.



如果在函数,模块,类后面输入 '?'按住Ctrl-Entry的话就回跳出帮助文档.如果是两个??的话连,源代码都会给出来的



**快捷键：**

Esc + F 在代码中查找、替换，忽略输出。

Esc + O 在cell和输出结果间切换。

选择多个cell:

Shift + J 或 Shift + Down 选择下一个cell。

Shift + K 或 Shift + Up 选择上一个cell。

一旦选定cell，可以批量删除/拷贝/剪切/粘贴/运行。当你需要移动notebook的一部分时这个很有用。

Shift + M 合并cell.

[**命令模式**](https://www.baidu.com/s?wd=%E5%91%BD%E4%BB%A4%E6%A8%A1%E5%BC%8F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Yzm1b3Pjf4uHfYPjDznhDk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6snjm4PHT4rjbLPWfsnHn3Ps)

**(按键 Esc 开启)**

Enter : 转入编辑模式

Shift-Enter : 运行本单元，选中下个单元

Ctrl-Enter : 运行本单元

Alt-Enter : 运行本单元，在其下插入新单元

Y : 单元转入代码状态

M :单元转入markdown状态

R : 单元转入raw状态

1 : 设定 1 级标题

2 : 设定 2 级标题

3 : 设定 3 级标题

4 : 设定 4 级标题

5 : 设定 5 级标题

6 : 设定 6 级标题

Up : 选中上方单元

K : 选中上方单元

Down : 选中下方单元

J : 选中下方单元

Shift-K : 扩大选中上方单元

Shift-J : 扩大选中下方单元

A : 在上方插入新单元

B : 在下方插入新单元

X : 剪切选中的单元

C : 复制选中的单元

Shift-V : 粘贴到上方单元

V : 粘贴到下方单元

Z : 恢复删除的最后一个单元

D,D : 删除选中的单元

Shift-M : 合并选中的单元

Ctrl-S : 文件存盘

S : 文件存盘

L : 转换行号

O : 转换输出

Shift-O : 转换输出滚动

Esc : 关闭页面

Q : 关闭页面

H : 显示快捷键帮助

I,I : 中断Notebook内核

0,0 : 重启Notebook内核

Shift : 忽略

Shift-Space : 向上滚动

Space : 向下滚动

**编辑模式 ( Enter 键启动)**

Tab : 代码补全或缩进

Shift-Tab : 提示

Ctrl-] : 缩进

Ctrl-[ : 解除缩进

Ctrl-A : 全选

Ctrl-Z : 复原

Ctrl-Shift-Z : 再做

Ctrl-Y : 再做

Ctrl-Home : 跳到单元开头

Ctrl-Up : 跳到单元开头

Ctrl-End : 跳到单元末尾

Ctrl-Down : 跳到单元末尾

Ctrl-Left : 跳到左边一个字首

Ctrl-Right : 跳到右边一个字首

Ctrl-Backspace : 删除前面一个字

Ctrl-Delete : 删除后面一个字

Esc : 进入[命令模式](https://www.baidu.com/s?wd=%E5%91%BD%E4%BB%A4%E6%A8%A1%E5%BC%8F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Yzm1b3Pjf4uHfYPjDznhDk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6snjm4PHT4rjbLPWfsnHn3Ps)

Ctrl-M : 进入[命令模式](https://www.baidu.com/s?wd=%E5%91%BD%E4%BB%A4%E6%A8%A1%E5%BC%8F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Yzm1b3Pjf4uHfYPjDznhDk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6snjm4PHT4rjbLPWfsnHn3Ps)

Shift-Enter : 运行本单元，选中下一单元

Ctrl-Enter : 运行本单元

Alt-Enter : 运行本单元，在下面插入一单元

Ctrl-Shift-- : 分割单元

Ctrl-Shift-Subtract : 分割单元

Ctrl-S : 文件存盘

Shift : 忽略

Up : 光标上移或转入上一单元

Down :光标下移或转入下一单元

Jupyter Notebook的27个窍门，技巧和快捷键

Jupyter延伸性很好，支持多种编程语言，可以很轻松地安装在个人电脑或者任何服务器上——只要有ssh或者http接入就可以啦。最棒的一点是，它完全免费哦。

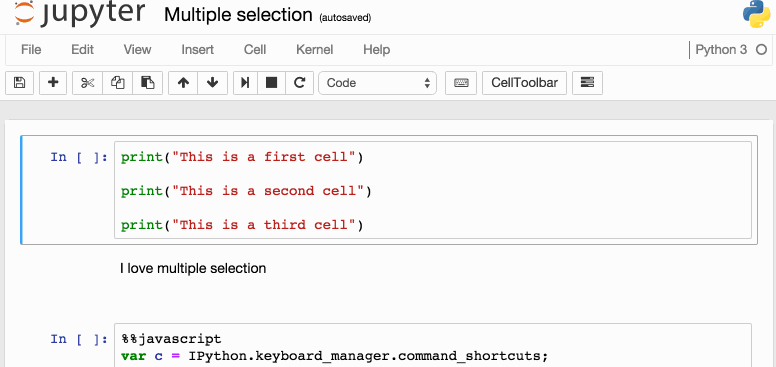
Jupyter 界面

默认情况下，Jupyter Notebook 使用Python内核，这就是为什么它原名 IPython Notebook。Jupyter notebook是Jupyter项目的产物——Jupyter这个名字是它要服务的三种语言的缩写：Julia，PYThon和R，这个名字与“木星（jupiter）”谐音。本文将介绍27个轻松使用Jupyter的小窍门和技巧。

**◆ ◆ ◆**

**1.快捷键**

高手们都知道，快捷键可以节省很多时间。Jupyter在顶部菜单提供了一个快捷键列表：Help > Keyboard Shortcuts 。每次更新Jupyter的时候，一定要看看这个列表，因为不断地有新的快捷键加进来。另外一个方法是使用Cmd + Shift + P (  Linux 和 Windows下 Ctrl + Shift + P亦可)调出命令面板。这个对话框可以让你通过名称来运行任何命令——当你不知道某个操作的快捷键，或者那个操作没有快捷键的时候尤其有用。这个功能与苹果电脑上的Spotlight搜索很像，一旦开始使用，你会欲罢不能。



几个我的最爱：

Esc + F 在代码中查找、替换，忽略输出。

Esc + O 在cell和输出结果间切换。

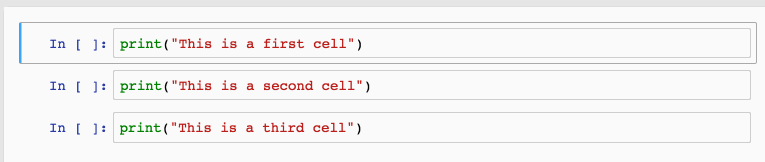
选择多个cell:

Shift + J 或 Shift + Down 选择下一个cell。

Shift + K 或 Shift + Up 选择上一个cell。

一旦选定cell，可以批量删除/拷贝/剪切/粘贴/运行。当你需要移动notebook的一部分时这个很有用。

Shift + M 合并cell.



**◆ ◆ ◆**

**2.变量的完美显示**

有一点已经众所周知。把变量名称或没有定义输出结果的语句放在cell的最后一行，无需print语句，Jupyter也会显示变量值。当使用Pandas DataFrames时这一点尤其有用，因为输出结果为整齐的表格。

鲜为人知的是，你可以通过修改内核选项ast\_note\_interactivity，使得Jupyter对独占一行的所有变量或者语句都自动显示，这样你就可以马上看到多个语句的运行结果了。

In [1]: from IPython.core.interactiveshell import InteractiveShell

       InteractiveShell.ast\_node\_interactivity = "all"

In [2]: from pydataset import data

       quakes = data('quakes')

       quakes.head()

       quakes.tail()

Out[2]:

       lat long    depth   mag stations

       1   -20.42  181.62  562 4.8 41

       2   -20.62  181.03  650 4.2 15

       3   -26.00  184.10  42  5.4 43

       4   -17.97  181.66  626 4.1 19

       5   -20.42  181.96  649 4.0 11

Out[2]:

       lat long    depth   mag stations

       996 -25.93  179.54  470 4.4 22

       997 -12.28  167.06  248 4.7 35

       998 -20.13  184.20  244 4.5 34

       999 -17.40  187.80  40  4.5 14

       1000    -21.59  170.56  165 6.0 119

如果你想在各种情形下（Notebook和Console）Jupyter都同样处理，用下面的几行简单的命令创建文件~/.ipython/profile\_default/ipython\_config.py即可实现：

c = get\_config()

# Run all nodes interactively

c.InteractiveShell.ast\_node\_interactivity = "all"

**◆ ◆ ◆**

**3.轻松链接到文档**

在Help 菜单下，你可以找到常见库的在线文档链接，包括Numpy，Pandas，Scipy和Matplotlib等。

另外，在库、方法或变量的前面打上?，即可打开相关语法的帮助文档。

In [3]: ?str.replace()

       Docstring:
       S.replace(old, new[, count]) -> str
       Return a copy of S with all occurrences of substring
       old replaced by new.  If the optional argument count is
       given, only the first count occurrences are replaced.
       Type:      method\_descriptor

**◆ ◆ ◆**

**4.在notebok里作图**

**在notebook里作图，有多个选择：**

- matplotlib （事实标准）（http://matplotlib.org/），可通过%matplotlib inline 激活，（https://www.dataquest.io/blog/matplotlib-tutorial/）

- %matplotlib notebook 提供交互性操作，但可能会有点慢，因为响应是在服务器端完成的。

- mpld3（https://github.com/mpld3/mpld3） 提供matplotlib代码的替代性呈现（通过d3），虽然不完整，但很好。

- bokeh（http://bokeh.pydata.org/en/latest/） 生成可交互图像的更好选择。

- plot.ly（https://plot.ly/） 可以生成非常好的图，可惜是付费服务。

**◆ ◆ ◆**

**5.Jupyter Magic命令**

上文提到的%matplotlib inline 是Jupyter Magic命令之一。

推荐阅读Jupyter magic命令的相关文档

（http://ipython.readthedocs.io/en/stable/interactive/magics.html），它一定会对你很有帮助。下面是我最爱的几个：

**◆ ◆ ◆**

**6.Jupyter Magic-%env:设置环境变量**

不必重启jupyter服务器进程，也可以管理notebook的环境变量。有的库（比如theano）使用环境变量来控制其行为，%env是最方便的途径。

In [55]:    # Running %env without any arguments
           # lists all environment variables
           # The line below sets the environment
           # variable OMP\_NUM\_THREADS
           %env OMP\_NUM\_THREADS=4

           env: OMP\_NUM\_THREADS=4

**◆ ◆ ◆**

**7.Jupyter Magic-%run:运行python代码**

%run 可以运行.py格式的python代码——这是众所周知的。不那么为人知晓的事实是它也可以运行其它的jupyter notebook文件，这一点很有用。

注意：使用%run 与导入一个python模块是不同的。

In [56]:    # this will execute and show the output from
           # all code cells of the specified notebook
           %run ./two-histograms.ipynb

**◆ ◆ ◆**

**8.Jupyter Magic-%load：从外部脚本中插入代码**

该操作用外部脚本替换当前cell。可以使用你的电脑中的一个文件作为来源，也可以使用URL。

In [ ]:     # Before Running
           %load ./hello\_world.py
In [61]:    # After Running
           # %load ./hello\_world.py
           if \_\_name\_\_ == "\_\_main\_\_":
               print("Hello World!")

           Hello World!

**◆ ◆ ◆**

**9.Jupyter Magic-%store：在notebook文件之间传递变量**

%store 命令可以在两个notebook文件之间传递变量。

In [62]:    data = 'this is the string I want to pass to different notebook'
           %store data
           del data # This has deleted the variable

           Stored 'data' (str)

现在，在一个新的notebook文档里……

In [1]: %store -r data
       print(data)

       this is the string I want to pass to different notebook

**◆ ◆ ◆**

**10.Jupyter Magic-%who：列出所有的全局变量**

不加任何参数， %who 命令可以列出所有的全局变量。加上参数 str 将只列出字符串型的全局变量。

In [1]: one = "for the money"
       two = "for the show"
       three = "to get ready now go cat go"
       %who str

       one  three   two

**◆ ◆ ◆**

**11.Jupyter Magic-计时**

有两种用于计时的jupyter magic命令： %%time 和 %timeit.当你有一些很耗时的代码，想要查清楚问题出在哪时，这两个命令非常给力。

仔细体会下我的描述哦。

%%time 会告诉你cell内代码的单次运行时间信息。

In [4]: %%time
       import time
       for \_ in range(1000):
           time.sleep(0.01)# sleep for 0.01 seconds

       CPU times: user 21.5 ms, sys: 14.8 ms, total: 36.3 ms
       Wall time: 11.6 s

%%timeit 使用了Python的 timeit 模块，该模块运行某语句100，000次（默认值），然后提供最快的3次的平均值作为结果。

In [3]: import numpy
       %timeit numpy.random.normal(size=100)

       The slowest run took 7.29 times longer than the fastest. This could mean that an intermediate result is being cached.
       100000 loops, best of 3: 5.5 µs per loop

**◆ ◆ ◆**

**12.Jupyter Magic-writefile and %pycat:导出cell内容/显示外部脚本的内容**

使用%%writefile magic可以保存cell的内容到外部文件。 而%pycat功能相反，把外部文件语法高亮显示（以弹出窗方式）。

In [7]: %%writefile pythoncode.py
       import numpy
       def append\_if\_not\_exists(arr, x):
           if x not in arr:
               arr.append(x)
       def some\_useless\_slow\_function():
           arr = list()
           for i in range(10000):
               x = numpy.random.randint(0, 10000)
               append\_if\_not\_exists(arr, x)

       Writing pythoncode.py

In [8]: %pycat pythoncode.py

       import numpy
       def append\_if\_not\_exists(arr, x):
           if x not in arr:
               arr.append(x)
       def some\_useless\_slow\_function():
           arr = list()
           for i in range(10000):
               x = numpy.random.randint(0, 10000)
               append\_if\_not\_exists(arr, x)

**◆ ◆ ◆**

**13.Jupyter Magic-%prun：告诉你程序中每个函数消耗的时间**

使用%prun+函数声明会给你一个按顺序排列的表格，显示每个内部函数的耗时情况，每次调用函数的耗时情况，以及累计耗时。

In [47]:    %prun some\_useless\_slow\_function()

        26324 function calls in 0.556 seconds
  Ordered by: internal time
  ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   10000    0.527    0.000    0.528    0.000 <ipython-input-46-b52343f1a2d5>:2(append\_if\_not\_exists)
   10000    0.022    0.000    0.022    0.000 {method 'randint' of 'mtrand.RandomState' objects}
       1    0.006    0.006    0.556    0.556 <ipython-input-46-b52343f1a2d5>:6(some\_useless\_slow\_function)
    6320    0.001    0.000    0.001    0.000 {method 'append' of 'list' objects}
       1    0.000    0.000    0.556    0.556 <string>:1(<module>)
       1    0.000    0.000    0.556    0.556 {built-in method exec}
       1    0.000    0.000    0.000    0.000 {method 'disable' of '\_lsprof.Profiler' objects}

**◆ ◆ ◆**

**14.Jupyter Magic-用%pdb调试程序**

Jupyter 有自己的调试界面The Python Debugger (pdb)（https://docs.python.org/3.5/library/pdb.html），使得进入函数内部检查错误成为可能。

Pdb中可使用的命令见链接（https://docs.python.org/3.5/library/pdb.html#debugger-commands）

In [ ]: %pdb
       def pick\_and\_take():
           picked = numpy.random.randint(0, 1000)
           raise NotImplementedError()
       pick\_and\_take()
       Automatic pdb calling has been turned ON
       ---------------------------------------------------------------------------
       NotImplementedError                       Traceback (most recent call last)
       <ipython-input-24-0f6b26649b2e> in <module>()
             5     raise NotImplementedError()
             6
       ----> 7 pick\_and\_take()
       <ipython-input-24-0f6b26649b2e> in pick\_and\_take()
             3 def pick\_and\_take():
             4     picked = numpy.random.randint(0, 1000)
       ----> 5     raise NotImplementedError()
             6
             7 pick\_and\_take()
       NotImplementedError:
       > <ipython-input-24-0f6b26649b2e>(5)pick\_and\_take()
             3 def pick\_and\_take():
             4     picked = numpy.random.randint(0, 1000)
       ----> 5     raise NotImplementedError()
             6
             7 pick\_and\_take()
       ipdb>

**◆ ◆ ◆**

**15.末句函数不输出**

有时候不让末句的函数输出结果比较方便，比如在作图的时候，此时，只需在该函数末尾加上一个分号即可。

In [4]: %matplotlib inline
       from matplotlib import pyplot as plt
       import numpy
       x = numpy.linspace(0, 1, 1000)\*\*1.5
In [5]: # Here you get the output of the function
       plt.hist(x)
Out[5]:
       (array([ 216.,  126.,  106.,   95.,   87.,   81.,   77.,   73.,   71.,   68.]),
        array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ]),
        <a list of 10 Patch objects>)

In [6]: # By adding a semicolon at the end, the output is suppressed.
       plt.hist(x);

**◆ ◆ ◆**

**16.运行Shell命令**

在notebook内部运行shell命令很简单，这样你就可以看到你的工作文件夹里有哪些数据集。

In [7]: !ls \*.csv

nba\_2016.csv             titanic.csv
pixar\_movies.csv         whitehouse\_employees.csv

**◆ ◆ ◆**

**17.用LaTex写公式**

当你在一个Markdown单元格里写LaTex时，它将用MathJax呈现公式：如

**$$ P(A \mid B) = \frac{P(B \mid A) , P(A)}{P(B)} $$**

会变成

**◆ ◆ ◆**

**18.在notebook内用不同的内核运行代码**

如果你想要，其实可以把不同内核的代码结合到一个notebook里运行。

只需在每个单元格的起始，用Jupyter magics调用kernal的名称：

%%bash

%%HTML

%%python2

%%python3

%%ruby

%%perl

In [6]: %%bash
       for i in {1..5}
       do
          echo "i is $i"
       done

           i is 1
           i is 2
           i is 3
           i is 4
           i is 5

**◆ ◆ ◆**

**19.给Jupyter安装其他的内核**

Jupyter的优良性能之一是可以运行不同语言的内核。下面以运行R内核为例说明：

简单的方法：通过Anaconda安装R内核

conda install -c r r-essentials

稍微麻烦的方法：手动安装R内核

如果你不是用Anaconda，过程会有点复杂，首先，你需要从CRAN安装R。

之后，启动R控制台，运行下面的语句：

install.packages(c('repr', 'IRdisplay', 'crayon', 'pbdZMQ', 'devtools'))
devtools::install\_github('IRkernel/IRkernel')
IRkernel::installspec()  # to register the kernel in the current R installation

**◆ ◆ ◆**

**20.在同一个notebook里运行R和Python**

要这么做，最好的方法事安装rpy2（需要一个可以工作的R），用pip操作很简单：

**pip install rpy2**

然后，就可以同时使用两种语言了，甚至变量也可以在二者之间公用：

In [1]: %load\_ext rpy2.ipython
In [2]: %R require(ggplot2)
Out[2]: array([1], dtype=int32)
In [3]: import pandas as pd
       df = pd.DataFrame({
               'Letter': ['a', 'a', 'a', 'b', 'b', 'b', 'c', 'c', 'c'],
               'X': [4, 3, 5, 2, 1, 7, 7, 5, 9],
               'Y': [0, 4, 3, 6, 7, 10, 11, 9, 13],
               'Z': [1, 2, 3, 1, 2, 3, 1, 2, 3]
           })
In [4]: %%R -i df
       ggplot(data = df) + geom\_point(aes(x = X, y= Y, color = Letter, size = Z))

**◆ ◆ ◆**

**21.用其他语言写函数**

有时候numpy的速度有点慢，我想写一些更快的代码。

原则上，你可以在动态库里编译函数，用python来封装…

但是如果这个无聊的过程不用自己干，岂不更好？

你可以在cython或fortran里写函数，然后在python代码里直接调用。

首先，你要先安装：

!pip install cython fortran-magic
In [ ]: %load\_ext Cython
In [ ]: %%cython
       def myltiply\_by\_2(float x):
           return 2.0 \* x
In [ ]: myltiply\_by\_2(23.)

我个人比较喜欢用Fortran，它在写数值计算函数时十分方便。更多的细节在（http://arogozhnikov.github.io/2015/09/08/SpeedBenchmarks.html）。

In [ ]: %load\_ext fortranmagic
In [ ]: %%fortran
       subroutine compute\_fortran(x, y, z)
           real, intent(in) :: x(:), y(:)
           real, intent(out) :: z(size(x, 1))
           z = sin(x + y)
       end subroutine compute\_fortran
In [ ]: compute\_fortran([1, 2, 3], [4, 5, 6])

还有一些别的跳转系统可以加速python 代码。更多的例子见（http://arogozhnikov.github.io/2015/09/08/SpeedBenchmarks.html）

你可以在cython或fortran里写函数，然后在python代

**◆ ◆ ◆**

**22.支持多指针**

Jupyter支持多个指针同步编辑，类似Sublime Text编辑器。按下Alt键并拖拽鼠标即可实现。

**◆ ◆ ◆**

**23.Jupyter外界拓展**

Jupyter-contrib extensions（https://github.com/ipython-contrib/jupyter\_contrib\_nbextensions）是一些给予Jupyter更多更能的延伸程序，包括jupyter spell-checker和code-formatter之类.

下面的命令安装这些延伸程序，同时也安装一个菜单形式的配置器，可以从Jupyter的主屏幕浏览和激活延伸程序。

!pip install https://github.com/ipython-contrib/jupyter\_contrib\_nbextensions/tarball/master
!pip install jupyter\_nbextensions\_configurator
!jupyter contrib nbextension install --user
!jupyter nbextensions\_configurator enable --user

**◆ ◆ ◆**

**24.从Jupyter notebook创建演示稿**

Damian Avila的RISE（https://github.com/damianavila/RISE）允许你从已有的notebook创建一个powerpoint形式的演示稿。

你可以用conda来安装RISE：

conda install -c damianavila82 rise

或者用pip安装：

pip install RISE

然后运行下面的代码来安装和激活延伸程序：

jupyter-nbextension install rise --py --sys-prefix
jupyter-nbextension enable rise --py --sys-prefix

**◆ ◆ ◆**

**25.Jupyter输出系统**

Notebook本身以HTML的形式显示，单元格输出也可以是HTML形式的，所以你可以输出任何东西：视频/音频/图像。

这个例子是浏览我所有的图片，并显示前五张图的缩略图。

In [12]:    import os
           from IPython.display import display, Image
           names = [f for f in os.listdir('../images/ml\_demonstrations/') if f.endswith('.png')]
           for name in names[:5]:
               display(Image('../images/ml\_demonstrations/' + name, width=100))

我们也可以用bash命令创建一个相同的列表，因为magics和bash运行函数后返回的是python 变量：

In [10]:    names = !ls ../images/ml\_demonstrations/\*.png
           names[:5]
Out[10]:    ['../images/ml\_demonstrations/colah\_embeddings.png',
            '../images/ml\_demonstrations/convnetjs.png',
            '../images/ml\_demonstrations/decision\_tree.png',
            '../images/ml\_demonstrations/decision\_tree\_in\_course.png',
            '../images/ml\_demonstrations/dream\_mnist.png']

**◆ ◆ ◆**

**26.大数据分析**

很多方案可以解决查询/处理大数据的问题：

ipyparallel（https://github.com/ipython/ipyparallel）（之前叫 ipython cluster） 是一个在python中进行简单的map-reduce运算的良好选择。我们在rep中使用它来并行训练很多机器学习模型。

pyspark（http://www.cloudera.com/documentation/enterprise/5-5-x/topics/spark\_ipython.html）

spark-sql magic %%sql（https://github.com/jupyter-incubator/sparkmagic）

**◆ ◆ ◆**

**27.分享notebook**

分享notebook最方便的方法是使用notebook文件（.ipynb），但是对那些不使用notebook的人，你还有这些选择：

通过File > Download as > HTML  菜单转换到html文件。

用gists（https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/gist.github.com）或者github分享你的notebook文件。这两个都可以呈现notebook，示例见链接（https://github.com/dataquestio/solutions/blob/master/Mission202Solution.ipynb）

如果你把自己的notebook文件上传到github的仓库，可以使用很便利的Mybinder（http://mybinder.org/）服务，允许另一个人进行半个小时的Jupyter交互连接到你的仓库。

用jupyterhub（https://github.com/jupyterhub/jupyterhub）建立你自己的系统，这样你在组织微型课堂或者工作坊，无暇顾及学生们的机器时就非常便捷了。

将你的notebook存储在像dropbox这样的网站上，然后把链接放在nbviewer（http://nbviewer.jupyter.org/），nbviewer可以呈现任意来源的notebook。

用菜单File > Download as > PDF 保存notebook为PDF文件。如果你选择本方法，我强烈建议你读一读Julius Schulz的文章（http://blog.juliusschulz.de/blog/ultimate-ipython-notebook）

用Pelican从你的notebook创建一篇博客（https://www.dataquest.io/blog/how-to-setup-a-data-science-blog/）。