# Jupyter notebook快速入门教程



本篇将给大家介绍一款超级好用的工具：Jupyter notebook。

为什么要介绍这款工具呢？

如果你想使用Python学习数据分析或数据挖掘，那么它应该是你第一个应该知道并会使用的工具，它很容易上手，用起来非常方便，是个对新手非常友好的工具。而事实也证明它的确很好用，在数据挖掘平台 Kaggle 上，使用 Python 的数据爱好者绝大多数使用 jupyter notebook 来实现分析和建模的过程，因此，如果你想学习机器学习，数据挖掘，那么这款软件你真的应该了解一下。

本篇博主总结了关于Jupyter notebook的一些关键点，帮助大家快速了解并使用它。

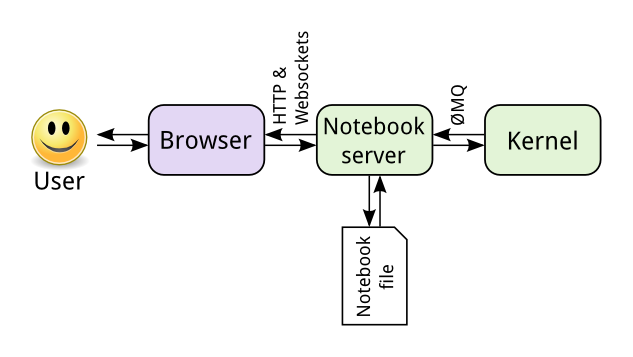
## **什么是Jupyter notebook？**

Jupyter notebook 是一种 Web 应用，它能让用户将说明文本、数学方程、代码和可视化内容全部组合到一个易于共享的文档中，非常方便研究和教学。在原始的 Python shell 与 IPython 中，可视化在单独的窗口中进行，而文字资料以及各种函数和类脚本包含在独立的文档中。但是，notebook 能将这一切集中到一处，让用户一目了然。

Jupyter notebook特别适合做数据处理，其用途可以包括数据清理和探索、可视化、机器学习和大数据分析。

## **Jupyter notebook是如何工作的？**

Jupyter notebook 源于 Fernando Perez 发起的 IPython 项目。IPython 是一种交互式 shell，与普通的 Python shell 相似，但具有一些更高级的功能，例如语法高亮显示和代码补全，还有一些 magic 操作，十分方便。Jupyter notebook 将 IPython 做成了一种 Web 应用，我们可以通过它的基本架构更清楚的了解：



可以看到，这里的核心是 notebook 的服务器。用户通过浏览器连接到该服务器，而 notebook 呈现为 Web 应用。用户在 Web 应用中编写的代码通过该服务器发送给内核，内核运行代码，并将结果发送回该服务器。然后，任何输出都会返回到浏览器中。保存 notebook 时，它将作为 JSON 文件（文件扩展名为 .ipynb）写入到该服务器中。

此架构的一个优点是，内核无需运行 Python。由于 notebook 和内核分开，因此可以在两者之间发送任何语言的代码。例如，早期的两个非 Python 内核分别是 R 语言和 Julia 语言。使用 R 内核时，用 R 编写的代码将发送给执行该代码的 R 内核，这与在 Python 内核上运行 Python 代码完全一样。IPython notebook 已被改名，因为 notebook 变得与编程语言无关。新的名称 Jupyter 由 Julia、Python 和 R 组合而成。

## **安装Jupyter notebook**

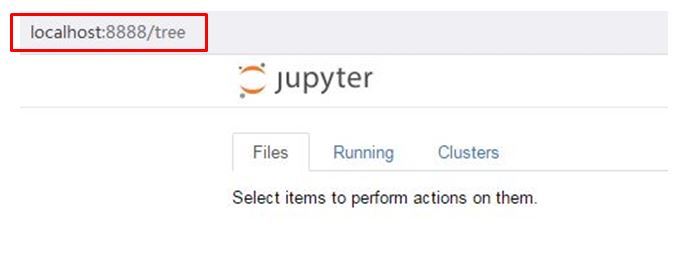
最简单的方法就是使用 Anaconda，其发行版附带了 Jupyter notebook。

在 conda 环境下安装 Jupyter notebook 可以使用 conda install jupyter notebook。当然，也可以通过 pip 来安装 pip install jupyter notebook。

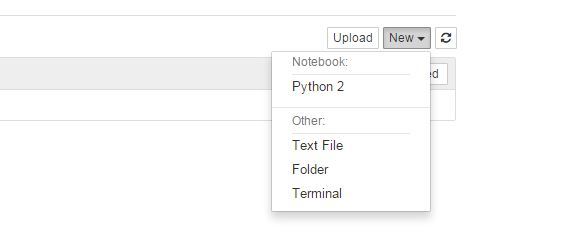
## **启动 notebook 服务器**

启动 notebook 很简单，只需要在终端环境下输入 jupyter notebook， 服务器就会在当前操作的目录下启动。当然你可以建立一些专门用来运行 notebook 的文件夹，尤其对于不同的 Python版本以及一些项目（后面会提到）。

启动后，默认的 notebook 服务器的运行地址是 http://localhost:8888。只要 notebook 服务器仍在运行，你随时都能通过在浏览器中输入 http://localhost:8888 返回到 web 页面中。



可以通过点击 “New” 创建新的 notebook、文本文件、文件夹或终端。



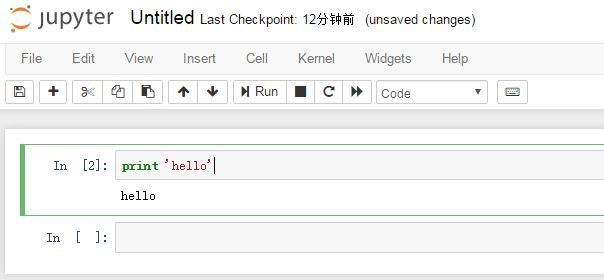
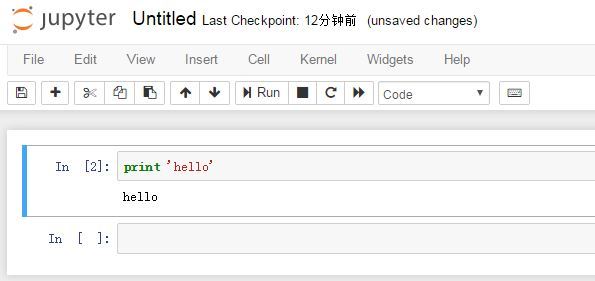
“Notebook”下的列表显示了已安装的内核。这个示例中可以看到版本是 Python 2.7，因此列出了 Python 2.7 内核。当然，如果你同时安装了其它内核比如 Python 3，那么它也会在列表中同时出现。这又是 notebook 的另一大好处，就是可以同时管理多个版本，当你同时需要 Python 2 和 Python 3，或者 Scala 2.10 和 2.11 的内核的时候，是十分方便的。

而对于关闭 notebook，可以这样操作：

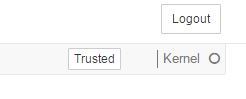


## **notebook 界面**

notebook 界面由基本的单元格组成，每个单元格在可编辑状态下可以任意的输入代码和注释说明（markdown）。默认的是代码格式，也就是下图中所示的 code。

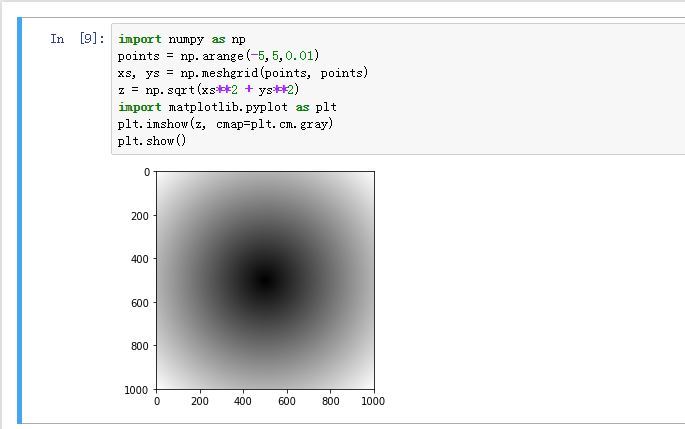
单元格绿色代表内容可编辑状态，蓝色代表单元格可操作状态（比如删除单元格，必须回到蓝色），而蓝色与绿色之间可以用 Esc 和 Enter 来切换。



Kernel 的小圆圈在空闲状态下是空的，而当运行代码时，会被填满，所以可以通过观察 Kernel 的状态观察程序是否运行完成。

## **代码单元格**

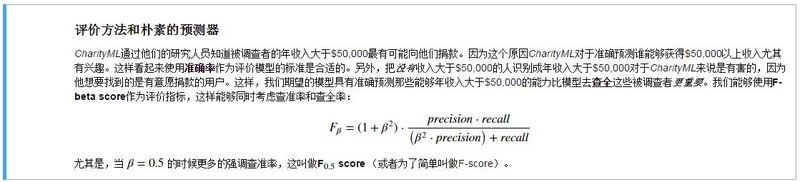
notebook 中的大部分工作均在代码单元格中完成。编写和执行代码都在这里，就像我们平时在 IDE 软件里敲代码一样，给变量赋值、定义函数和类、导入包等。执行单元格代码可以通过 Shift + Enter 来完成。下面是一个示例：



## **Markdown 单元格**

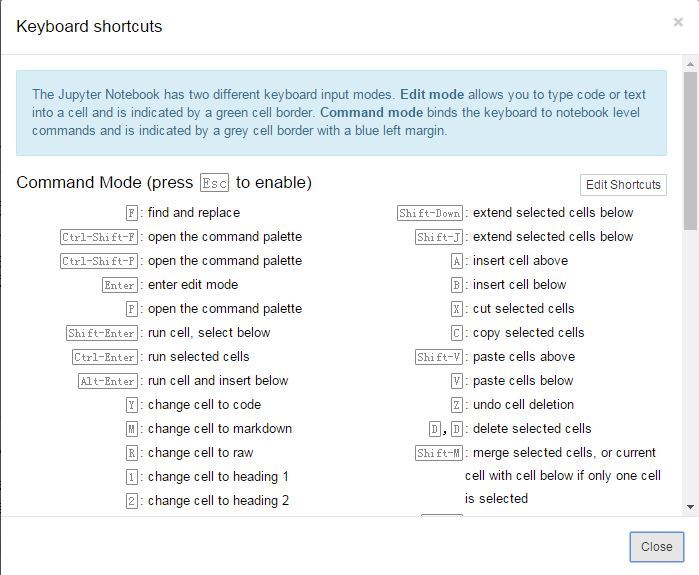
Markdown 是格式化语法，可以加入链接、将文本样式设为粗体或斜体和设置代码格式。像代码单元格一样，按 Shift + Enter 或 Ctrl + Enter 可运行 Markdown 单元格，这会将 Markdown 呈现为格式化文本。

Markdown 在这里就不详细介绍了，如果不清楚可以查看官网：<http://www.markdown.cn/>。下面是一个 Markdown 的示例：



## **快捷键**

notebook 自带一组快捷键，能让你快速使用键盘与单元格交互，而无需使用鼠标和工具栏。熟悉这些快捷键需要花费一点时间，但如果能熟练掌握，将大大加快你在 notebook 中的工作速度。所有的快捷键就不在这里展示了，因为这些快捷键可以通过单元格 蓝色 状态下按 "h" 来查看：

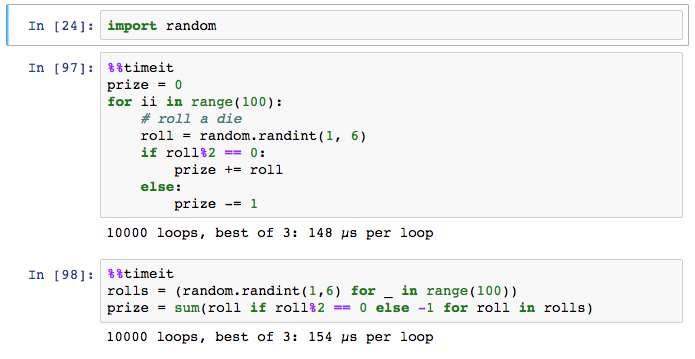


## **Magic 关键字**

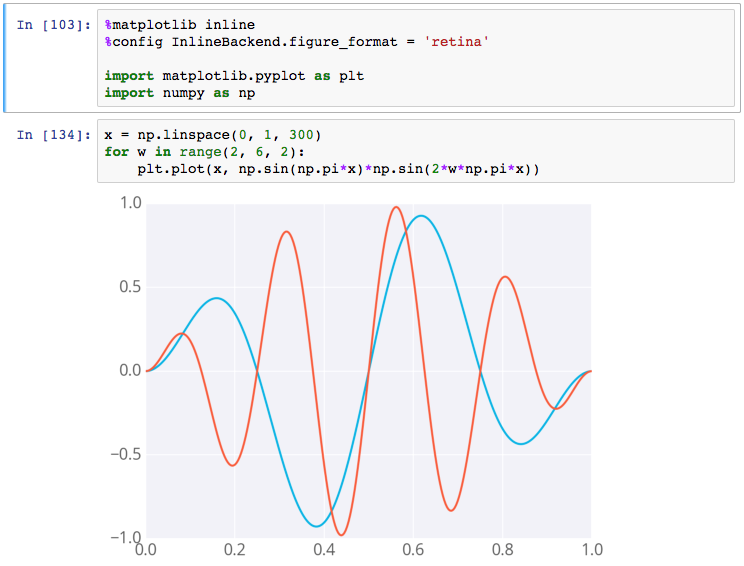
Magic关键字是 IPython 的一些高级用法，可以运行特殊的命令，然后控制 notebook。例如，在 notebook 中可以使用 %matplotlib 将 matplotlib 设置为以交互方式工作。

Magic 命令的前面带有一个或两个百分号（% 或 %%），分别代表行 Magic 命令和单元格 Magic 命令。行 Magic 命令仅应用于编写 Magic 命令时所在的行，而单元格 Magic 命令应用于整个单元格。

如果要测算整个单元格的运行时间，请使用 %%timeit，如下所示：



如果要在 notebook 中嵌入可视化内容，可以说使用 %matplotlib inline，如下所示:



默认情况下，图形呈现在各自的窗口中。但是，你可以通过命令传递参数，以选择特定的“后端”（呈现图像的软件）。要直接在 notebook 中呈现图形，应将通过命令 %matplotlib inline 内联后端一起使用。

提示：在分辨率较高的屏幕（例如 Retina 显示屏）上，notebook 中的默认图像可能会显得模糊。可以在 %matplotlib inline 之后使用 %config InlineBackend.figure\_format = 'retina' 来呈现分辨率较高的图像。

当然，还有很多 Magic 关键字的使用，这里只列出了两个常用的介绍给大家。更多内容请参考：[http://ipython.readthedocs.io...](http://ipython.readthedocs.io/en/stable/interactive/magics.html)

## **总结**

通过本篇，相信你已经了解并知道如何使用 Jupyter notebook 了，但是还需要一些实际的操作才能更熟练的使用它，包括一些快捷键的使用，公众号后台回复 "notebook快捷键" 可获取一份练习快捷键的辅助材料。

原文链接：<https://segmentfault.com/a/1190000014831082>

一、什么是jupyter

1、简介：

jupyter notebook是一种 Web 应用，能让用户将说明文本、数学方程、代码和可视化内容全部组合到一个易于共享的文档中。它可以直接在代码旁写出叙述性文档，而不是另外编写单独的文档。也就是它可以能将代码、文档等这一切集中到一处，让用户一目了然。

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。Jupyter Notebook 已迅速成为数据分析，机器学习的必备工具。因为它可以让数据分析师集中精力向用户解释整个分析过程。我们可以通过Jupyter notebook写出了我们的学习笔记。但是jupyter远远不止支持上面的三种语言，目前能够使用的语言他基本上都能支持，包括C、C++、C#，java、Go等等。

jupyter notebook和我们前面所讲的两篇系列文章ipython其实都是来自同一个产品族，它的前身叫做ipython notebook，至于后面为什么更名这不得而知，这也就是为什么很多文章总是默认将ipython就说成是ipython notebook的原因了。但是既然已经更名了，我们还是区别对待ipython和jupyter notebook。

2、安装与打开

安装这里就不说了，使用过anaconda的人都清楚，在安装anaconda的时候会一起打包安装，如果想要自己安装，可以通过pip或者是conda安装，

怎么打开也很简单，可以直接在anaconda的菜单里面打开jupyter notebook，也可以通过命令行，输入jupyter notebook或者是jupyter-notebook都可以。

我们说了，jupyter notebook会在浏览器中中打开，是一种Web应用，故而有web路径和端口号了，我的打开之后，在浏览器的地址栏会显示如下：

http://localhost:8888/tree

当一次打开多个jupyter notebook的时候，端口号会依次递增8889，8890依次递增。

3、jupyter的作用

前面说了一大堆，也不清楚jupyter到底有啥好的啊，到底有一些什么功能呢？

在介绍 Jupyter Notebook 的功能之前，让我们先来看一个概念：文学编程 ( Literate programming )，这是由 Donald Knuth 提出的编程方法。传统的结构化编程，人们需要按计算机的逻辑顺序来编写代码；与此相反，文学编程则可以让人们按照自己的思维逻辑来开发程序。

简单来说，文学编程的读者不是机器，而是人。 我们从写出让机器读懂的代码，过渡到向人们解说如何让机器实现我们的想法，其中除了代码，更多的是叙述性的文字、图表等内容。这么一看，这不正是数据分析人员所需要的编码风格么？不仅要当好一个程序员，还得当好一个作家。那么 Jupyter Notebook 就是不可或缺的一款集编程和写作于一体的效率工具。

以下列举了 Jupyter Notebook 的众多优点：

（1）极其适合数据分析，想象一下如下混乱的场景：你在终端中运行程序，可视化结果却显示在另一个窗口中，包含函数和类的脚本存在其他文档中，更可恶的是你还需另外写一份说明文档来解释程序如何执行以及结果如何。此时 Jupyter Notebook 从天而降，将所有内容收归一处，你是不是顿觉灵台清明，思路更加清晰了呢？

（2）支持多语言，也许你习惯使用 R 语言来做数据分析，或者是想用学术界常用的 MATLAB 和 Mathematica，这些都不成问题，只要安装相对应的核（kernel）即可。这里列出了 Jupyter 支持的所有语言，供您参考。

分享便捷，支持以网页的形式分享，GitHub 中天然支持 Notebook 展示，也可以通过 nbviewer 分享你的文档。当然也支持导出成 HTML、Markdown 、PDF 等多种格式的文档。

（3）远程运行，在任何地点都可以通过网络链接远程服务器来实现运算

（4）交互式展现，不仅可以输出图片、视频、数学公式，甚至可以呈现一些互动的可视化内容，比如可以缩放的地图或者是可以旋转的三维模型。这就需要交互式插件（Interactive widgets）来支持，更多内容请参考这里。

常见的一些jupyter高级应用如：

数学公式编辑

幻灯片制作

魔术关键字

二、jupyter notebook的简单使用

1、常用的jupyter notebook一些命令如下：指的是在cmd中使用jupyter命令

（1）查看jupyter notebook的相关帮助

jupyter-notebook --help

jupyter-notebook --help-all （会显示更详细的信息）

jupyter-notebook -h （只有一个横杠噢，熟悉命令使用的小伙伴都知道）

（2）jupyter notebook常用的子命令——subcommand

list ：列出当前的所打开的jupyter notebook的一些信息，比如：

(base) C:\Users\lenovo>jupyter-notebook list

Currently running servers:

http://localhost:8888/?token=5751bd2f4a5675c59c15c9cdfaa7ecbe431ba68b7d23a3c7 :: C:\Users\lenovo

http://localhost:8889/?token=a9f1a6a33a5b0fa93f51552aecbb99a3dec321fdd5e0e74a :: C:\Users\lenovo

 这里我打开了两个jupyter notebook，一个端口为8888，一个为8889

stop：关闭所给定的端口号的那一个jupyter

(base) C:\Users\lenovo>jupyter-notebook stop 8888

Shutting down server on port 8888 ...

(base) C:\Users\lenovo>jupyter notebook list

Currently running servers:

http://localhost:8889/?token=a9f1a6a33a5b0fa93f51552aecbb99a3dec321fdd5e0e74a :: C:\Users\lenovo

关闭了8888，只剩下一个8889了。

password ：给某一个打开的jupyter notebook 服务设置密码，后面直接输入所要添加的密码即可。

（3）jupyter notebook命令的常用选项，下面进介绍一些常用的，想要知道所有的详细使用可以使用

jupyter notebook --help-all 查看jupyter命令的详细信息：

--generate-config  产生默认的配置文件()这个是重点，后面会讲到）

--no-browser 启动jupyter notebook之后不打开浏览器（默认情况下是会打开一个浏览器界面的）

--pylab 同前面讲解ipython的时候很类似，在jupyter notebook里面集成，也可以在jupyter notebook里面使用

            %pylab 或者是%matplotlib魔术命令（这是最常用的，后面也会讲到）

--config=<Unicode>  制定一个完全路径的配置文件名称，关于配置文件后面会详解，默认是default，即系统指定的默认配置文件

--ip=<Unicode>  默认是: 'localhost'，从前面的现实中也可以看出来，表示的是notebook服务器会监听的IP地址，我们也可以手动指定

--port=<Int>  指定jupyter notebook打开浏览器的端口号，默认是: 8888，也可以手动输入

--notebook-dir=<Unicode> 默认使用默认的目录，表示的是notebook和kernels的目录

--browser=<Unicode> 还可以指定某一个特定的浏览器打开，默认使用系统默认的浏览器打开，是可以自己选定浏览器的。

2、jupyter的简单使用

当我们打开jupyter notebook（不管用什么样的方式打开，使用菜单打开或者是命令行打开是一样的）会在默认的浏览器中看到这样的界面：

但是每一台不同的电脑打开之后的界面是不同的，仔细观察就会发现，这里面现实的一些文件夹就是电脑用户目录下面的文件夹，这是为什么呢？那是因为jupyter notebook在启动的时候总是有一个默认的目录，一般情况下，使用户的目录。

那怎么查看默认目录，或者是我想要修改默认目录该怎么做呢？

实际上，jupyter的一些默认设置都设置在一个默认的配置文件中，我们可以查看并且修改配置文件中的内容

（1）查看配置文件

在cmd中使用如下命令：jupyter-notebook --generate-config

(base) C:\Users\lenovo>jupyter-notebook --generate-config

Writing default config to: C:\Users\lenovo\.jupyter\jupyter\_notebook\_config.py

可见我的默认配置文件在：C:\Users\lenovo\.jupyter\jupyter\_notebook\_config.py中，现在我们查看配置文件的信息。得到如下：

打开之后发现，这本质上应该是一个python的 py文件，但是实际上全部都是用井号#开头的，这不是意味着这都是一些注释吗？可以这么说，但实际上观察会发现，有的有两个井号开头 ##，这才是注释文本，而那些以一个井号#开头的实际上就是默认的配置信息，也就是我们要修改的。当然这个文件很长，我不会一个一个讲解每一个配置信息是什么意思，我指将如何自定义默认目录。

我们通过编辑器，搜索directory定位到如下位置：

## The directory to use for notebooks and kernels.

#c.NotebookApp.notebook\_dir = ''

这句话就是设置jupyter的目录的，可见默认值是没有的，我们现在比如要使它指定到我们自己的目录，比如我要让它指定到F盘中的myjupyter文件夹，则修改如下：

## The directory to use for notebooks and kernels.

c.NotebookApp.notebook\_dir = 'F:\myjupyter'

保存修改的文件并关闭！然后再打开，发现如下信息，从下面的信息可以看见这样一句话：

Serving notebooks from local directory: F:\myjupyter

这就说明修改成功。

(base) C:\Users\lenovo>jupyter-notebook

[I 14:17:57.432 NotebookApp] JupyterLab beta preview extension loaded from D:\ProgramData\Anaconda3\lib\site-packages\jupyterlab

[I 14:17:57.432 NotebookApp] JupyterLab application directory is D:\ProgramData\Anaconda3\share\jupyter\lab

[I 14:18:00.231 NotebookApp] Serving notebooks from local directory: F:\myjupyter

[I 14:18:00.232 NotebookApp] 0 active kernels

[I 14:18:00.232 NotebookApp] The Jupyter Notebook is running at:

[I 14:18:00.232 NotebookApp] http://localhost:8888/?token=ad2aa0097b0687bbd8b5b11909101533630e781e7c28753b

[I 14:18:00.232 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

[C 14:18:00.234 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,

to login with a token:

http://localhost:8888/?token=ad2aa0097b0687bbd8b5b11909101533630e781e7c28753b&token=ad2aa0097b0687bbd8b5b11909101533630e781e7c28753b

[I 14:18:00.786 NotebookApp] Accepting one-time-token-authenticated connection from ::1

注意事项：

第一、文件夹必须先创建好，比如这里F盘下面的myjupyter文件夹要先创建好；不然会jupyter初始化时会找不到目录

第二、要取消注释，c前面的#要去掉（我还不知道这是为什么？哪位知道望告知！）

第三、要注意文件名不可以是数字

除此之外，上面的这些操作，推荐全部使用anaconda自带的Anaconda Promp进行操作，因为它提供了非常好的环境管理功能，而在系统自身的cmd中操作，可能因为环境变量设置等相关的一些原因，不成功。比如我在多个环境中都安装了jupyter，但是cmd中环境变量所指向的那个是A，我们修改的可能是B的配置文件，这样通过cmd启动jupyter发现依然未改变。

现在打开，则发现web界面变成了如下：

三、jupyter运行环境的配置——一python运行环境为例

jupyter notebook本质上是一个web应用程序，我们可以在上面书写代码，但是代码本身的运行环境是需要自己安装的，没有运行环境，即使是在jupyter notebook里面书写的代码怡然没有办法运行。因为代码本身，web应用程序是不认识的。

幸运的是，在使用anaconda安装的时候，会默认将安装jupyter，而且会安装一个Python的运行环境，所以打开jupyter的时候，可以直接看见这个运行环境，在jupyter里面称之为内核kernel，如下所示：

在右上角点击下拉按钮，会得到如图所示的结果，

第一个python3 ，表示的就是默认的python3 kernel，它是随着anaconda一起安装的；

Text File ,表示的是新建一个文本文件

Folder ，表示的是新建一个文件夹

Terminal ，表示的是在浏览器中新建一个用户终端，即类似于cmd的shell。

但是，这是远远不够的，因为我不可能所有的程序都是使用这一个python kernel，使用过TensorFlow的人都知道如何创建一个新的运行环境，打包TensorFlow所依赖的各种包。本文以python为例，我还有另外两个环境，一个是TensorFlow，一个是pytorch，还有一个是，mxnet。他们都是使用conda创建的python运行环境。其实就一句话：

conda create -n tensorflow python=3.6

conda创建运行环境的方法这里就不详细说明了，可以参阅相关文章。注意：因为anaconda的服务器在国外，上面的执行速度实在是太慢，甚至等了半天，然后连接失败，所以，我们可以使用清华大学的镜像进行下载，在使用上面执行命令之前，添加一句：conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/

再使用：conda create -n tensorflow python=3.6

这会快非常多！！！

注意：

TUNA 还提供了 Anaconda 仓库的镜像，运行以下命令:

conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/

conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/

conda config --set show\_channel\_urls yes

即可添加 Anaconda Python 免费仓库。

直接粘贴复制以上代码即可。另外，要一行一行运行，并且每一行运行完没有任何结果，直接运行下一行即可。

三行代码运行完，可以运行 conda install numpy 测试一下吧。

1、原始的方法解决多个Python运行环境问题

就是走在每一个python的运行环境里面都使用conda install notebook 安装一个与之对应的notebook，然后我在做开发的时候，需要用什么环境，我就在每一个Python环境中的script文件夹下面打开对应的哪个jupyter notebook，这当然没问题。但这样做不是很高级，我需要安装很多次，而且每次在使用jupyter notebook的时候都只能使用一个环境，局限性很大。

2、更高级的办法

现在，比如我已经在anaconda里面创建了一个名为tensorflow的环境，但是，怎么把它添加到jupyter里面呢？

（1）方法一：

首先在anaconda prompt里面激活我们需要的环境，然后执行下面一个命令：

(base) C:\Users\lenovo>activate python27

(python27) C:\Users\lenovo>python -m ipykernel install --name python27

Installed kernelspec python27 in C:\ProgramData\jupyter\kernels\python27

和

(base) C:\Users\lenovo>activate tensorflow

(tensorflow) C:\Users\lenovo>python -m ipykernel install --name tensorflow

Installed kernelspec tensorflow in C:\ProgramData\jupyter\kernels\tensorflow

总结：两步走，

第一步：激活相应的环境

第二部分：执行命令 python -m ipykernel install --name 环境名称

现在打开我们的jupyter notebook，查看python环境如下：

补充：也有人用下面的方法：

先激活某一个Python环境：然后再执行下面两个语句：

conda install -n 环境名称 ipykernel

python -m ipykernel install --user

注意：上面这种方式都是可以的，但问题是还是需要针对每一个环境安装一次ipykernel，而且有时候不知怎么回事，总是会遇见一些乱七八糟的错误，我还没搞清楚是为什么，如果谁知道，望告知。

（2） 方法二——一步到位的方法

在我创建完我需要的运行环境之后，然后只需要在base运行环境中执行一个命令即可。

(base) C:\Users\lenovo>conda install nb\_conda

将会将所有的kernel全部添加进去，这种方法是最快的，而且最不容易出错，推荐使用。

（3）补充方法——综合前面两者

（1）一步到位：

(base) C:\Users\lenovo>conda install nb\_conda

(base) C:\Users\lenovo>conda install nb\_conda\_kernels

两种方法均可以

（2）分步完成：

第一步：激活某一个环境activate mxnet

第二步：在环境中安装ipykernel：pip install ipykernel

第三步：再执行命令：

         python -m ipykernel install --user --name mxnet --display-name mymxnet

后面的蓝色部分可以省略。第一个mxnet指的是我激活的那一个环境名，这里是mxnet；第二个mymxnet是我要在jupyter里面显示的内核名称，是自己自定义的，我定义为mymxnet。

全文总结：关于ipython系列的文章到这里就结束了，三篇文章，从ipython到jupyter notebook，希望对于一些有需要的小伙伴有所帮助，本来还打算写一篇关于jupyter notebook的使用的，但是这种教程网上实在太多了，就不在这里画蛇添足了。