# IPython Notebook 简介

目录[-]

 [基础](http://my.oschina.net/lionets/blog/274760#OSC_h1_1)

 [Tab 键自动完成](http://my.oschina.net/lionets/blog/274760#OSC_h3_2)

 [内省](http://my.oschina.net/lionets/blog/274760#OSC_h3_3)

 [魔术命令](http://my.oschina.net/lionets/blog/274760#OSC_h3_4)

 [历史输入和输出变量](http://my.oschina.net/lionets/blog/274760#OSC_h3_5)

 [Notebook](http://my.oschina.net/lionets/blog/274760" \l "OSC_h1_6)

 [演示文档](http://my.oschina.net/lionets/blog/274760#OSC_h3_7)

 [操作指南](http://my.oschina.net/lionets/blog/274760#OSC_h3_8)

在 IPython 的官网（ipython.org）上，介绍其的第一句话便是：IPython 是一个加强版的交互式 Shell。另外很多介绍 IPython 的文章也常以这句话开头，但这句话实在是等于没说。让 IPython 在各种 shell（IDE） 中脱颖而出，成为科学计算标配的，并不是按 Tab 键代码补完、以 % 开头的魔术命令这些大家都有的东西，而是与 matplotlib 这个数据可视化（绘图）包的深度集成以及奇妙的 Notebook。

IPython 较早的富 GUI 实现应该是 Qt Console。过去在标准 shell 里绘图时，弹出的绘图窗口会接管 shell 会话的控制权，你想继续输入命令就必须先把绘图窗口关掉。这对于希望同时实现可视化和交互式过程的数据分析用户来说显然是难以忍受的，因此 Qt console 站出来解决了这个问题。在 Qt console 中通过 matplotlib 绘制的图形会独立嵌于控制台中，并不影响你继续输入命令。

2011年，由 Brian Granger 领导的 IPython 团队开始开发一种**基于Web技术的交互式计算文档格式**，即 IPython Notebook。为什么说它是文档格式，而非计算工具呢？实际上它两者都是。Notebook 在交互上使用了 C/S 结构，它通过 Tornado 建立一个 shell 服务器，并使用浏览器作为客户端。另外 notebook 页面都被保存为 .ipynb的类 JSON 文件格式。这种文件格式也是 Notebook 最吸引人的地方。

IPython Notebook 更详细的介绍在下面第二节中给出，第一节讲的是一些对 IPython 的各种实现通用的功能。

## 基础

### Tab 键自动完成

和其他 IDE 差不多，自己多试试就好了

### 内省

在变量名或命令的前面或后面加一个 “？” 并执行，可以用于显示该对象的一些通用信息，如对象类型、文档字符串等，这就叫做**对象内省**。这种操作查看到的信息，尤其是函数和类的信息，比通常直接引用变量名然后回车所看到的（\_\_repr\_\_）要好。“？” 的另一个用法是可以搜索 IPython 的命名空间，配合通配符使用效果如下：

In [1]:import numpy as np

In [2]:np.\*load\*?

np.load

np.loads

np.loadtxt

np.pkgload

使用双问号“??”还可以查看对象的源代码（如果可见的话）。

### 魔术命令

在 IPython 的会话环境中，所有文件都可以通过 %run 命令来当做脚本执行，并且文件中的变量也会随即导入当前命名空间。即，对于一个模块文件，你对他使用 %run 命令的效果和 from module import \* 相同，除非这个模块文件定义了 main 函数（if \_\_name\_\_ == '\_\_main\_\_:'），这种情况下 main 函数还会被执行。

这种以 % 开头的命令在 IPython 中被称为魔术命令，用于加强 shell 的功能。常用的魔术命令有：

|  |  |
| --- | --- |
| %quickref | 显示 IPython 快速参考 |
| %magic | 显示所有魔术命令的详细文档 |
| %debug | 从最新的异常跟踪的底部进入交互式调试器 |
| %pdb | 在异常发生后自动进入调试器 |
| %reset | 删除 interactive 命名空间中的全部变量 |
| %run *script.py* | 执行 *script.py* |
| %prun *statement* | 通过 cProfile 执行对 *statement* 的逐行性能分析 |
| %time *statement* | 测试 *statement* 的执行时间 |
| %timeit *statement* | 多次测试 *statement* 的执行时间并计算平均值 |
| %who、%who\_ls、%whos | 显示 interactive 命名空间中定义的变量，信息级别/冗余度可变 |
| %xdel *variable* | 删除 *variable*，并尝试清除其在 IPython 中的对象上的一切引用 |
| !*cmd* | 在系统 shell 执行 *cmd* |
| output=!*cmd args* | 执行*cmd* 并赋值 |
| %bookmark | 使用 IPython 的目录书签系统 |
| %cd *direcrory* | 切换工作目录 |
| %pwd | 返回当前工作目录（字符串形式） |
| %env | 返回当前系统变量（以字典形式） |

对魔术命令不熟悉的话可以通过 %magic 查看详细文档；对某一个命令不熟悉的话，可以通过 %cmd? 内省机制查看特定文档。值得一提的是，IPython 中使用 del 命令无法删除所有的变量引用，因此垃圾回收机制也无法启用，所以有些时候你会需要使用 %xdel 或者 %reset。

### 历史输入和输出变量

与标准 Shell 类似，IPython 中也可以通过 \_ 和 \_\_ 访问上一次和上上一次的输出。同时你肯定注意到了，IPython 中每一次的输入输出都有序号。访问历史 X 行输出的方法为：\_X；访问历史 X 行输入的方法为：\_iX。因为访问历史输出的使用概率较历史输入大很多，所以访问历史输出仅使用下划线加行号即可，同时为了区分，访问历史输入时需添加小写字母 “i”，代表 “in”。

In [24]:1+1

Out[24]:2

In [25]:\_i24

Out[25]:'1+1'

In [26]:\_24

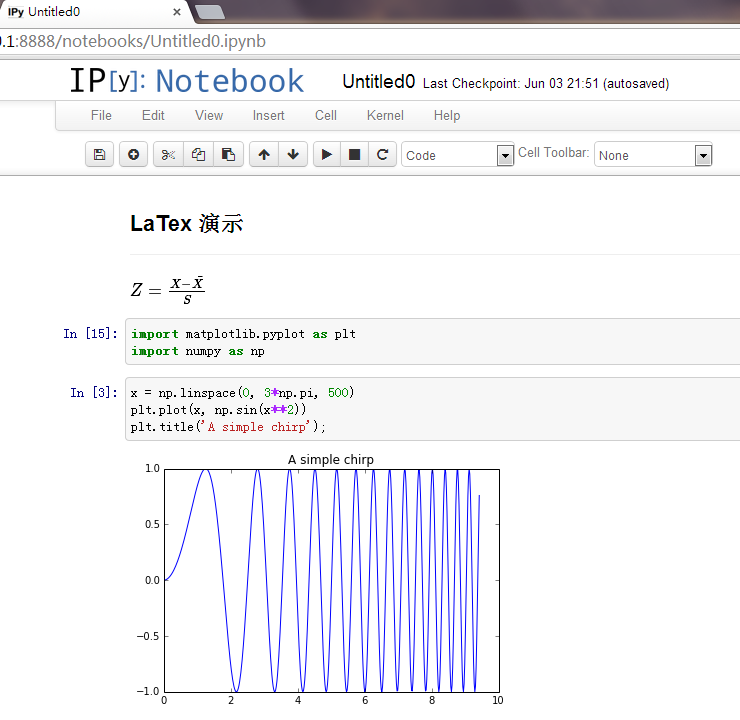
Out[26]:2

## Notebook

Notebook 的官方信息可以从 ipython.org/notebook 获得，不过好像被墙了。上不去的话也可以访问它的 github 页面，ipython/examples/Notebook 目录下有很多可供参考的内容。前面说过 Notebook 有一种 .ipynb 的文件格式，当你打开这个例程目录下的某个文件后，就能体会到 Notebook 的奇妙之处了。

### 演示文档

目前在各种 Python 研讨会上，一种流行的演示手段就是使用 IPython Notebook，然后再将 .ipynb 文件发布到网上以供所有人查阅。除了前面说过的可以内嵌 matplotlib 绘图外，Notebook 还同时提供了对 LaTex 和 MarkDown 的支持！

[](http://static.oschina.net/uploads/space/2014/0606/110718_lOcd_660175.png)

如上图便展示了一个 .ipynb 文件的示例页面。其中一对 In Out 会话被视作一个单元，称为 cell。第一个 cell 里我写入的内容其实是：

##LaTex 演示

---

$Z=\frac{X-\bar{X}}{S}$

分别使用了 MarkDown 和 LaTex 的语法。按下 Shift + Enter 后这段内容就被渲染成了图片中的样子。

cell 特别亲切的地方在于：它可以作为一个类似“段落”的概念来进行编辑，不管是执行前还是执行后，而且既可以针对内容进行编辑，也可以对 cell 整体应用 copy、paste、cut 等操作，甚至还可以前后移动 cell 的位置。这带来的好处是，在大量试验性的交互操作过后，他不会像普通 shell 那样留下无数没用的 IO 内容。如果某条命令的输出不理想或者报了错，你就可以回头编辑后重新运行，或把它移位或干脆删掉。这样在很久的一段交互过程后，Notebook 留下的反而是一份干净整洁的文档。

.ipynb 文件使用的是一种类 JSON 的文本格式，就像这样：

"worksheets": [

{

"cells": [

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"##LaTex \u6f14\u793a\n",

"---\n",

"$Z=\\frac{X-\\bar{X}}{S}$"

]

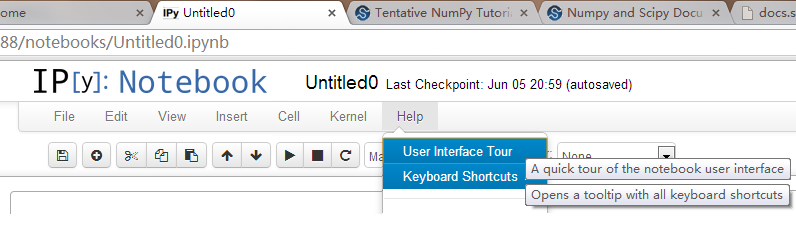
},

这种交互式计算环境，对绘图、数学公式、简易排版语法的支持，还有一种方便的文档格式，共同帮助 Notebook 成为了 Python 科学计算的理想工具。另外这种使用浏览器和服务进程的 C/S 结构还暗含了一种远程连接可能，抛开安全性不谈的话，其在教学方面也有很高的潜力。

### 操作指南

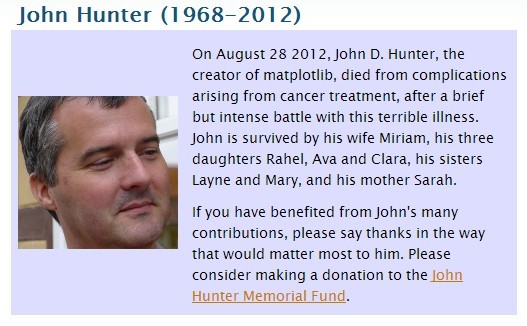
当通过 IPython Notebook.exe 进入应用时，首先打开的是 Home 页面，地址一般为：http://127.0.0.1:8888/tree。Home 目录下会列出所有的历史文件记录，右上角则有一个 “New Notebook” 按钮可以新建一个会话。

在 .ipynb 文件的交互页面，需要注意的除了标准的 IPython 语法外，就是页面抬头处的菜单栏和工具栏了。Notebook 很贴心地提供了非常友好的帮助页面，因此本文不再赘述。基本看完这两个页面（内容很少）后就能对 Notebook 的操作方式了解的差不多。

[](http://static.oschina.net/uploads/space/2014/0606/110735_PIs4_660175.png)

# [ipython notebook](http://www.cnblogs.com/zhuangzhuang1988/archive/2012/11/19/2777195.html)

先说下IPython的创始人挂了,缅怀下吧.他同时也是另一个NX项目的作者matplotlib项目的作者.



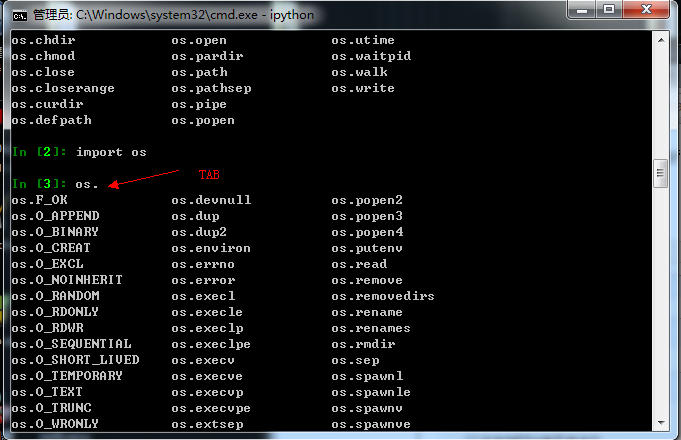
来自 <http://matplotlib.org/>

下面就说IPython把.

相信很多人都用过Python,也用命令行下敲过print "hello xxx",的东西的.

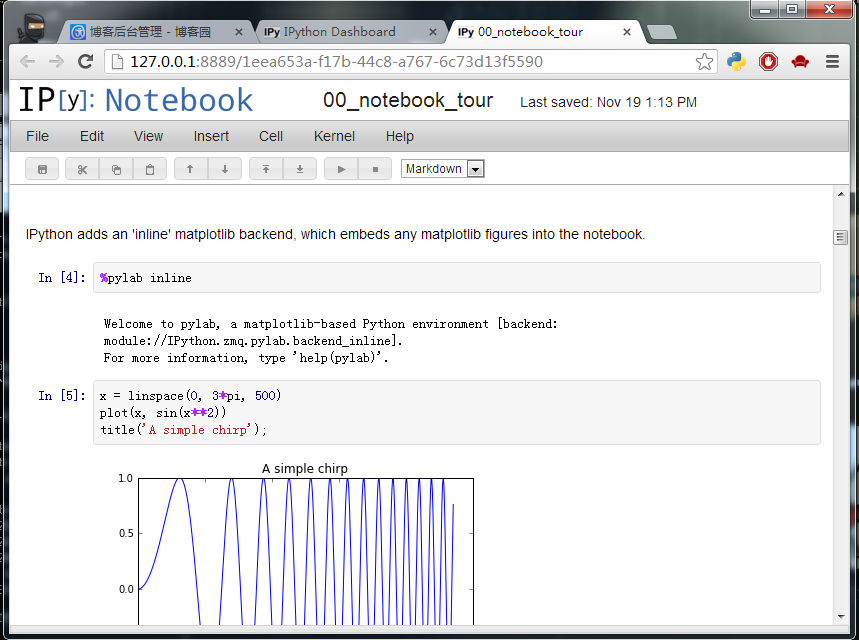
IPython乃是一款增强的Python命令行工具.还有个好像叫BPython的.(先不理他).

IPython超强优点.(自动补全)功能.



好了.先pass.(下面的自己看help)主要还是推荐IPython的另一个超强开发(演示)环境notebook.

先上图:



嗯.绝对NX.

先说安装吧.

首选安装Python 2.6/2.7版本的. (3.x的暂时不推荐,不是不好,而是因为好多的库都不兼容)

在windows还得装个pyreadline (补全用的).

pyzmq:并发用的,是基于消息的网络模型的东东(ZeroMQ) 单独玩也是很好玩的

tornado:Http服务器.异步xx啥的.单独玩也好玩.

MathJax:渲染用的可以将公式直接渲染出来 很好看的.(如果没有安装的话,它自动使用在线版本的).不过推荐还是使用离线版本的(速度快嘛.) 安装这个的话得这样.

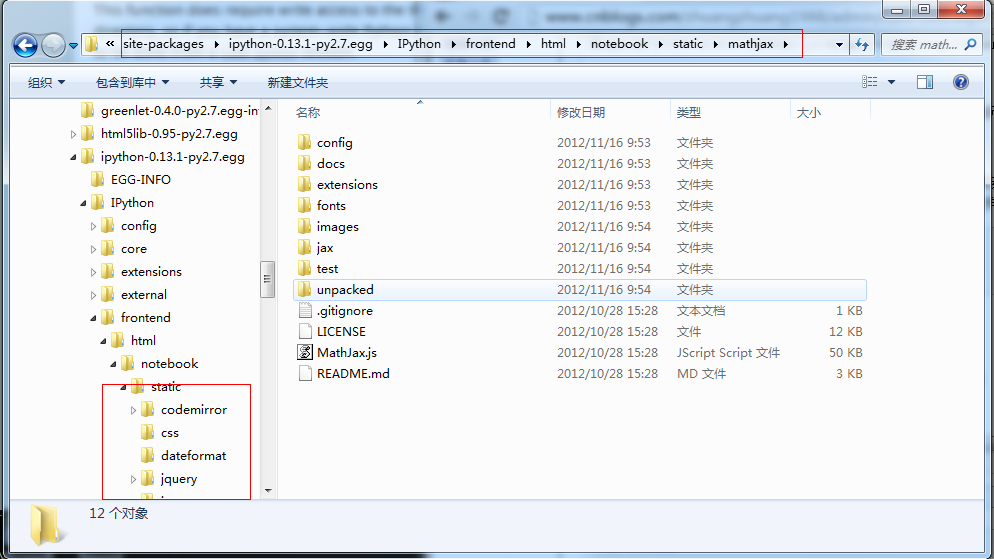
from IPython.external.mathjax import install\_mathjax

install\_mathjax()

看上去还是挺麻烦的.翻开代码一看直接把MathJax下载下来.

默认使用的v1.1版本的我下载的是2.1的 地址在<https://github.com/mathjax/MathJax/archive/v2.1.zip>

下载完解压到特定的路径下就好了.注意下.



还有个最重要的.浏览器.ipython使用了websocket,它是不支持IE的.一般Chrome就好了.更多浏览器看,<http://ipython.org/ipython-doc/rel-0.13.1/install/install.html#dependencies-for-the-ipython-html-notebook> 这里就行了.

安装完后设置下path. `C:\Python27\Scripts`

启动很简单. 打开命令行,切换到某个目录下,输入ipython notebook.它会启动服务器.并打开浏览器.

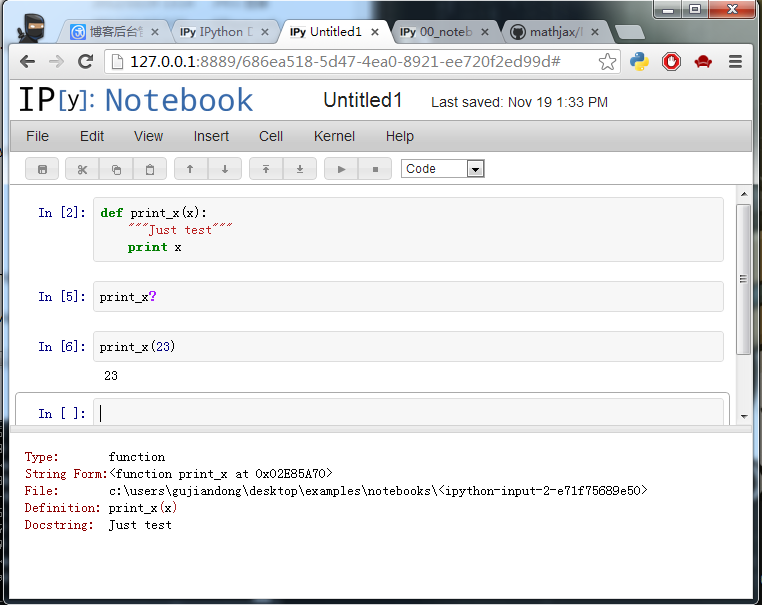
如下:



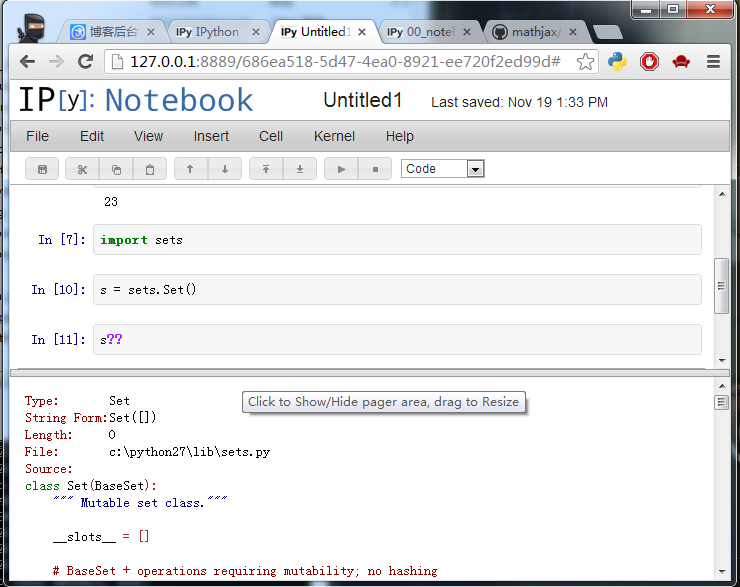
它会自动读取该目录下面的.ipynb文件.并显示.

如果要新建一个文件的话 点按钮`New notebook`就好了.

在新的网页中可以输入简单的Python命令测试.



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.如图.

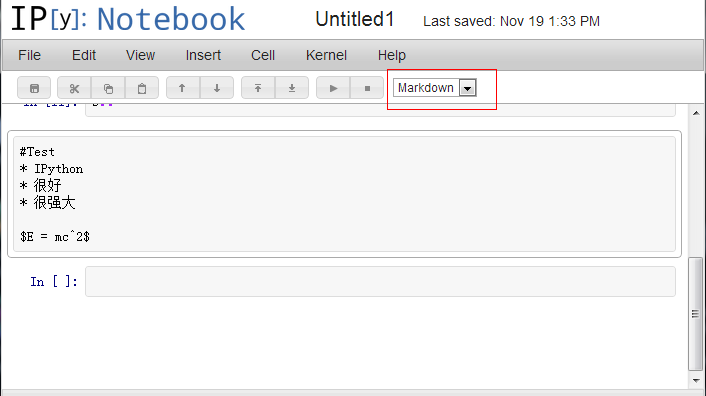


在试试它的.别的功能.

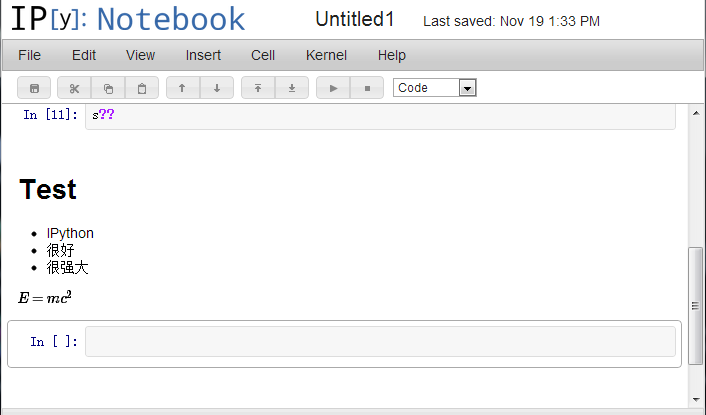
忘了说下IPython notebook是跟mathematica 环境相似的.

可以是用Markdown写些说明啥的,不过得先选择下Markdown.

渲染前:

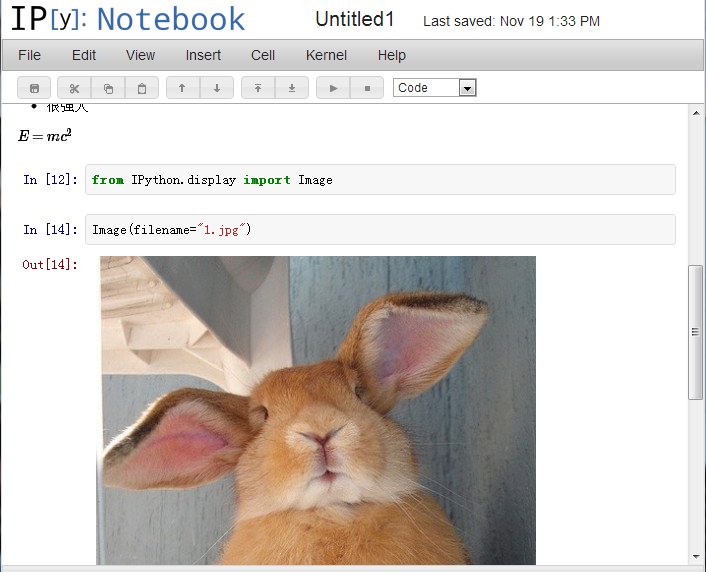


渲染后:



基本上是即时显示的.

显示图片:



Image函数也可以使用网上的图片的.

使用画图功能,使用前先得安装几个重要的包.

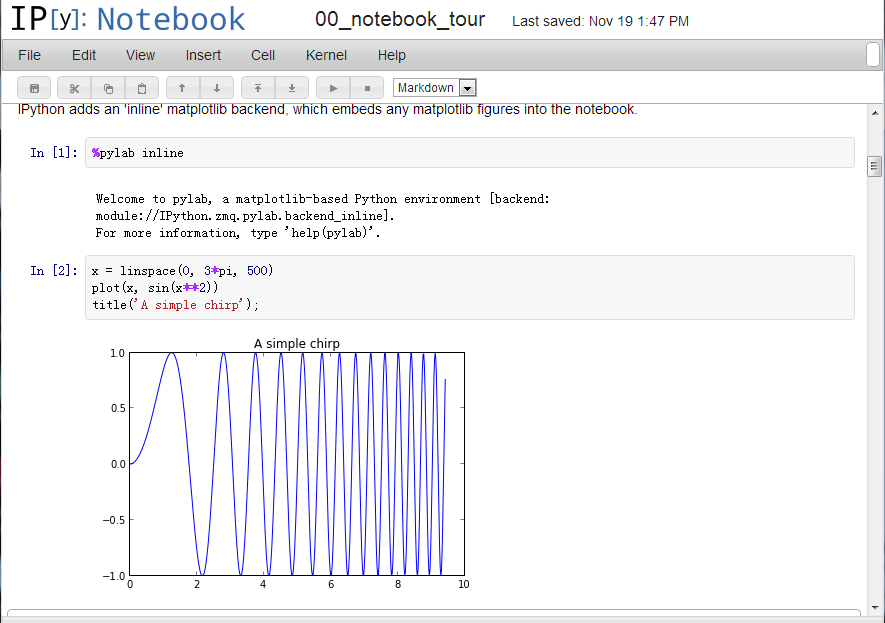
numpy+scipy+mathplotlib这三个加起来可以抵得上1/n个matlab了.(n不好说).

如果要在python中使用符号运算的话还可以考虑sympy库.(额,从来没有超过1.0的库).不过ipython notebook 也提供了支持.

使用时 可以在命令行下输入 ipython notebook --pylab inline.这样的话它就会导入pylab(mathplotlib)库.而且显示图片时可以嵌在网页中的.

当然也可以在网页中输入 %pylab inline启动这个功能的.

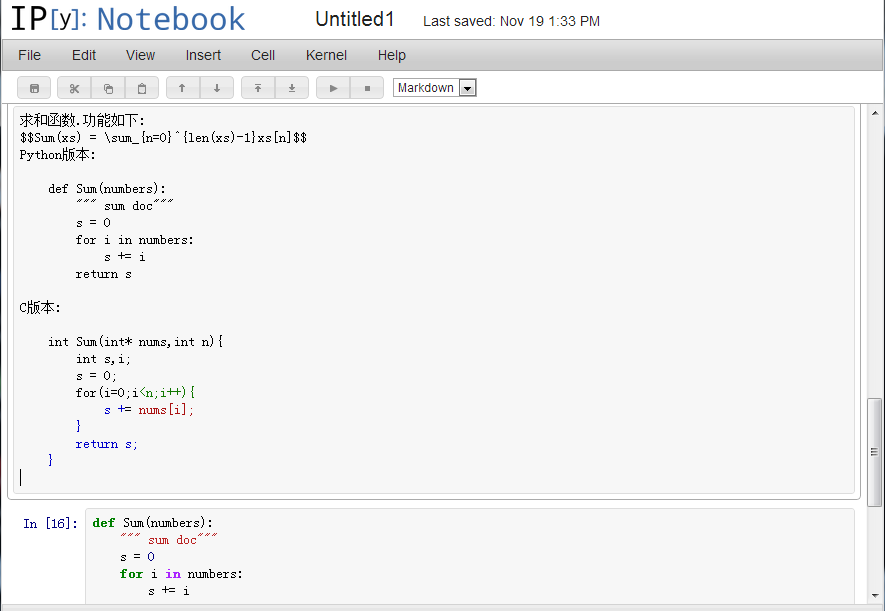
效果如下:



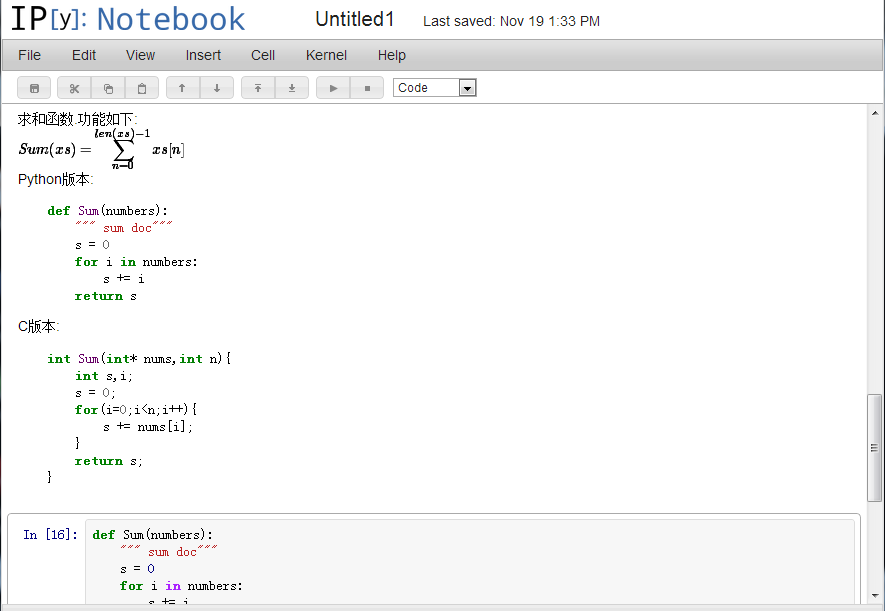
忘了还有个.

在markdown中是支持代码着色的.不过不知道用的是哪种着色方案,但是够用了.(太花了容易影响思考的,不过比豆瓣好些,豆瓣也算是做到了极简).

渲染前:



渲染后:

很漂亮吧! 耶.

当然这只是最简单的应用了.

IPython还支持并行运算(parallel)啥的.

更多例子<https://github.com/ipython/ipython/tree/master/docs/examples> .