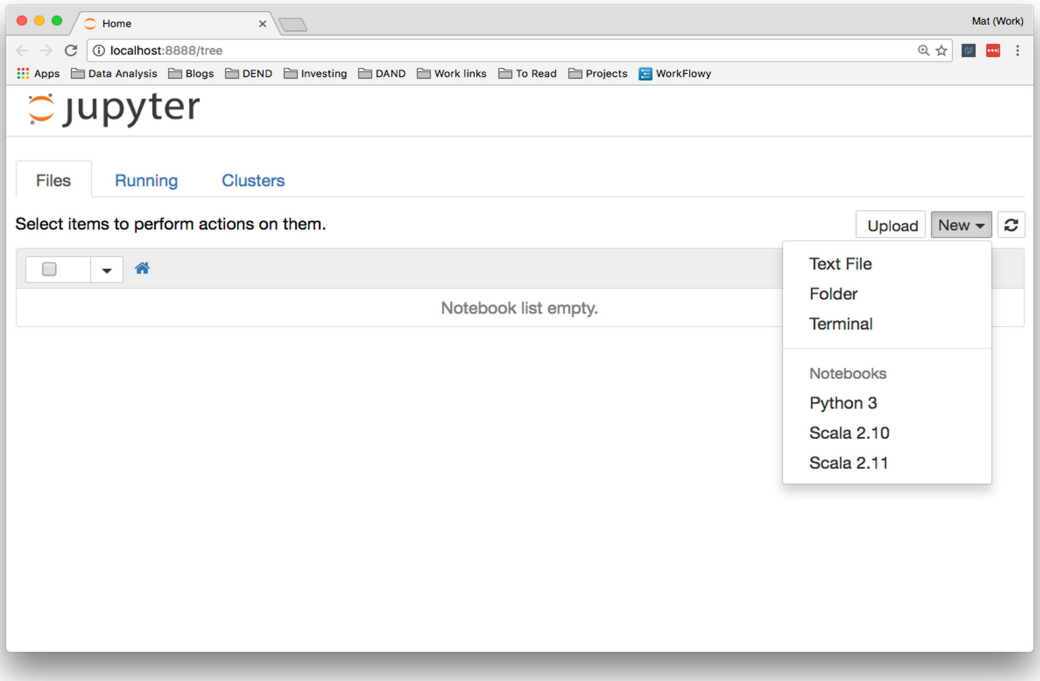
安装 Jupyter notebook

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

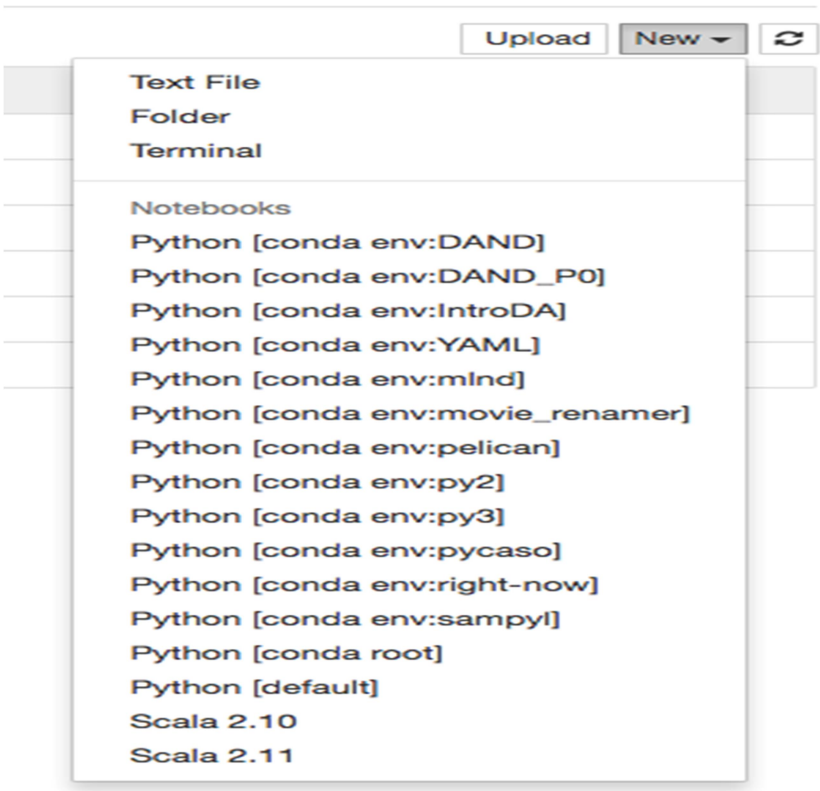
要启动 notebook 服务器，请在终端或控制台中输入 `jupyter notebook`。服务器会在你运行此命令的目录中启动。这意味着任何 notebook 文件都会保存在该目录下。你通常希望在 notebook 文件所在的目录中启动服务器，不过你也可以在文件系统中导航到 notebook 文件所在的位置。
运行此命令时（请自己试一下！），服务器主页会在浏览器中打开。默认情况下，notebook 服务器的运行地址是 `http://localhost:8888`。该地址的含义是：`localhost` 表示你的计算机，而 `8888` 是服务器的通信端口。只要 notebook 服务器仍在运行，你随时都能通过在浏览器中输入 http://localhost:8888 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。

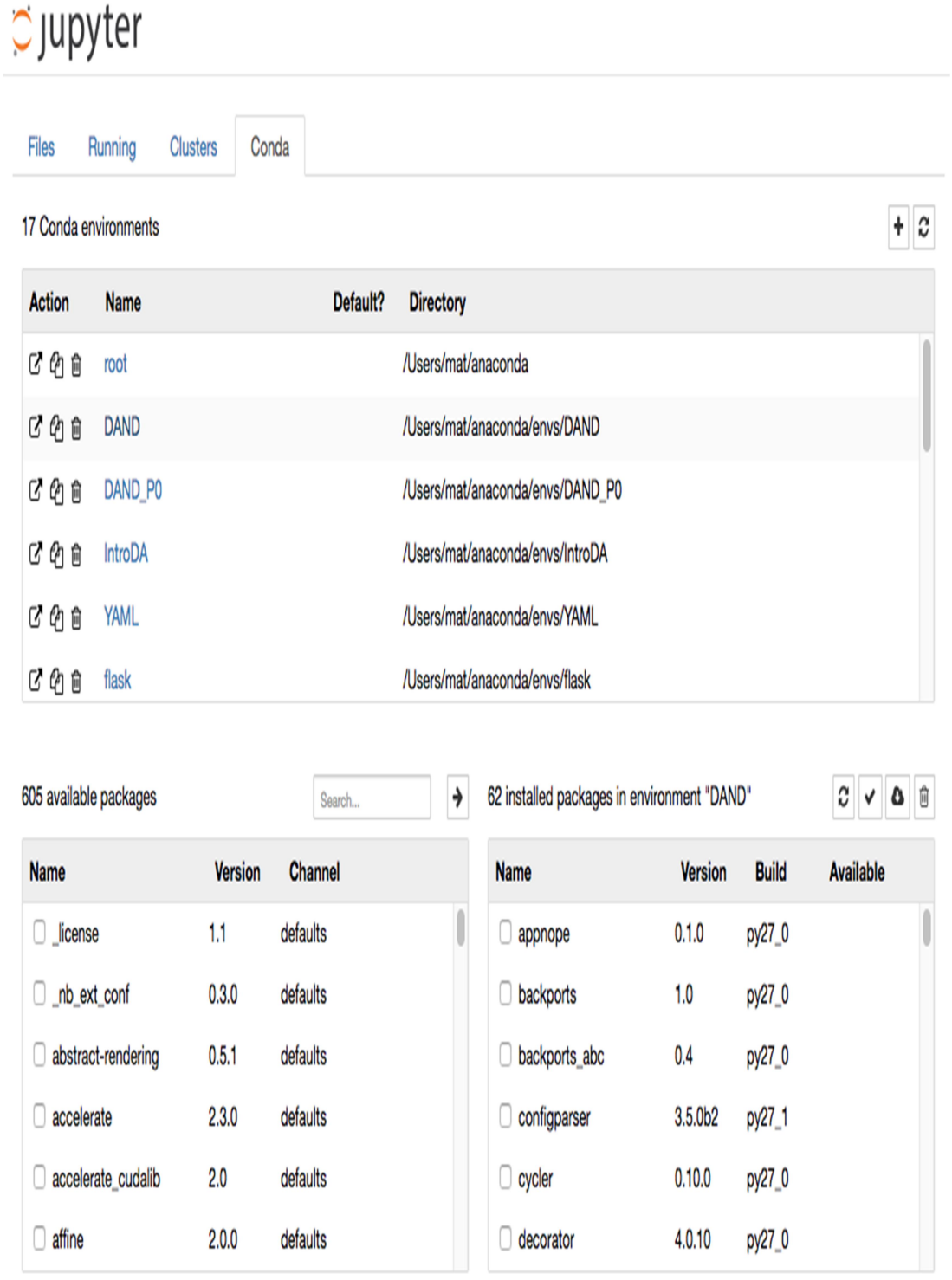
在右侧，你可以点击“New”（新建），创建新的 notebook、文本文件、文件夹或终端。“Notebooks”下的列表显示了你已安装的内核。由于我在 Python 3 环境中运行服务器，因此列出了 Python 3 内核。你在这里看到的可能是 Python 2。我还安装了用于 Scala 2.10 和 2.11 的内核，因此它们出现在列表中。

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** 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。

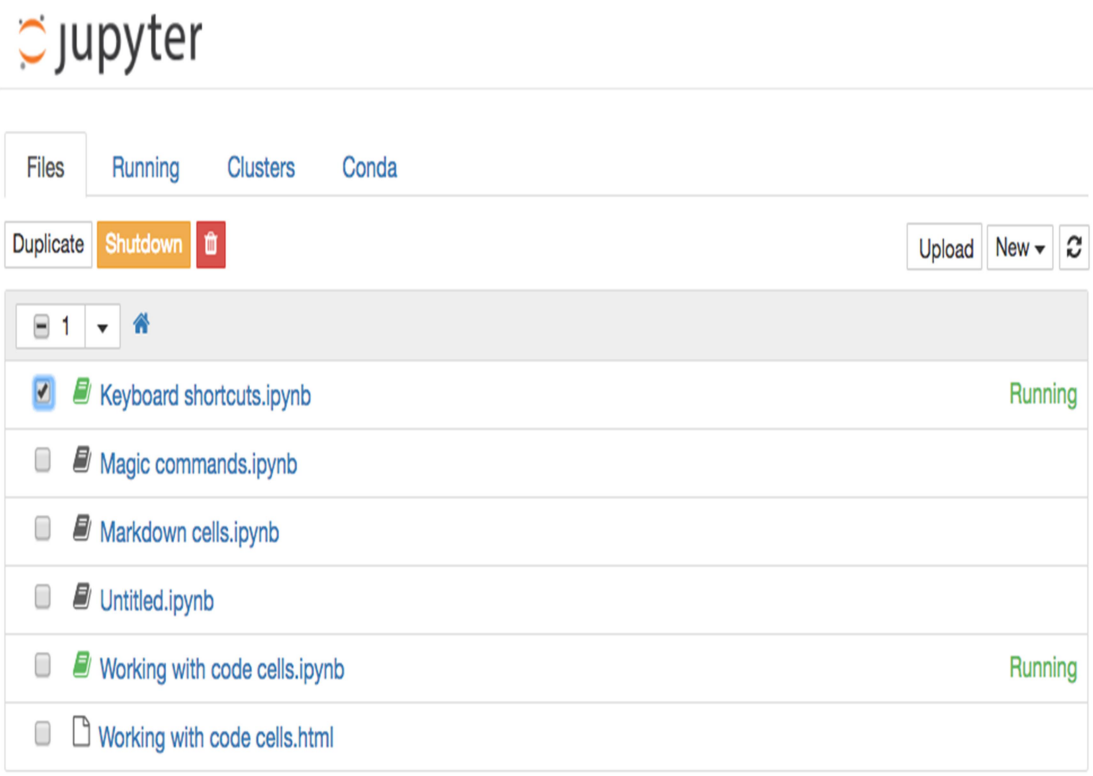


Jupyter 中的 Conda 选项卡

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

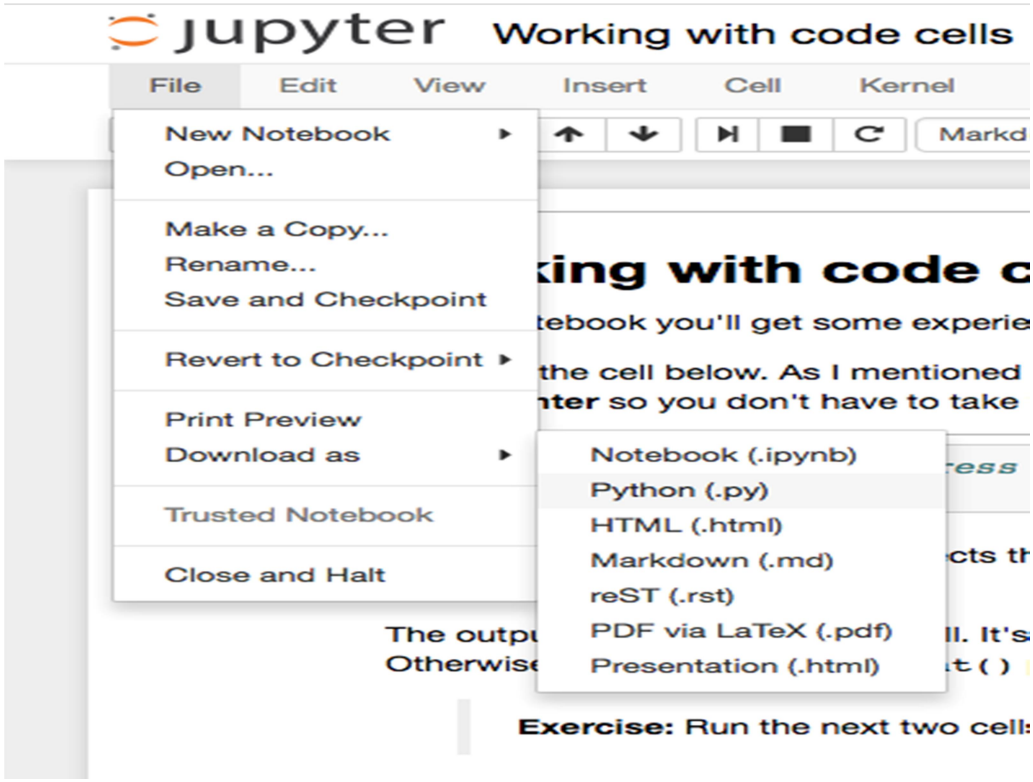
小键盘符号代表命令面板。点击它会弹出一个带有搜索栏的面板，供你搜索不同的命令。这能切实帮助你加快工作速度，因此你将无需使用鼠标翻查各个菜单。你只需打开命令面板，然后键入要执行的操作。例如，如果要合并两个单元格：

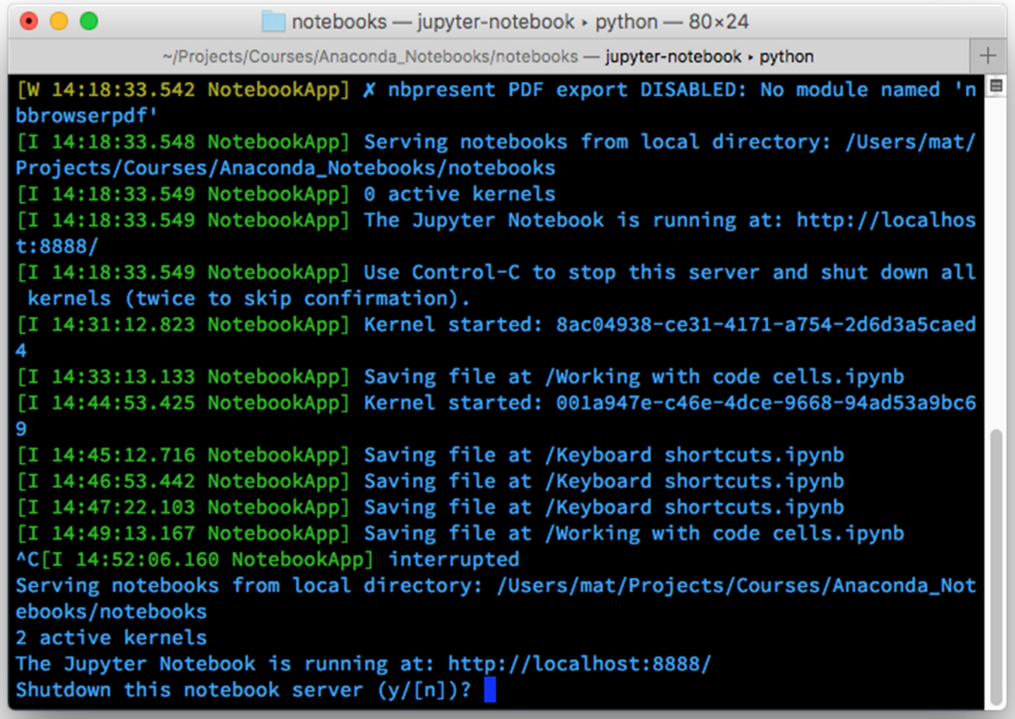
更多事项

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** 和 **reST** 格式很合适。


jupyter
Working with code cells
File
Edit
View
Insert
Cell
Kernel
New Notebook
Open...
Make a Copy...
Rename...
Save and Checkpoint
Revert to Checkpoint
Print Preview
Download as
Trusted Notebook
Close and Halt
Notebook (.ipynb)
Python (.py)
HTML (.html)
Markdown (.md)
reST (.rst)
PDF via LaTeX (.pdf)
Presentation (.html)
Exercise: Run the next two cells


notebooks — jupyter-notebook ▸ python — 80×24
~/Projects/Courses/Anaconda_Notebooks/notebooks — jupyter-notebook ▸ python
[W 14:18:33.542 NotebookApp] ✗ nbpresent PDF export DISABLED: No module named 'nbbrowserpdf'
[I 14:18:33.548 NotebookApp] Serving notebooks from local directory: /Users/mat/Projects/Courses/Anaconda_Notebooks/notebooks
[I 14:18:33.549 NotebookApp] 0 active kernels
[I 14:18:33.549 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/
[I 14:18:33.549 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[I 14:31:12.823 NotebookApp] Kernel started: 8ac04938-ce31-4171-a754-2d6d3a5caed4
[I 14:33:13.133 NotebookApp] Saving file at /Working with code cells.ipynb
[I 14:44:53.425 NotebookApp] Kernel started: 001a947e-c46e-4dce-9668-94ad53a9bc69
[I 14:45:12.716 NotebookApp] Saving file at /Keyboard shortcuts.ipynb
[I 14:46:53.442 NotebookApp] Saving file at /Keyboard shortcuts.ipynb
[I 14:47:22.103 NotebookApp] Saving file at /Keyboard shortcuts.ipynb
[I 14:49:13.167 NotebookApp] Saving file at /Working with code cells.ipynb
^C[I 14:52:06.160 NotebookApp] interrupted
Serving notebooks from local directory: /Users/mat/Projects/Courses/Anaconda_Notebooks/notebooks
2 active kernels
The Jupyter Notebook is running at: http://localhost:8888/
Shutdown this notebook server (y/[n])?

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**，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。

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）。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（一般在键盘数字 1 左边）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](#) 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。

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：

```
In [21]: from math import sqrt

         def fibo1(n): # Recursive Fibonacci number
             if n == 0:
                 return 0
             elif n == 1:
                 return 1
             return fibo1(n-1) + fibo1(n-2)

         def fibo2(n): # Closed form
             return ((1+sqrt(5))**n-(1-sqrt(5))**n)/(2**n*sqrt(5))

In [22]: %timeit fibo1(20)

         100 loops, best of 3: 3.49 ms per loop

In [23]: %timeit fibo2(20)

         The slowest run took 16.75 times longer than the fastest. This could mean
          that an intermediate result is being cached.
         1000000 loops, best of 3: 1.08 µs per loop
```

如果要测算整个单元格的运行时间，请使用 `%%timeit`，如下所示：

```
In [24]: import random

In [97]: %%timeit
         prize = 0
         for ii in range(100):
             # roll a die
             roll = random.randint(1, 6)
             if roll%2 == 0:
                 prize += roll
             else:
                 prize -= 1

         10000 loops, best of 3: 148 µs per loop

In [98]: %%timeit
         rolls = (random.randint(1,6) for _ in range(100))
         prize = sum(roll if roll%2 == 0 else -1 for roll in rolls)

         10000 loops, best of 3: 154 µs per loop
```

在 notebook 中嵌入可视化内容

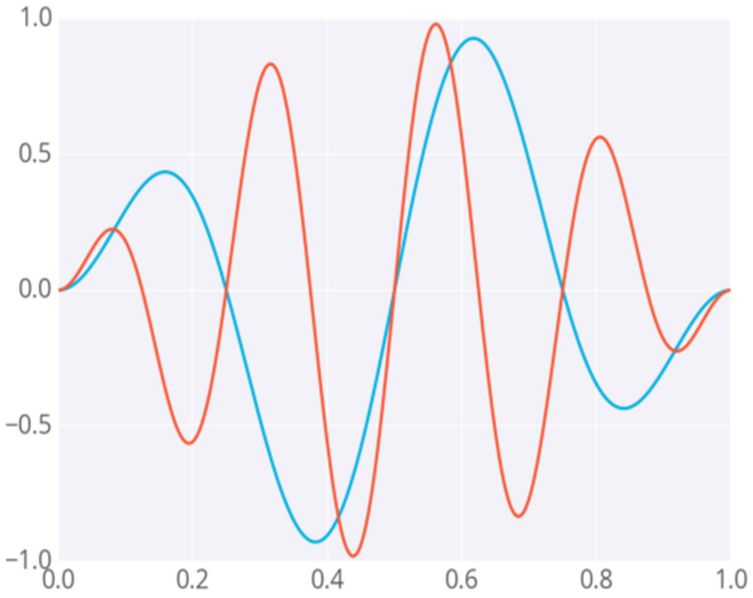
如前所述，notebook 允许你将图像与文本和代码一起嵌入。这在你使用 `matplotlib` 或其他绘图包创建可视化内容时最为有用。在 notebook 中可以使用 `%matplotlib` 将 `matplotlib` 设置为以交互方式工作。默认情况下，图形呈现在各自的窗口中。但是，你可以通过命令传递参数，以选择特定的“后端”（呈现图像的软件）。要直接在 notebook 中呈现图形，应将通过命令 `%matplotlib inline` 内联后端一起使用。

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。

```
In [103]: %matplotlib inline
          %config InlineBackend.figure_format = 'retina'

          import matplotlib.pyplot as plt
          import numpy as np
```

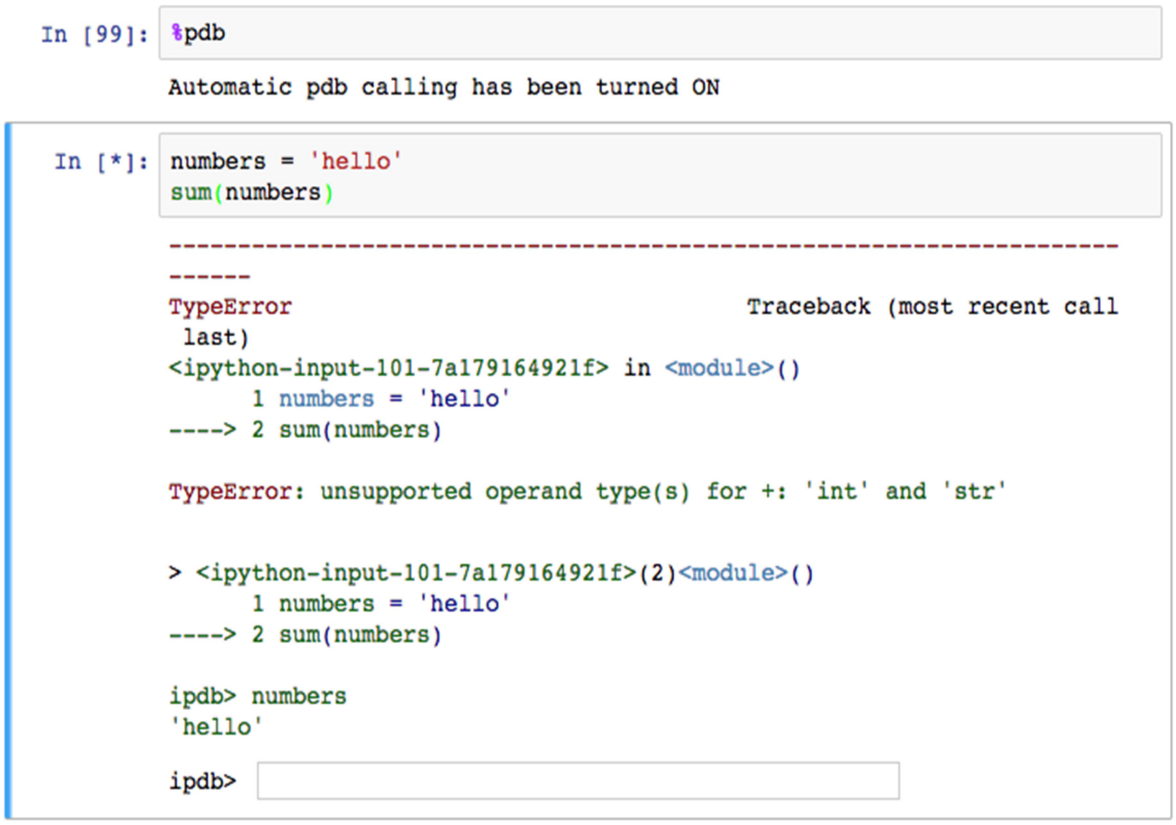
```
In [134]: x = np.linspace(0, 1, 300)
          for w in range(2, 6, 2):
              plt.plot(x, np.sin(np.pi*x)*np.sin(2*w*np.pi*x))
```



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。



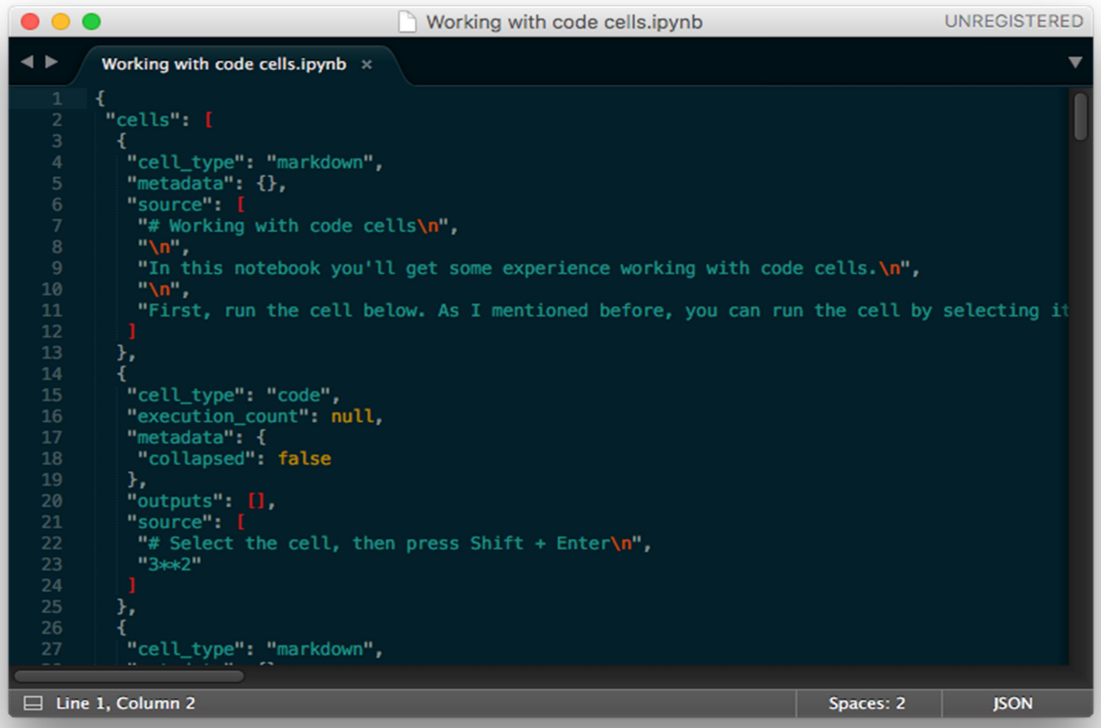
在 notebook 中进行调试

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，并提示你检查代码。

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。

转换 notebook

Notebook 只是扩展名为 `.ipynb` 的大型 **JSON** 文件。



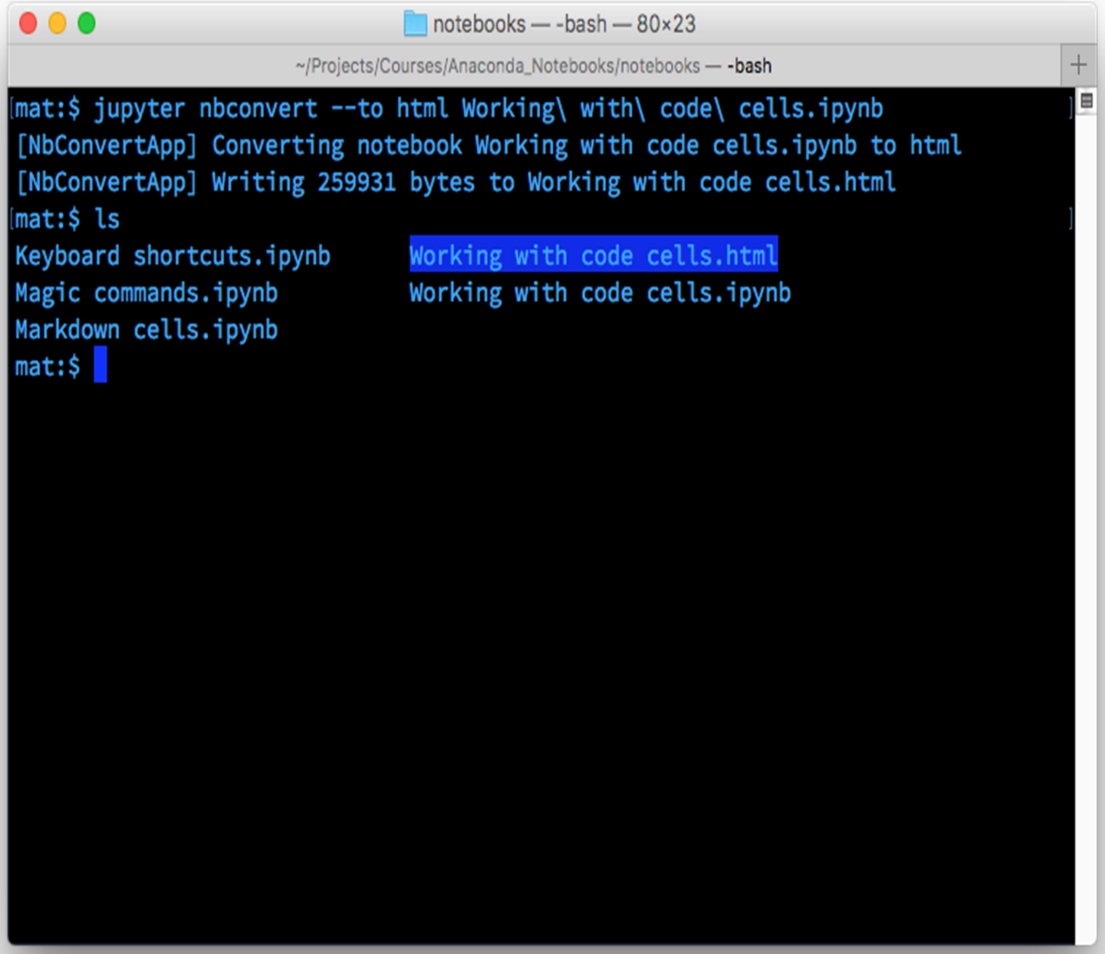
在文本编辑器中打开的 notebook 文件显示 JSON 数据

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。


notebooks — -bash — 80×23
~/Projects/Courses/Anaconda_Notebooks/notebooks — -bash
mat:$ jupyter nbconvert --to html Working\ with\ code\ cells.ipynb
[NbConvertApp] Converting notebook Working with code cells.ipynb to html
[NbConvertApp] Writing 259931 bytes to Working with code cells.html
mat:$ ls
Keyboard shortcuts.ipynb
Magic commands.ipynb
Markdown cells.ipynb
Working with code cells.html
Working with code cells.ipynb
mat:$