**Jupyter notebook 是什么？**

欢迎学习本课程——如何使用 [**Jupyter**](http://jupyter.org/) notebook。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。例如，不久前我共享了我最爱的一个 Jupyter notebook ，它分析了 [**LIGO 实验**](https://www.ligo.caltech.edu/news/ligo20160211)探测到的**[两个碰撞的黑洞所发出的引力波](https://losc.ligo.org/s/events/GW150914/GW150914_tutorial.html" \t "_blank)**。你可以下载数据，运行 notebook 中的代码，重复整个分析，实际上等于你自己探测引力波！

Jupyter Notebook 已迅速成为处理数据的必备工具。其用途包括**[数据清理和探索](http://nbviewer.jupyter.org/github/jmsteinw/Notebooks/blob/master/IndeedJobs.ipynb" \t "_blank)**、可视化、**[机器学习](http://nbviewer.jupyter.org/github/masinoa/machine_learning/blob/master/04_Neural_Networks.ipynb" \t "_blank)**和**[大数据分析](http://nbviewer.jupyter.org/github/tdhopper/rta-pyspark-presentation/blob/master/slides.ipynb" \t "_blank)**。我为我的个人博客创建了**[一个 notebook 示例](https://github.com/mcleonard/blog_posts/blob/master/body_fat_percentage.ipynb" \t "_blank)**，它展示了 notebook 的许多特点。这项工作通常在终端中完成，也即使用普通的 Python shell 或 IPython 完成。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。

GitHub 上也直接支持 Jupyter notebook 的渲染。借助此出色的功能，你可以轻松地共享工作。**[http://nbviewer.jupyter.org/](http://nbviewer.jupyter.org/" \t "_blank)** 也会提供 GitHub 代码库中的 notebook ，以及存储在其他地方的 notebook。

**文学化编程**

notebook 是 Donald Knuth 在 1984 年提出的**[文字表达化编程](http://www.literateprogramming.com/" \t "_blank)**的一种形式。在文字表达化编程中，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。用 Donald Knuth 的话来说：

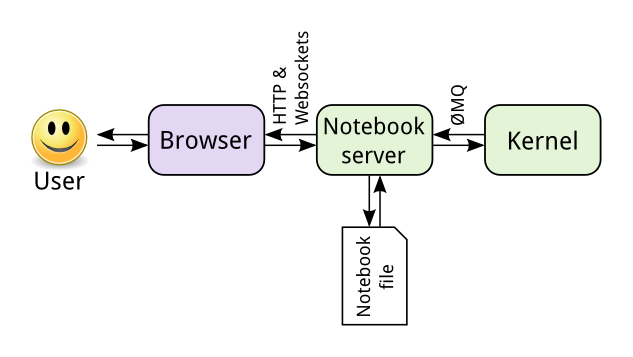
让我们集中精力向人们解释我们希望计算机做什么，而不是指示计算机做什么。

归根到底，代码是写给人看到，不是写给计算机看的。notebook 恰恰提供了这种能力。你能够直接在代码旁写出叙述性文档。这不仅对阅读 notebook 的人很有用，而且对你将来回头分析代码也很有用。

说点题外话：最近，文字表达化编程这个概念已经发展成为一门完整的编程语言，即 [**Eve**](http://www.witheve.com/)。

**notebook 如何工作**

Jupyter notebook 源自 Fernando Perez 发起的 [**IPython 项目**](https://ipython.org/)。IPython 是一种交互式 shell，与普通的 Python shell 相似，但具有一些很好的功能（例如语法高亮显示和代码补全）。最初，notebook 的工作方式是，将来自 Web 应用（你在浏览器中看到的 notebook）的消息发送给 IPython 内核（在后台运行的 IPython 应用程序）。内核执行代码，然后将结果发送回 notebook。当前架构与之相似，具体见下图。



摘自 [**Jupyter 文档**](https://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html)

核心是 notebook 的服务器。你通过浏览器连接到该服务器，而 notebook 呈现为 Web 应用。你在 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。之后，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**](https://www.r-project.org/) 语言和 [**Julia**](http://julialang.org/) 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 **Ju**lia、**Pyt**hon 和 **R** 组合而成。如果有兴趣，不妨看看**[可用内核的列表](https://github.com/ipython/ipython/wiki/IPython-kernels-for-other-languages" \t "_blank)**。

另一个优点是，你可以在任何地方运行 notebook 服务器，并且可通过互联网访问服务器。通常，你会在存储所有数据和 notebook 文件的自有计算机上运行服务器。但是，你也可以在远程计算机或云实例（如 Amazon 的 EC2）上**[设置服务器](http://jupyter-notebook.readthedocs.io/en/latest/public_server.html" \t "_blank)**。之后，你就可以在世界上任何地方通过浏览器访问 notebook。

# 安装 Jupyter notebook

目前，安装 Jupyter 的最简单方法是使用 Anaconda。该发行版附带了 Jupyter notebook。你能够在默认环境下使用 notebook。

要在 conda 环境中安装 Jupyter notebook，请使用 conda install jupyter notebook。

也可以通过 pip 使用 pip install jupyter notebook 来获得 Jupyter notebook。

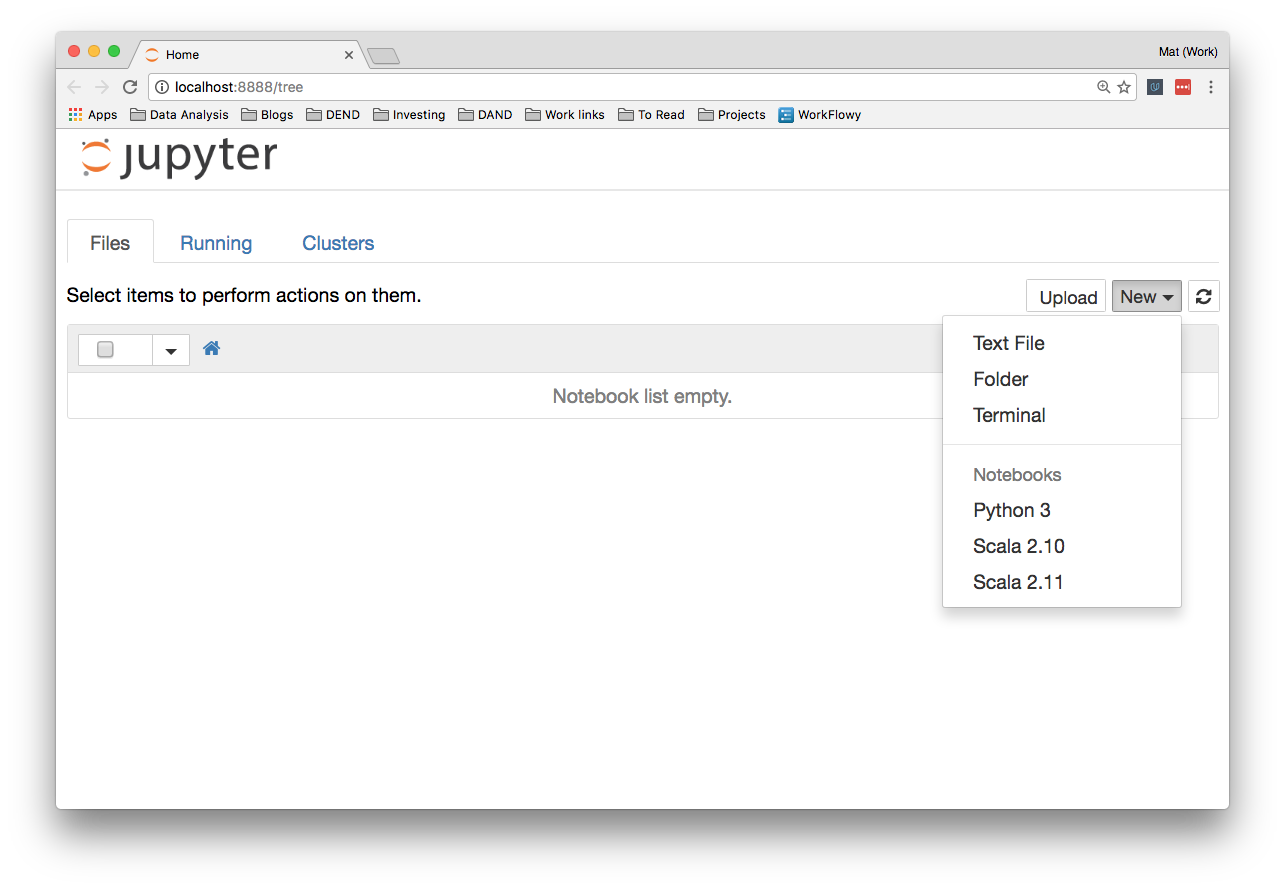
# 启动 notebook 服务器

要启动 notebook 服务器，请在终端或控制台中输入 jupyter notebook。服务器会在你运行此命令的目录中启动。这意味着任何 notebook 文件都会保存在该目录下。你通常希望在 notebook 文件所在的目录中启动服务器，不过你也可以在文件系统中导航到 notebook 文件所在的位置。

运行此命令时（请自己试一下！），服务器主页会在浏览器中打开。默认情况下，notebook 服务器的运行地址是 http://localhost:8888。该地址的含义是：localhost 表示你的计算机，而 8888 是服务器的通信端口。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**](http://localhost:8888/) 返回到 web 页面中。

如果同时启动了另一个 notebook 服务器，新服务器会尝试使用端口 8888，但由于此端口已被占用，因此新服务器会在端口 8889 上运行。之后，你可以通过 http://localhost:8889 连接到新服务器。每个额外的 notebook 服务器都会像这样增大端口号。

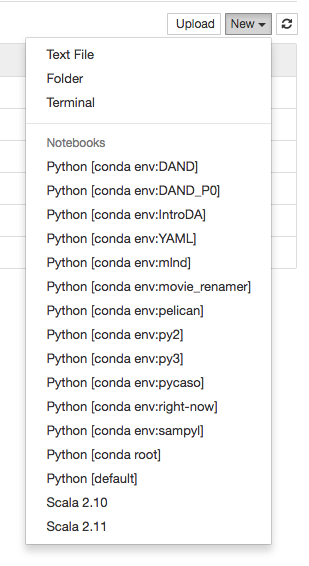
如果你尝试启动自己的服务器，它应类似以下所示：



你可能会看到上面列表中的一些文件和文件夹，具体取决于你在哪里启动服务器。

在右侧，你可以点击“New”（新建），创建新的 notebook、文本文件、文件夹或终端。“Notebooks”下的列表显示了你已安装的内核。由于我在 Python 3 环境中运行服务器，因此列出了 Python 3 内核。你在这里看到的可能是 Python 2。我还安装了用于 Scala 2.10 和 2.11 的内核，因此它们出现在列表中。

如果在 conda 环境中运行 Jupyter notebook 服务器，则你还能选择环境中任何其他的内核（见下图）。要创建新的 notebook，请点击你要使用的内核。

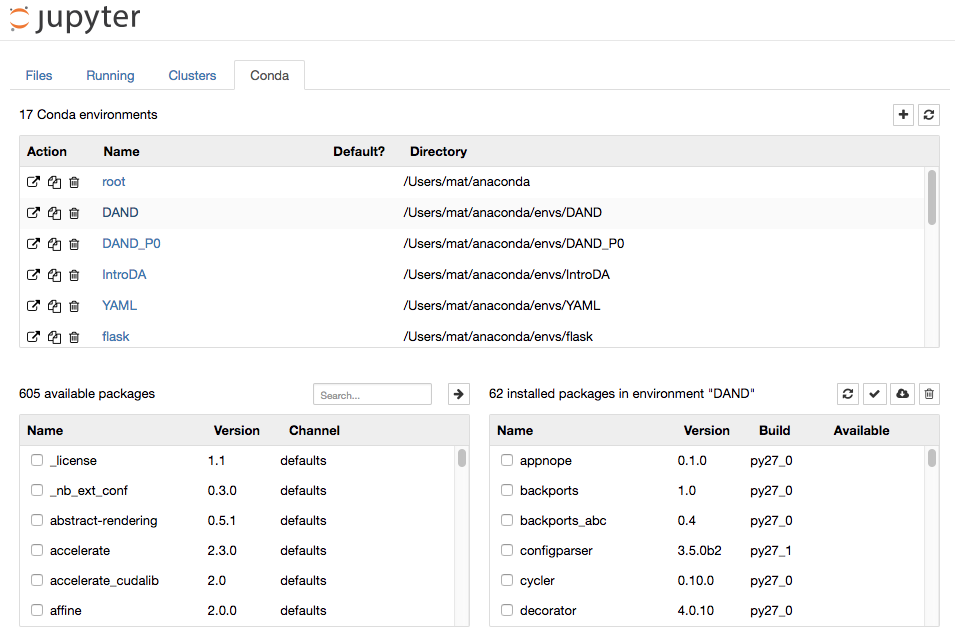


Jupyter 中的 Conda 环境

顶部的选项卡是 Files（文件）、Running（运行）和 Cluster（集群）。Files（文件）显示当前目录中的所有文件和文件夹。点击 Running（运行）选项卡会列出所有正在运行的 notebook。可以在该选项卡中管理这些 notebook。

过去，在 Clusters（集群）中创建多个用于并行计算的内核。现在，这项工作已经由 [**ipyparallel**](https://ipyparallel.readthedocs.io/en/latest/intro.html) 接管，因此该选项卡如今用处不多。

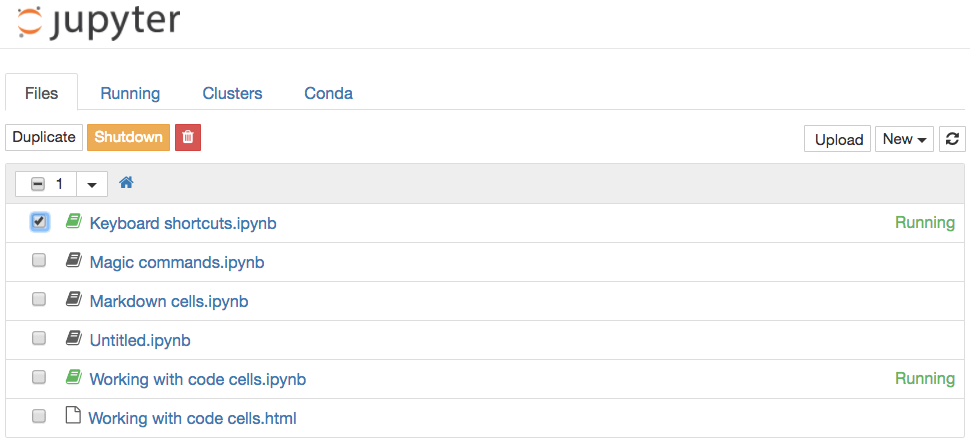
如果在 conda 环境中运行 notebook 服务器，则你还能访问以下所示的“Conda”选项卡。可以通过该选项卡管理 Jupyter 中的环境。你可以执行多种操作，例如创建新的环境、安装包、更新包、导出环境。



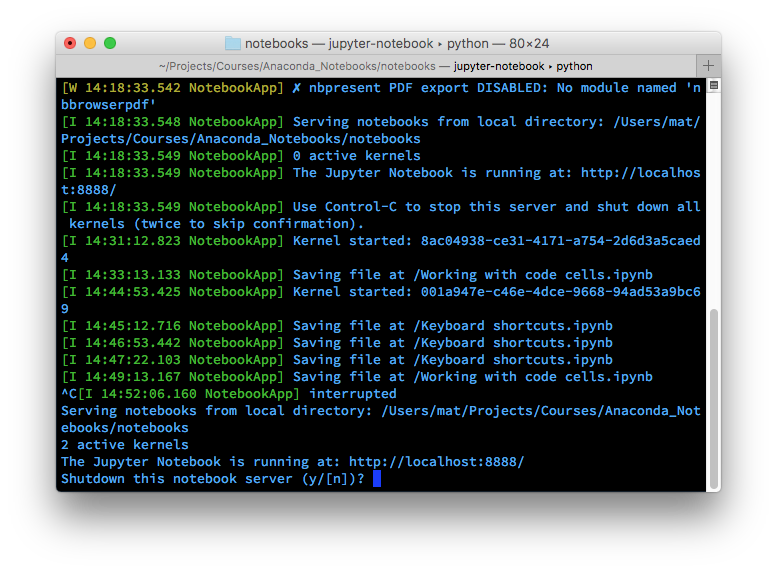
Jupyter 中的 Conda 选项卡

### 关闭 Jupyter

通过在服务器主页上选中 notebook 旁边的复选框，然后点击“Shutdown”（关闭），你就可以关闭各个 notebook。但是，在这样做之前，请确保你保存了工作！否则，在你上次保存后所做的任何更改都会丢失。下次运行 notebook 时，你还需要重新运行代码。

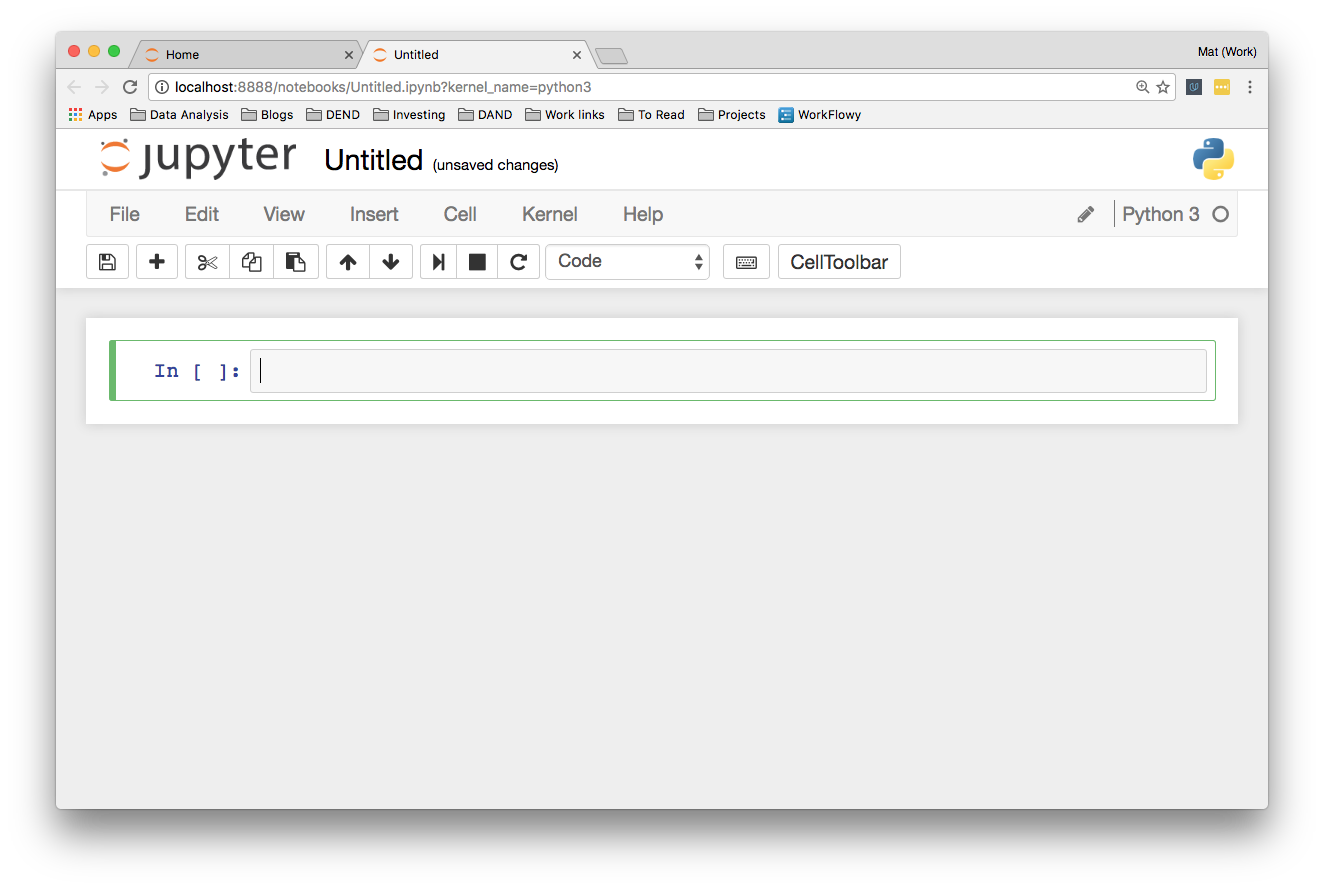


通过在终端中按两次 Ctrl + C，可以关闭整个服务器。再次提醒，这会立即关闭所有运行中的 notebook，因此，请确保你保存了工作！



# notebook 界面

创建新的 notebook 时，你会看到如下所示的界面：



请随意尝试和四处浏览一下。

你会看到外框为绿色的一个小方框。它称为单元格。单元格是你编写和运行代码的地方。你也可以更改其类型，以呈现 Markdown（一种常用于编写 Web 内容的格式化语法）。我会在后面更详细地介绍 Markdown。在工具栏中点击“Code”，将其改为 Markdown，然后改回来。小型的播放按钮用于运行单元格，而向上和向下的箭头用于上下移动单元格。

运行代码单元格时，单元格下方会显示输出。单元格还会被编号（左侧会显示 In [1]:）。这能让你知道运行的代码和运行顺序（如果运行了多个单元格的话）。在 Markdown 模式下运行单元格会将 Markdown 呈现为文本。

## 工具栏

从左侧开始，工具栏上的其他控件是：

* 软盘符号表示“保存”。请记得保存 notebook！
* + 按钮用于创建新的单元格
* 然后是用于剪切、复制和粘贴单元格的按钮。
* 运行、停止、重新启动内核
* 单元格类型：代码、Markdown、原始文本和标题
* 命令面板（见下文）
* 单元格工具栏，提供不同的单元格选项（例如将单元格用作幻灯片）

### 命令面板

小键盘符号代表命令面板。点击它会弹出一个带有搜索栏的面板，供你搜索不同的命令。这能切实帮助你加快工作速度，因此你将无需使用鼠标翻查各个菜单。你只需打开命令面板，然后键入要执行的操作。例如，如果要合并两个单元格：

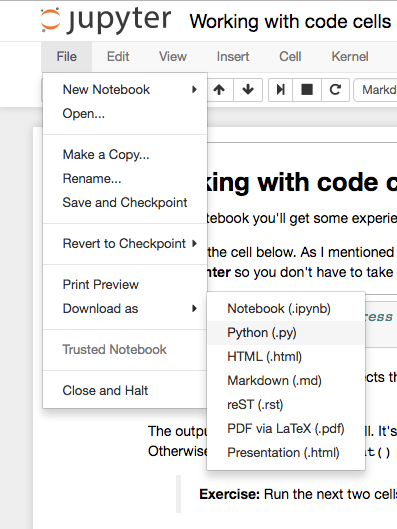
## 更多事项

顶部显示了标题。点击它可以将 notebook 重命名。

右侧是内核类型（在我的例子中是 Python 3），旁边是一个小圆形。在内核运行单元格时，会填充这个小圆形。对于大多数快速运行的操作，并不会填充它。它是一个小型指示器，在代码会运行较久时让你知道其实际是在运行中的。

工具栏包含了保存按钮，但 notebook 也会定期自动保存。标题右侧会注明最近一次的保存。你可以使用保存按钮手动进行保存，也可以按键盘上的 Esc，然后按 s。按 Esc 键会变为命令模式，而 s 是“保存”的快捷键。我会在后面介绍命令模式和快捷键。

在“File”（文件）菜单中，你可以选择多种格式进行 notebook 的下载。通常，你会希望将它作为 HTML 文件下载，以便与不使用 Jupyter 的其他人共享。也可以将 notebook 作为普通的 Python 文件下载，此时所有代码都会像平常一样运行。要在博客或文档中使用 notebook，[**Markdown**](https://daringfireball.net/projects/markdown/) 和 [**reST**](http://docutils.sourceforge.net/rst.html) 格式很合适。



**代码单元格**

notebook 中的大部分工作均在代码单元格中完成。这是编写和执行代码的地方。在代码单元格中可以执行多种操作，例如编写代码、给变量赋值、定义函数和类、导入包等。在一个单元格中执行的任何代码在所有其他单元格中均可用。

我创建了一个 notebook，你可以将它当作练习来完成。请在下面下载此 notebook [**Working With Code Cells**](https://raw.githubusercontent.com/udacity/cn-deep-learning/master/tutorials/jupyter-notebook-tutorial/working-with-code-cells.ipynb)，然后从你自己的 notebook 服务器运行它（在你的终端中，转到包含此 notebook 文件的目录，然后输入 jupyter notebook）。浏览器可能会尝试不下载就打开此 notebook 文件。如果是这样，请右击链接并选择“链接另存为...”。

# Markdown 单元格

如前所述，单元格也可用于以 Markdown 编写的文本。Markdown 是格式化语法，可让你加入链接、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。加入文本可让你直接在代码旁写出叙述性文档，以及为代码和思路编写文档。

你可以**[在此处查找文档](https://daringfireball.net/projects/markdown/basics" \t "_blank)**，但我会提供简短的入门文档。

## 标题

要编写标题，可在文本前放置井号，即 #（英文读作 pound、hash 或 [**octothorpe**](http://www.worldwidewords.org/weirdwords/ww-oct1.htm)）。一个 # 呈现为 h1 标题，两个 # 是 h2 标题，依此类推。类似以下所示：

# Header 1

## Header 2

### Header 3

呈现为

# Header 1

## Header 2

### Header 3

## 链接

要在 Markdown 中添加链接，请在文本两侧加上方括号，并在 URL 两侧加上圆括号，例如：[Udacity's home page](https://www.udacity.com) 表示指向 [**Udacity's home page**](https://www.udacity.com/) 的链接。

## 强调效果

可以使用星号或下划线（\* 或 \_）来表示粗体或斜体，从而添加强调效果。对于斜体，在文本两侧加上一个星号或下划线，例如 \_gelato\_ 或 \*gelato\* 会呈现为 gelato。

粗体文本使用两个符号，例如 \*\*aardvark\*\* 或 \_\_aardvark\_\_ 会呈现为 **aardvark**。

只要在文本两侧使用相同的符号，星号和下划线的作用都一样。

## 代码

可以通过两种不同的方式显示代码，一种是与文本内联，另一种是将代码块与文本分离。要将代码变为内联格式，请在文本两侧加上反撇号。例如，`string.punctuation` 会呈现为 string.punctuation。

要创建代码块，请另起一行并用三个反撇号将文本包起来：

```

**import** requests

response = requests.get('https://www.udacity.com')

```

或者将代码块的每一行都缩进四个空格。

**import** requests

response = requests.get('https://www.udacity.com')

## 数学表达式

在 Markdown 单元格中，可以使用 [**LaTeX**](https://www.latex-project.org/) 符号创建数学表达式。notebook 使用 MathJax 将 LaTeX 符号呈现为数学符号。要启动数学模式，请在 LaTeX 符号两侧加上美元符号（例如 $y = mx + b$），以创建内联的数学表达式。对于数学符号块，请使用两个美元符号：

$$

y = \frac{a}{b+c}

$$

此功能的确很有用，因此，如果你没有用过 LaTeX，[**请阅读这篇入门文档**](http://data-blog.udacity.com/posts/2016/10/latex-primer/)，它介绍了如何使用 LaTeX 来创建数学表达式。

## 小结

在编写 Markdown 时，可以参考这个**[速查指南](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet" \t "_blank)**。我建议使用 Markdown 单元格，与使用一堆代码块相比，这使 notebook 变得更易于阅读。

# 快捷键

notebook 自带一组快捷键，能让你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。要详细了解这些快捷键和练习它们的用法，请在下面下载 notebook [**Keyboard Shortcuts**](https://raw.githubusercontent.com/udacity/cn-deep-learning/master/tutorials/jupyter-notebook-tutorial/keyboard-shortcuts.ipynb)。再次提醒，浏览器可能会尝试打开它，但请将它保存到计算机中。请右击链接并选择“链接另存为...”。

#### **辅助材料**

[**keyboard-shortcuts\_cn.ipynb**](https://d17h27t6h515a5.cloudfront.net/topher/2017/June/5936ed78_keyboard-shortcuts-cn/keyboard-shortcuts-cn.ipynb)

**Magic 关键字**

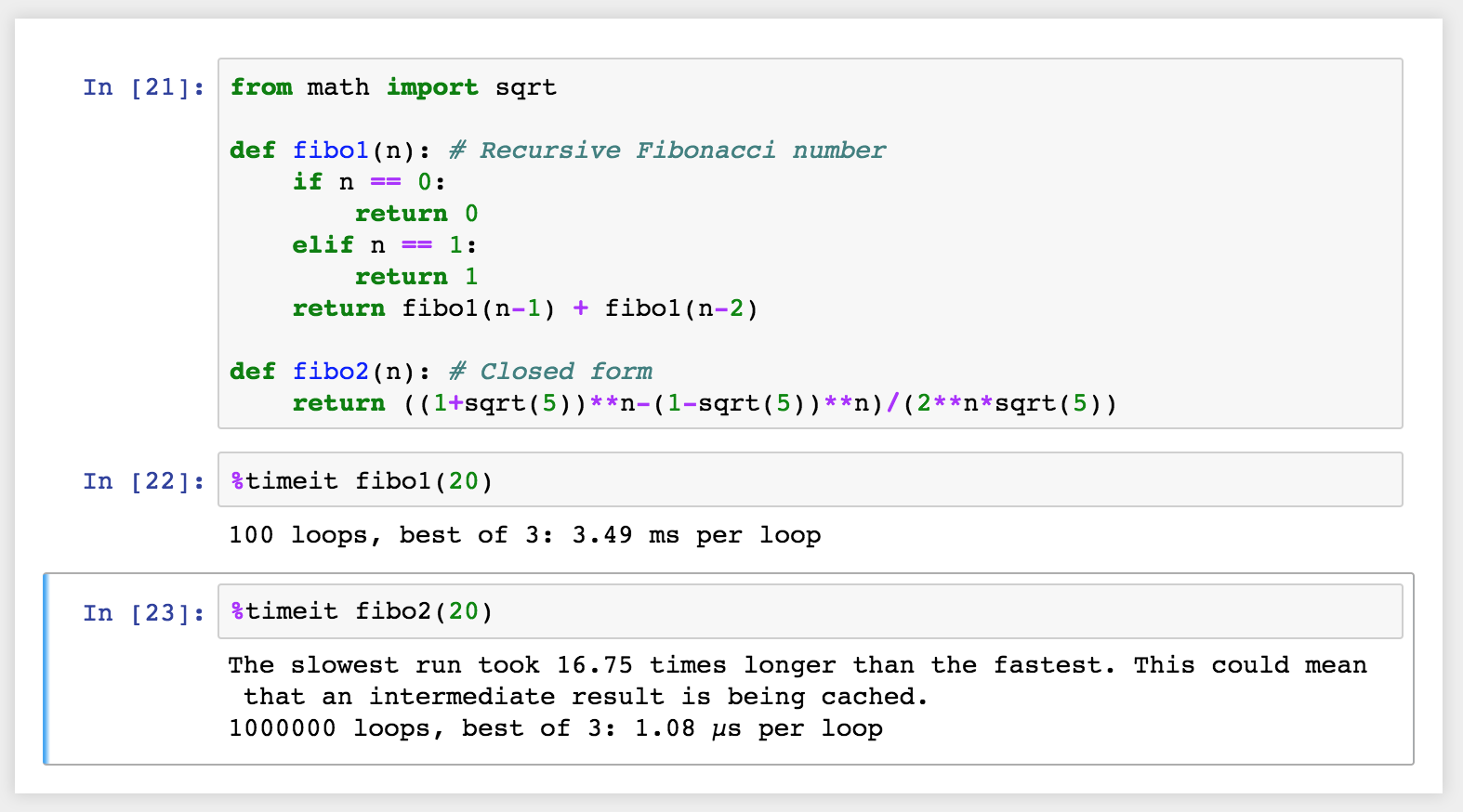
Magic 关键字是可以在单元格中运行的特殊命令，能让你控制 notebook 本身或执行系统调用（例如更改目录）。例如，在 notebook 中可以使用 %matplotlib 将 matplotlib 设置为以交互方式工作。

Magic 命令的前面带有一个或两个百分号（% 或 %%），分别对应行 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。

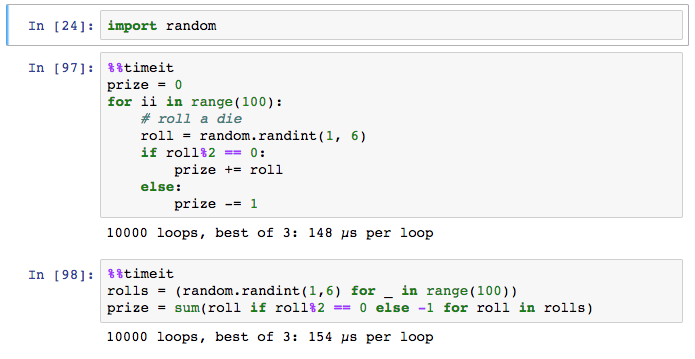
**注意：**这些 Magic 关键字是特定于普通 Python 内核的关键字。如果使用其他内核，这些关键字很有可能无效。

**代码计时**

有时候，你可能要花些精力优化代码，让代码运行得更快。在此优化过程中，必须对代码的运行速度进行计时。可以使用 Magic 命令 timeit 测算函数的运行时间，如下所示：



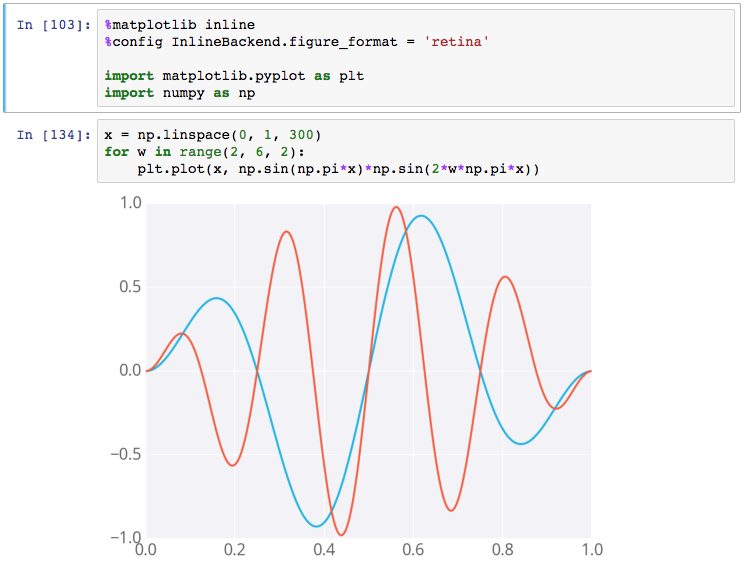
如果要测算整个单元格的运行时间，请使用 %%timeit，如下所示：



**在 notebook 中嵌入可视化内容**

如前所述，notebook 允许你将图像与文本和代码一起嵌入。这在你使用 matplotlib 或其他绘图包创建可视化内容时最为有用。在 notebook 中可以使用 %matplotlib 将 matplotlib 设置为以交互方式工作。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”](http://matplotlib.org/faq/usage_faq.html" \l "what-is-a-backend" \t "_blank)**（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。

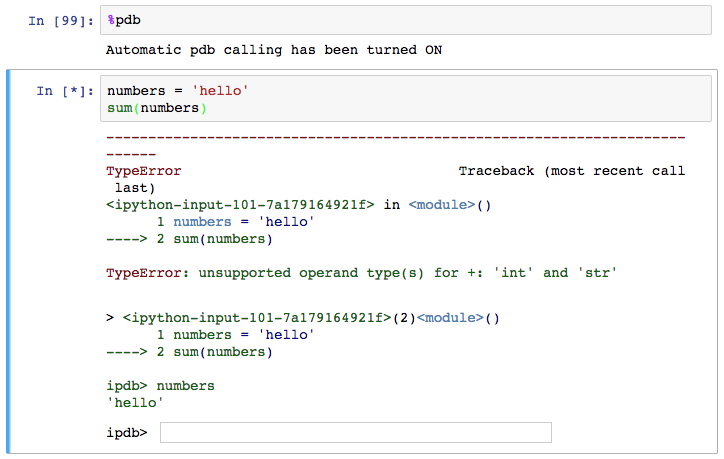
**提示：**在分辨率较高的屏幕（例如 Retina 显示屏）上，notebook 中的默认图像可能会显得模糊。可以在 %matplotlib inline 之后使用 %config InlineBackend.figure\_format = 'retina' 来呈现分辨率较高的图像。



notebook 中的图形示例

**在 notebook 中进行调试**

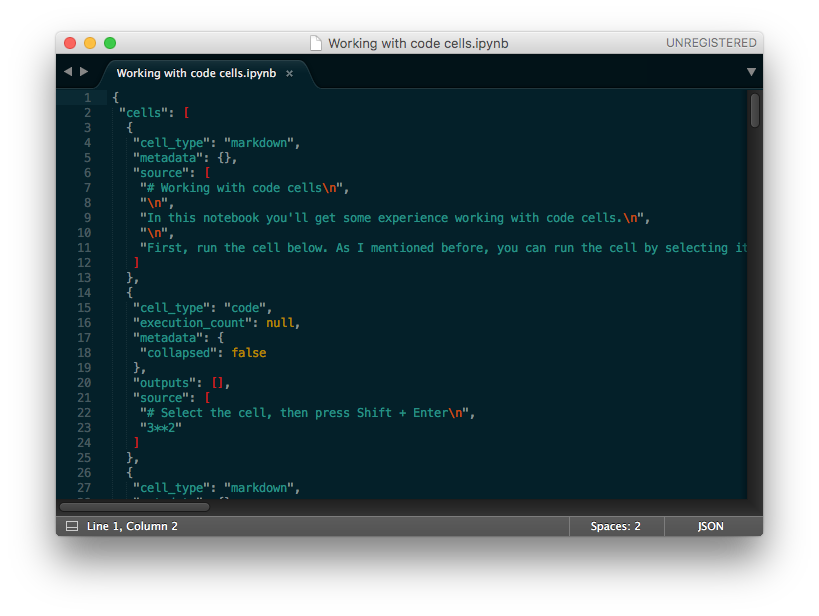
对于 Python 内核，可以使用 Magic 命令 %pdb 开启交互式调试器。出错时，你能检查当前命名空间中的变量。



在 notebook 中进行调试

# 在上图中，可以看到我尝试对字符串求和，这造成了错误。调试器指出了该错误，并提示转换 notebook

Notebook 只是扩展名为 .ipynb 的大型 [**JSON**](http://www.json.org/) 文件。



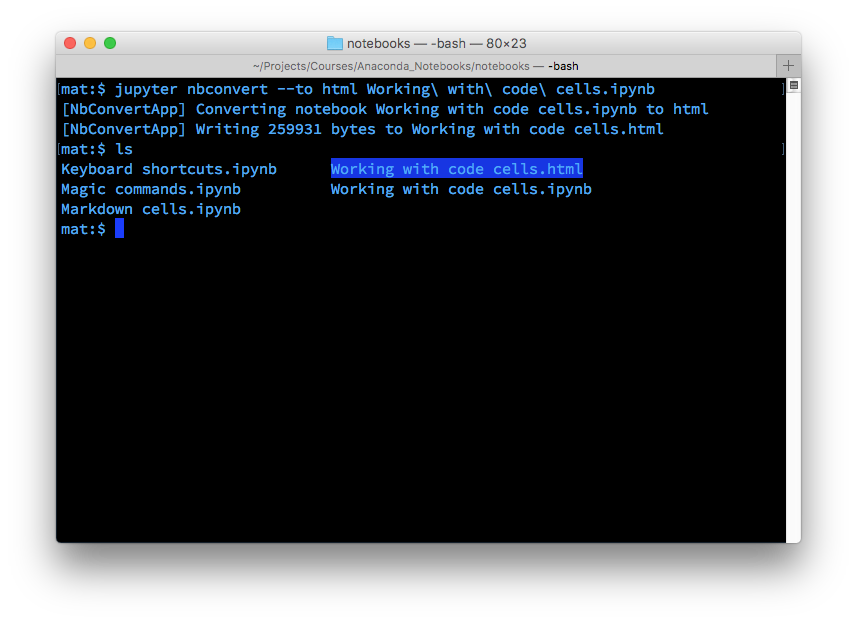
在文本编辑器中打开的 notebook 文件显示 JSON 数据

由于 notebook 是 JSON 文件，因此，可以轻松将其转换为其他格式。Jupyter 附带了一个名为 nbconvert 的实用程序，可将 notebook 转换为 HTML、Markdown、幻灯片等格式。

例如，要将 notebook 转换为 HTML 文件，请在终端中使用

jupyter nbconvert --to html notebook.ipynb

要将 notebook 与不使用 notebook 的其他人共享，转换为 HTML 很有用。而要在博客和其他接受 Markdown 格式化的文本编辑器中显示 notebook，Markdown 很合适。

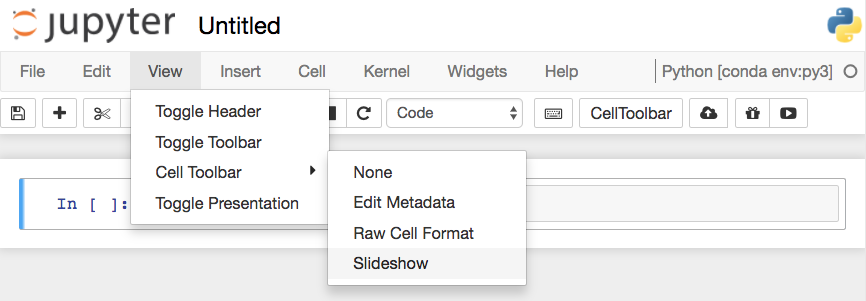


像平常一样，要详细了解 nbconvert，请阅读相关**[文档](https://nbconvert.readthedocs.io/en/latest/usage.html" \t "_blank)**。

**创建幻灯片**

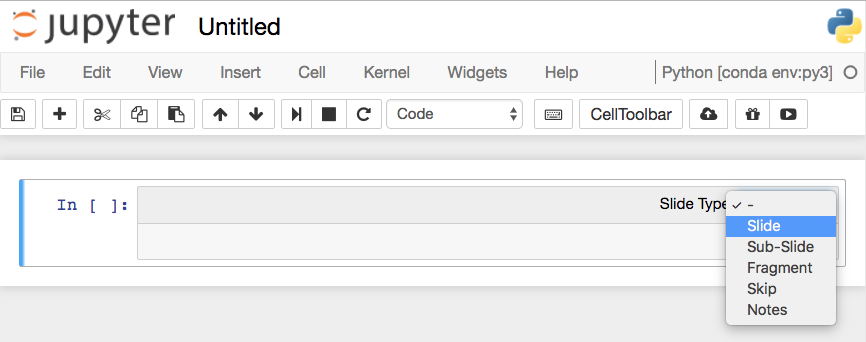
通过 notebook 创建幻灯片是我最爱的功能之一，**[你可以在浏览器中直接打开它](http://nbviewer.jupyter.org/format/slides/github/jorisvandenbossche/2015-PyDataParis/blob/master/pandas_introduction.ipynb" \l "/" \t "_blank)**，它介绍了用于处理数据的 Pandas。

在 notebook 中创建幻灯片的过程像平常一样，但需要指定作为幻灯片的单元格和单元格的幻灯片类型。在菜单栏中，点击“View”（视图）>“Cell Toolbar”（单元格工具栏）>“Slideshow”（幻灯片），以便在每个单元格上弹出幻灯片单元格菜单。



打开单元格的幻灯片工具栏

这会在每个单元格上显示一个下拉菜单，让你选择单元格在幻灯片中的显示方式。



选择幻灯片类型

**Slides**（幻灯片）是你从左向右移动的完整幻灯片。按向上或向下的箭头时，**Sub-slides**（子幻灯片）会出现在幻灯片中。**Fragments**（片段）最初是隐藏的，在你按下按钮时会出现。选择 **Skip**（忽略）会忽略幻灯片中的单元格，而选择 **Notes**（备注）会将为演讲者保留备注。

**运行幻灯片**

要通过 notebook 文件创建幻灯片，需要使用 nbconvert：

jupyter nbconvert notebook.ipynb --to slides

这只是将 notebook 转换为幻灯片必需的文件，你需要向其提供 HTTP 服务器才能真正看到演示文稿。

要转换它并立即看到它，请使用

jupyter nbconvert notebook.ipynb --to slides --post serve

要详细了解 pdb，请阅读[**此文档**](https://docs.python.org/3/library/pdb.html)。要退出调试器，在提示符中输入 q 即可。

**补充读物**

Magic 命令还有很多，我只是介绍了你将会用得最多的一些命令。要了解更多信息，请查看[**此列表**](http://ipython.readthedocs.io/en/stable/interactive/magics.html)，它列出了所有可用的 Magic 命令。