

# IPython 学习

---

## IPython介绍

`ipython` 是一个 `python` 的交互式 `shell`，比默认的 `python shell` 好用得多，支持变量自动补全，自动缩进，支持 `bash shell` 命令，内置了许多很有用的功能和函数。学习 `ipython` 将会让我们以一种更高的效率来使用 `python`。同时它也是利用 `Python` 进行科学计算和交互可视化的一个最佳的平台。

## 安装

安装 `ipython` 很简单，可以直接使用 `pip` 管理工具即可：

```
pip install ipython
```

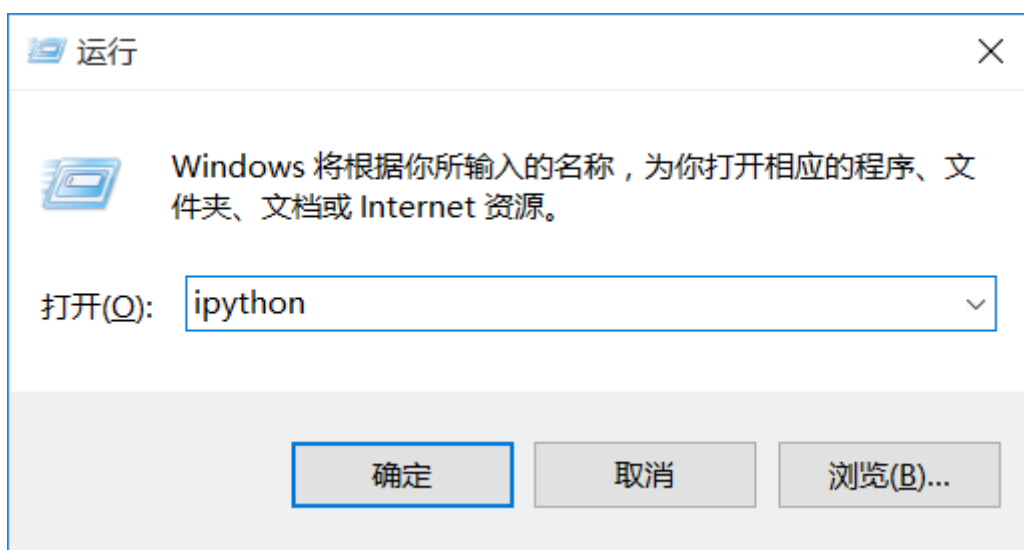
这条命令会自动安装 `IPython` 以及它的各种依赖包

如果我们也想在 `notebook` 中或者在 `Qt console` 中使用 `IPython`，我们还需要安装 `Jupyter`，如下命令：

```
pip install jupyter
```

## 运行

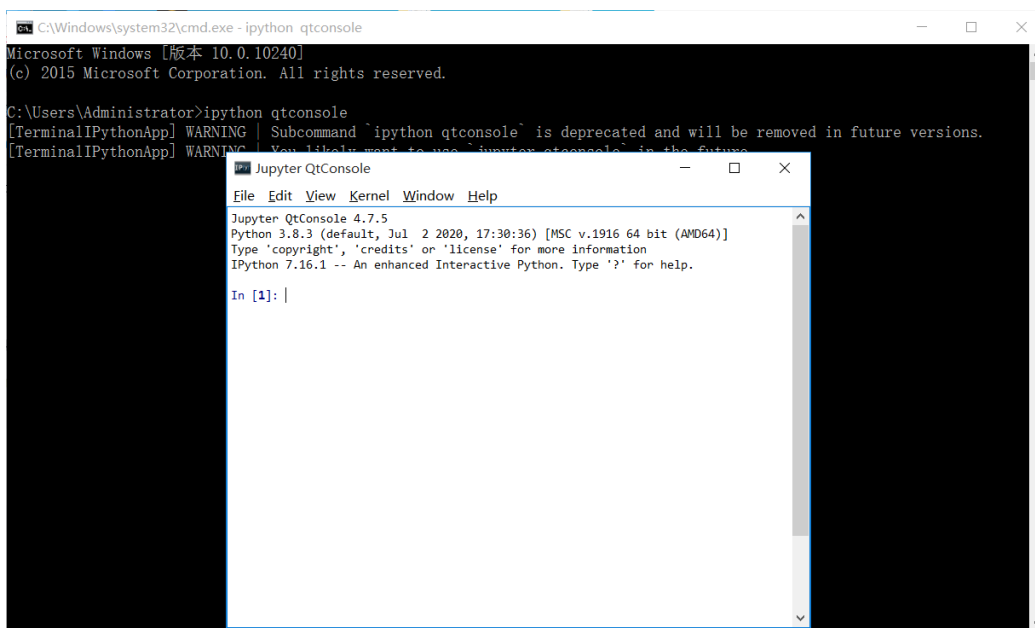
在 `cmd` 中输入 `ipython` 即可启动 `IPython`



```
IPython: C:\Users\Administrator
Type 'copyright', 'credits' or 'license' for more information
IPython 7.16.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

或者输入 `ipython qtconsole` 进入ipython图形交互界面：



输入`exit`命令或者"`Ctrl+D`" 快捷键可推出IPython。

## Tab键自动补全

在`shell`中输入表达式时，只要按下`Tab`键，当前命名空间中任何与输入的字符串相匹配的变量(对象或者函数等)就会被找出来：

```
IPython: C:\Users\Administrator
Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.16.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: x=[1,2,3]
In [2]: x.clear
and          NTUSER.DAT          %%bash          %hist          %notebook
if           NTUSER.D...M.blf    %bookmark      %history       ntuser.dat.LOG1
in           NTUSER.D...1.regtrans-ms %capture       %html          ntuser.dat.LOG2
is           NTUSER.D...2.regtrans-ms %cd            %javascript    ntuser.ini
not          NetHood           %cls           %js            %page
or           OneDrive          %cmd           %killbgscripts %paste
%%!          Pictures          %colors        %latex         %pastebin
"Application Data" PrintHood         %conda         %l1dir         %pdb
"Local Settings"  PycharmProjects %config        %load          %pdef
"My Documents"    Recent           %copy          %load_ext      %pdoc
"Saved Games"     Roaming          %cpaste        %loadpy        %perl
"VirtualBox VMs"  %%SVG            %ddir          %logoff        %pfile
AppData          Searches         %debug         %logon         %pinfo
Contacts          SendTo           %debug         %logstart      %pinfo2
Cookies          Templates        %dhist         %logstate      %nin
```

## 内省

在一个变量的前面或者后面加上一个问好?，就可以将该对象的一些通用信息显示出来。

```
IPython: C:\Users\Administrator

Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.16.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: x=[1,2,3]

In [2]: x?
Type:      list
String form: [1, 2, 3]
Length:    3
Docstring:
Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list.
The argument must be an iterable if specified.

In [3]: _
```

如果对象是一个函数或者实例方法，则它的docstring也会被显示出来

```
#定义一个函数
In [3]: def add(a,b):
...:
...:
...:     """
...:     Add Two numbers
...:     """
...:     return a+b
...:

#使用一个问号，可以查看该方法的内省信息：
In [4]: add?
Signature: add(a, b)
Docstring: Add Two numbers
File:      c:\users\administrator\<ipython-input-3-dca3c5508364>
Type:      function

#使用两个问号，就可以显示出该方法的源码
In [5]: add??
Signature: add(a, b)
Source:
def add(a,b):

    """
    Add Two numbers
    """
    return a+b
```

```
File:      c:\users\administrator\<ipython-input-3-dca3c5508364>
Type:      function
```

另外，我们可以使用通配符字符串查找出所有与该通配符字符串相匹配的名称，比如我们查找 `re` 模块下所有的包含 `find` 的函数：

```
In [6]: import re

In [7]: re.*find*?
re.findall
re.finditer
```

## 历史命令

在 IPython shell 中，使用历史的命令可以简单地使用上下翻页键即可，另外也可以使用 `hist` 命令查看所有历史输入。

```
In [7]: re.*find*?
re.findall
re.finditer

In [8]: hist
x=[1,2,3]
x?
def add(a,b):

    """
    Add Two  numbers
    """
    return a+b
add?
add??
import re
re.*find*?
hist
```

如果在 `hist` 命令之后加上 `-n`，即 `hist -n` 也可以显示出输入的序号：

```
In [9]: hist -n
1: x=[1,2,3]
2: x?
3:
def add(a,b):

    """
    Add Two  numbers
    """
```

```
    return a+b
4: add?
5: add??
6: import re
7: re.*find*?
8: hist
9: hist -n
```

在任何的交互会话中，我们的输入历史和输出历史都会被保存在 `In` 和 `Out` 变量中，并被序号进行索引。

另外，`_`，`__`，`___` 和 `_i`，`_ii`，`_iii` 变量保存着最后三个输出和输入对象。`_n` 和 `_in` (这里的 `n` 表示具体的数字) 变量返回第 `n` 个输出和输入的历史命令。比如：

```
In [10]: _i
Out[10]: 'hist -n'

In [12]: _ii
Out[12]: '_i'

In [13]: _iii
Out[13]: '_i'
```

## 使用 `%run` 命令运行脚本

所有的文件都可以通过 `%run` 命令当做 `Python` 程序来运行，输入 `%run` 路径+`python` 文件名称即可。

```
In [14]: %run D:\Pycharm\python练习\A.py
hello,world
```

## `%timeit` 命令快速测量代码运行时间

在一个交互式会话中，我们可以使用 `%timeit` 魔法命令快速测量代码运行时间。相同的命令会在一个循环中多次执行，多次运行时长的平均值作为该命令的最终评估时长。`-n` 选项可以控制命令在单词循环中执行的次数，`-r` 选项控制执行循环的次数。

```
In [15]: %timeit [x*x for x in range(100000)]
7.15 ms ± 107 µs per loop (mean ± std. dev. of 7 runs, 100 loops
each)
Compiler time: 0.34 s
```

## 使用`%debug`命令进行快速debug

`ipython`带有一个强大的调试器。无论何时控制台抛出了一个异常，我们都可以使用`%debug`魔法命令在异常点启动调试器。接着你就能调试模式下访问所有的本地变量和整个栈回溯。使用`u`和`d`向上和向下访问栈，使用`q`退出调试器。在调试器中输入`?`可以查看所有的可用命令列表。

我们也可以使用`%pdb`魔法命令来激活IPython调试器，这样，每当异常抛出时，调试器就会自动运行。

## 使用Pylab进行交互式计算

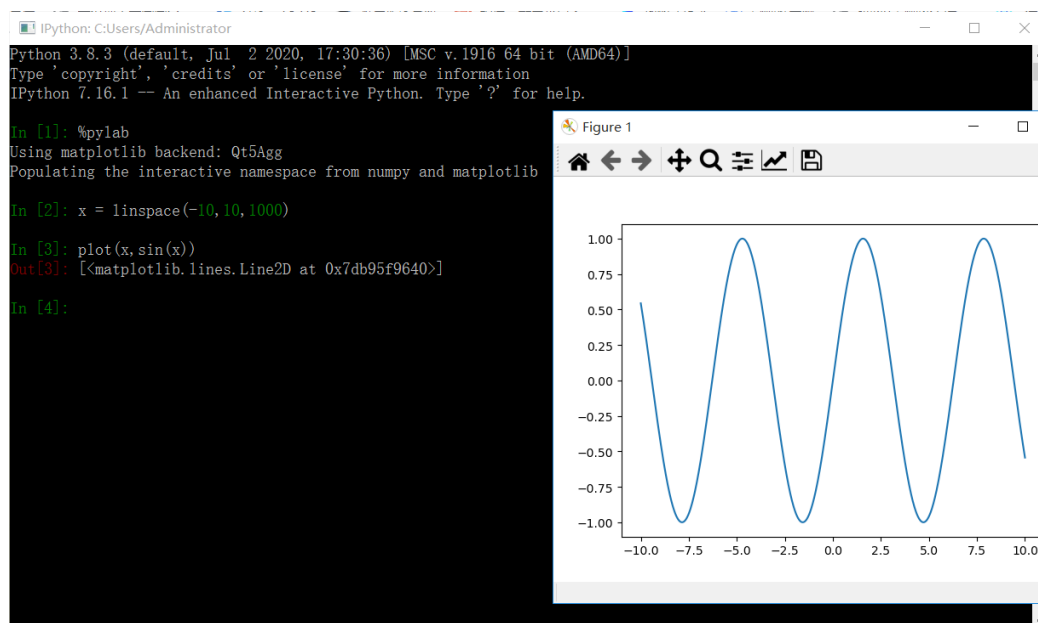
`%pylab`魔法命令可以使`Numpy`和`matplotlib`中的科学计算功能生效，这些功能被称为基于向量和矩阵的高效操作，交互可视化特性。它能够让我们在控制台进行交互式计算和动态绘图。

```
In [1]: %pylab
Using matplotlib backend: Qt5Agg
Populating the interactive namespace from numpy and matplotlib

In [2]: x = linspace(-10,10,1000)

In [3]: plot(x,sin(x))
Out[3]: [<matplotlib.lines.Line2D at 0x7db95f9640>]
```

在该示例中，我们首先定义了一个-10到10的线性空间中的1000个数值的向量，接着我们绘制了 $(x, \sin(x))$ 图像，这样我们就成功绘制出了`sin(x)`的函数图像：



### 在IPython中使用系统shell

我们可以在IPython中直接使用系统shell，并获取读取结果作为一个Python字符串列表。为了实现这种功能，我们需要使用感叹号`!`作为shell命令的前缀。比如现在在我的windows系统中，直接在IPython中ping百度

```
In [4]: !ping baidu.com
```

正在 Ping baidu.com [220.181.38.148] 具有 32 字节的数据:

来自 220.181.38.148 的回复: 字节=32 时间=34ms TTL=46

来自 220.181.38.148 的回复: 字节=32 时间=47ms TTL=46

来自 220.181.38.148 的回复: 字节=32 时间=28ms TTL=46

来自 220.181.38.148 的回复: 字节=32 时间=26ms TTL=46

220.181.38.148 的 Ping 统计信息:

数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),

往返行程的估计时间(以毫秒为单位):

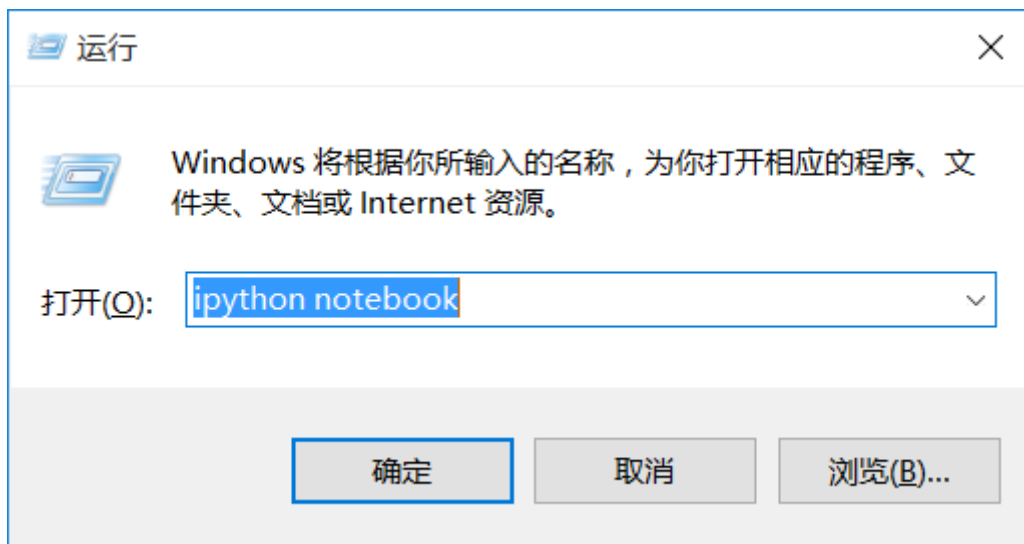
最短 = 26ms, 最长 = 47ms, 平均 = 33ms

## IPython Notebook

使用浏览器作为界面，向后台的IPython服务器发送请求。最终要的特点：可重复性的互动计算。这意味着我们可以重复更改并且执行曾经的输入记录。它可以保存成其他很多格式，比如Python脚本，HTML，PDF等，所以它可以记录我们的演算过程。很多课程，博客以及书籍都是用Notebook写的。

### 运行

当上面的中文的IPython后，IPython Notebook就已经算是安装好了。使用win+r打开运行窗口，输入ipython notebook，如果正确安装的话，这个命令就会默认在本地8888端口启动一个web服务，并自动打开浏览器，打开<http://localhost:8888/tree>页面，在这个页面我们可以看到当前目录下的所有文件夹以及ipynb文件。





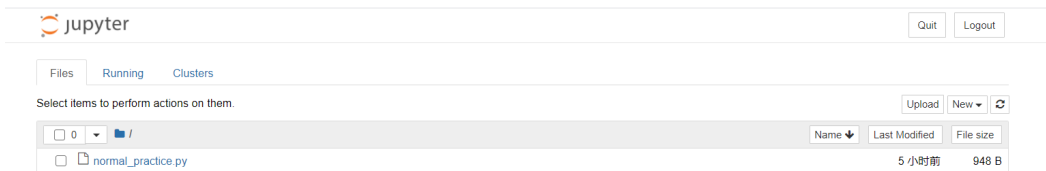
可以自己设置相应的目录打开，首先在cmd中进入相应的目录，然后输入ipython notebook即可进入。

```
C:\Windows\system32\cmd.exe - ipython notebook
Microsoft Windows [版本 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>D:
D:\>cd pycharm
D:\Pycharm>cd pytorchworks
D:\Pycharm\pytorchworks>ipython notebook
[TerminalIPythonApp] WARNING | Subcommand `ipython notebook` is deprecated and will be removed in future versions.
[TerminalIPythonApp] WARNING | You likely want to use `jupyter notebook` in the future
[I 15:48:57.472 NotebookApp] The port 8888 is already in use, trying another port.
[I 15:48:57.602 NotebookApp] JupyterLab extension loaded from D:\ANACONDA3\lib\site-packages\jupyterlab
[I 15:48:57.603 NotebookApp] JupyterLab application directory is D:\ANACONDA3\share\jupyter\lab
[I 15:48:57.606 NotebookApp] Serving notebooks from local directory: D:\Pycharm\pytorchworks
[I 15:48:57.606 NotebookApp] The Jupyter Notebook is running at:
[I 15:48:57.607 NotebookApp] http://localhost:8889/?token=0fb5bc046fd65caa73215abf6c2a122f5ab04cc587111997
[I 15:48:57.607 NotebookApp] or http://127.0.0.1:8889/?token=0fb5bc046fd65caa73215abf6c2a122f5ab04cc587111997
[I 15:48:57.607 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 15:48:57.648 NotebookApp]

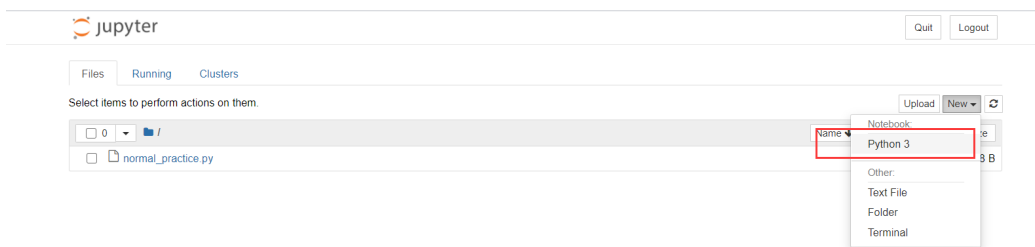
To access the notebook, open this file in a browser:
    file:///C:/Users/Administrator/AppData/Roaming/jupyter/runtime/nbserver-6012-open.html
Or copy and paste one of these URLs:
    http://localhost:8889/?token=0fb5bc046fd65caa73215abf6c2a122f5ab04cc587111997
    or http://127.0.0.1:8889/?token=0fb5bc046fd65caa73215abf6c2a122f5ab04cc587111997
```

进入到相应的浏览器页面，这个文件夹的内容会显示在上面。



可以打开已有的notebook文件，也可以新建一份





## 操作

- 1、**ctrl+enter** 会显示运行结果而不会创建新的输入框
- 2、**shift+enter** 小格子内的所有代码会运行，运行结果会立即出现在输出区域显示，且会创建一个新的输入框。
- 3、在一个输入框即Cell中使用回车即 **Enter** 键，表示换行，也就是说一个Cell中可以输入多条语句。



可以选择是文本格式，还是代码格式



# jupyter

## 基本操作

使用conda list查看所有conda中安装的包

```
(base) C:\Users\Administrator>conda list
# packages in environment at D:\ANACONDA3:
#
# Name                                Version                                Build                                Channel
_anaconda_depends                     2020.07                               py38_0                             https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/linux/
_ipyw_jlab_nb_ext_conf                 0.1.0                                py38_0                             defaults
anaconda                               custom                               py38_1                             https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/linux/
anaconda-navigator                     1.9.12                               py38_0                             defaults
bkcharts                              0.2                                   py38_0                             defaults
blas                                  1.0                                   mk1                                 defaults
ca-certificates                       2020.6.20                             hecda079_0                         https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/linux/
certifi                               2020.6.20                             py38h9bdc248_2                     https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/linux/
conda                                  4.9.0                                 py38h9bdc248_1                     https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/linux/
console_shortcut                       0.1.1                                4                                  defaults
get_terminal_size                     1.0.0                                h38e98db_0                         defaults
icc_rt                                2019.0.0                              h0cc432a_1                         defaults
intel-openmp                          2020.1                                216                                defaults
m2w64-gcc-libgfortran                 5.3.0                                 6                                  defaults
m2w64-gcc-libstdc++11                 5.3.0                                 7                                  defaults
m2w64-gcc-libstdc++6                   5.3.0                                 7                                  defaults
m2w64-gmp                              6.1.0                                 2                                  defaults
m2w64-libwinpthread-git                5.0.0.4634.697f757                   2                                  defaults
msys2-conda-epoch                     20160418                              1                                  defaults
navigator-updater                     0.2.1                                py38_0                             defaults
numpy                                  1.19.2                               py38hdf1ac2f_1                     https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/linux/
```

jupyter主要依靠的是这个包

```
get_terminal_size                     1.0.0                                h38e98db_0                         defaults
gevent                                1.4.0                                py36he774522_0                     defaults
glob2                                  0.7                                   py_0                               defaults
greenlet                              0.4.15                               py36hfa6e2cd_0                     defaults
h5py                                   2.9.0                                py36h5e291fa_0                     defaults
hdf5                                   1.10.4                               h7ebc959_0                         defaults
heapdict                              1.0.1                                py_0                               defaults
html5lib                              1.0.1                                py36_0                             defaults
icc_rt                                2019.0.0                              h0cc432a_1                         defaults
icu                                    58.2                                  ha66f8fd_1                         defaults
idna                                   2.8                                   py36_0                             defaults
imageio                               2.6.1                                py36_0                             defaults
imagesize                             1.1.0                                py36_0                             defaults
importlib_metadata                    0.23                                  py36_0                             defaults
intel-openmp                          2019.4                                245                                defaults
ipykernel                             5.1.3                                py36h39e3cac_0                     defaults
ipython                               7.9.0                                py36h39e3cac_0                     defaults
ipython_genutils                      0.2.0                                py36h3c5d0ee_0                     defaults
ipywidgets                            7.5.1                                py_0                               defaults
iso-639                               0.4.5                                py_0                               defaults
iso-639-2                             1.0                                   py_0                               defaults
isodate                               0.6.0                                py_0                               defaults
isort                                  4.3.21                               py36_0                             defaults
itsdangerous                          1.1.0                                py36_0                             defaults
jdcal                                  1.4.1                                py_0                               defaults
jedi                                   0.15.1                               py36_0                             defaults
jinja2                                2.10.3                               py_0                               defaults
```

进入到pytorch虚拟环境，我们选择在虚拟环境中安装jupyter

进入虚拟环境：conda activate pytorch

之后再输入代码：conda install nb\_conda 进行安装

```
setuptools 41.6.0 py36_0 defaults
six 1.12.0 py36_0 defaults
sqlite 3.30.1 he774522_0 defaults
tk 3.6.8 hfa6e2cd_0 defaults
torchvision 0.4.1 py36_cu92 pytorch
vc 14.1 h0510ff6_4 defaults
vs2015_runtime 14.16.27012 hf0eaf9b_0 defaults
wheel 0.33.6 py36_0 defaults
wincertstore 0.2 py36h7fe50ca_0 defaults
xz 5.2.4 h2fa13f4_4 defaults
zlib 1.2.11 h62dc97_3 defaults
zstd 1.3.7 h508b16e_0 defaults

(pytorch) C:\Users\Zhiyao>conda install nb_conda
Collecting package metadata (repodata.json): done
Solving environment: /
```

在pytorch环境中输入jupyter notebook 即可进入

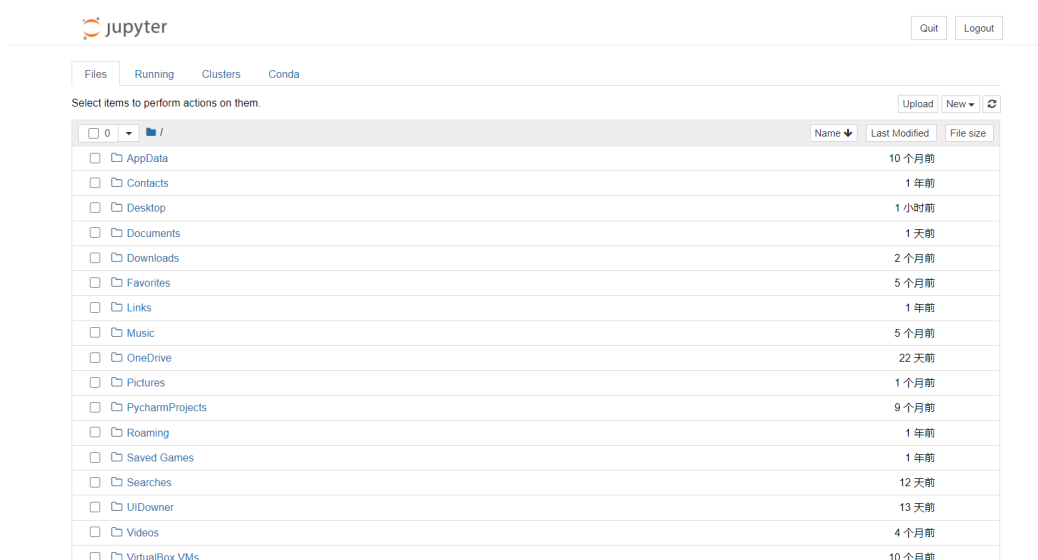
```
Anaconda Prompt (ANACONDA3) - jupyter notebook

(base) C:\Users\Administrator>conda activate pytorch

(pytorch) C:\Users\Administrator>jupyter notebook
[I 20:54:46.573 NotebookApp] [nb_conda_kernels] enabled, 2 kernels found
[I 20:54:48.090 NotebookApp] [nb_conda] enabled
[I 20:54:48.091 NotebookApp] Serving notebooks from local directory: C:\Users\Administrator
[I 20:54:48.091 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 20:54:48.092 NotebookApp] http://localhost:8888/?token=9d47cc8eba63fd568525594a133853d2ab4fec7fbc5f0644
[I 20:54:48.092 NotebookApp] or http://127.0.0.1:8888/?token=9d47cc8eba63fd568525594a133853d2ab4fec7fbc5f0644
[I 20:54:48.092 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 20:54:48.145 NotebookApp]

To access the notebook, open this file in a browser:
file:///C:/Users/Administrator/AppData/Roaming/jupyter/runtime/nbserver-9768-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=9d47cc8eba63fd568525594a133853d2ab4fec7fbc5f0644
or http://127.0.0.1:8888/?token=9d47cc8eba63fd568525594a133853d2ab4fec7fbc5f0644
```

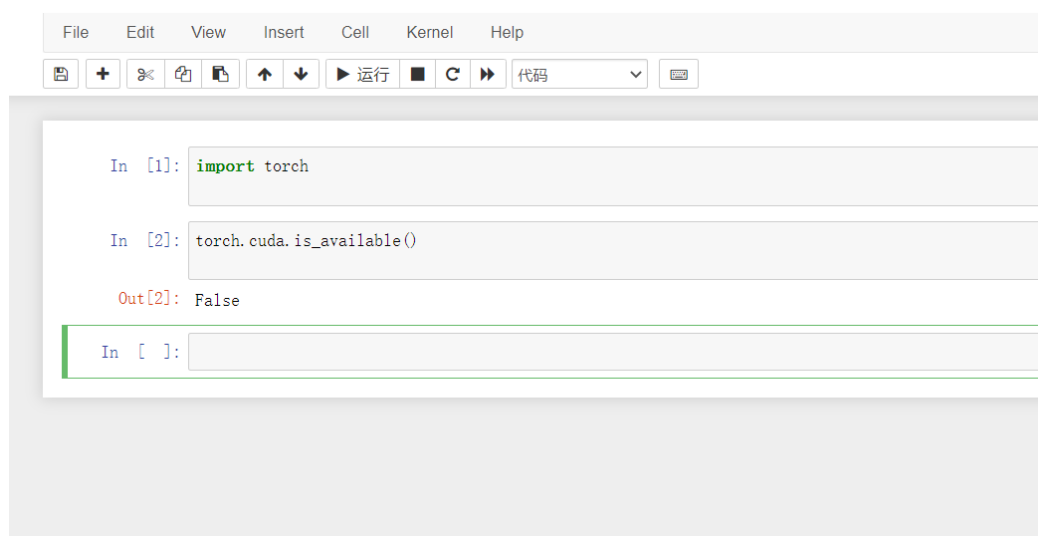
下面为打开的页面



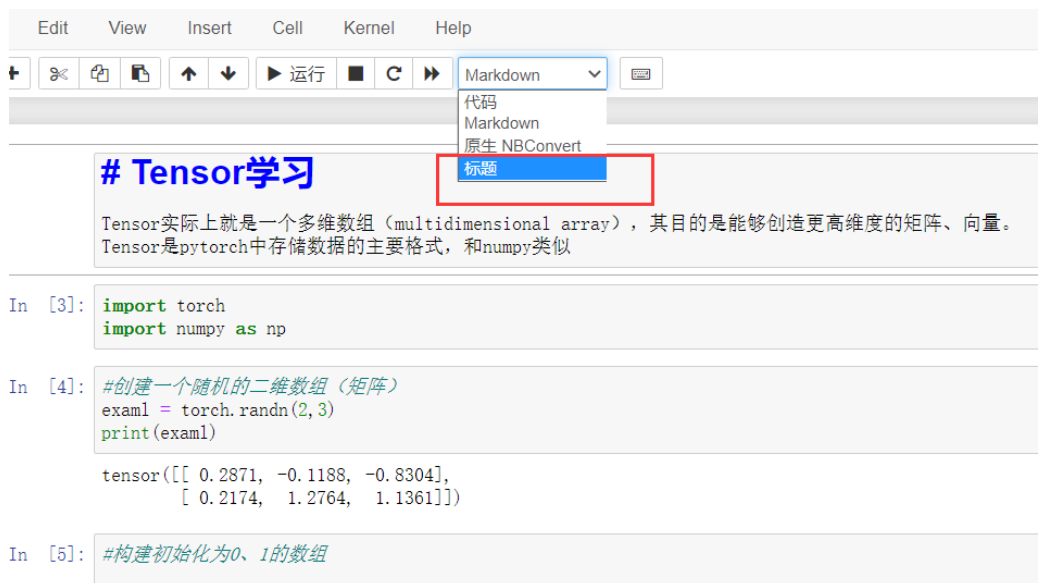
使用虚拟环境的pytorch创建案例



检验是否能运行，操作和使用ipython notebook 进入的页面一样



添加标题，通过选择格式添加标题



通过在前面添加不同数量的# 号来表示这是几级标题

## # Tensor学习

Tensor实际上就是一个多维数组 (multidimensional array)，其目的是能够创造更高维度的矩阵、向量。Tensor是pytorch中存储数据的主要格式，和numpy类似。

```
In [3]: import torch
import numpy as np
```

```
In [4]: #创建一个随机的二维数组 (矩阵)
exam1 = torch.randn(2, 3)
print(exam1)

tensor([[ 0.2871, -0.1188, -0.8304],
        [ 0.2174,  1.2764,  1.1361]])
```

```
In [5]: #构建初始化为0、1的数组

exam2 = torch.zeros(2, 3)
exam3 = torch.ones(2, 3)

print(exam2)
print(exam3)

tensor([[0., 0., 0.],
        [0., 0., 0.]])
tensor([[1., 1., 1.],
        [1., 1., 1.]])
```

```
In [7]: #从python的数组直接进行构造

exam4 = torch.Tensor([[1, 2, 4], [2, 3, 6]])
print(exam4)

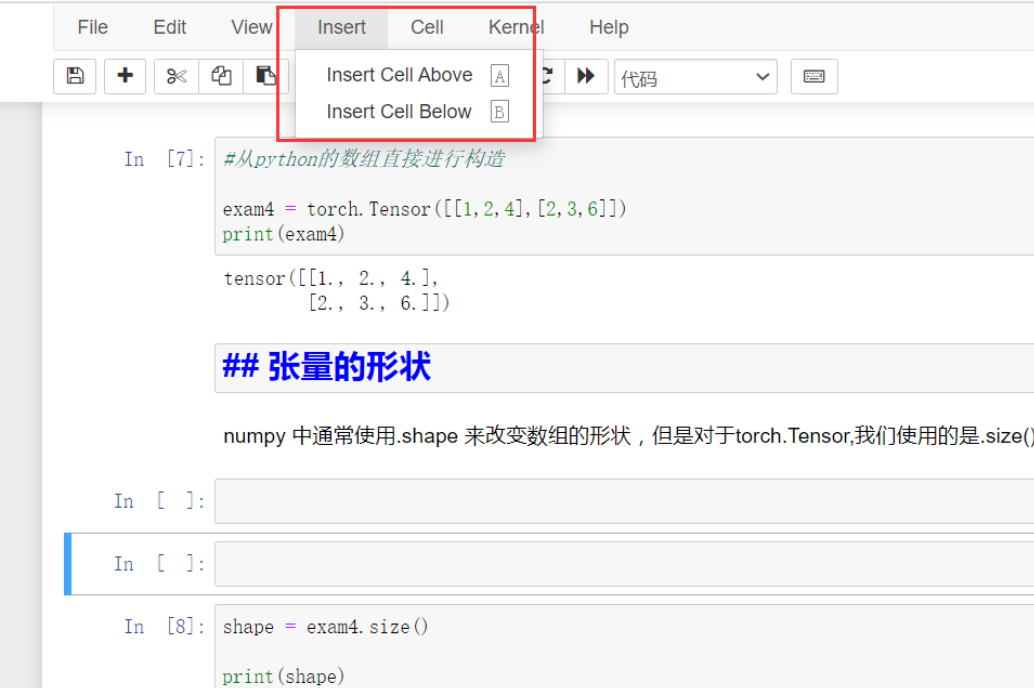
tensor([[1.,  2.,  4.],
        [2.,  3.,  6.]])
```

## ## 张量的形状

numpy 中通常使用.shape 来改变数组的形状，但是对于torch.Tensor,我们使用的是.size()

增添行

用鼠标点击已经运行的行，按键盘上的A、B 分别为向上添加一行或者向下添加一行。



```
In [7]: #从python的数组直接进行构造

exam4 = torch.Tensor([[1, 2, 4], [2, 3, 6]])
print(exam4)

tensor([[1.,  2.,  4.],
        [2.,  3.,  6.]])

## 张量的形状

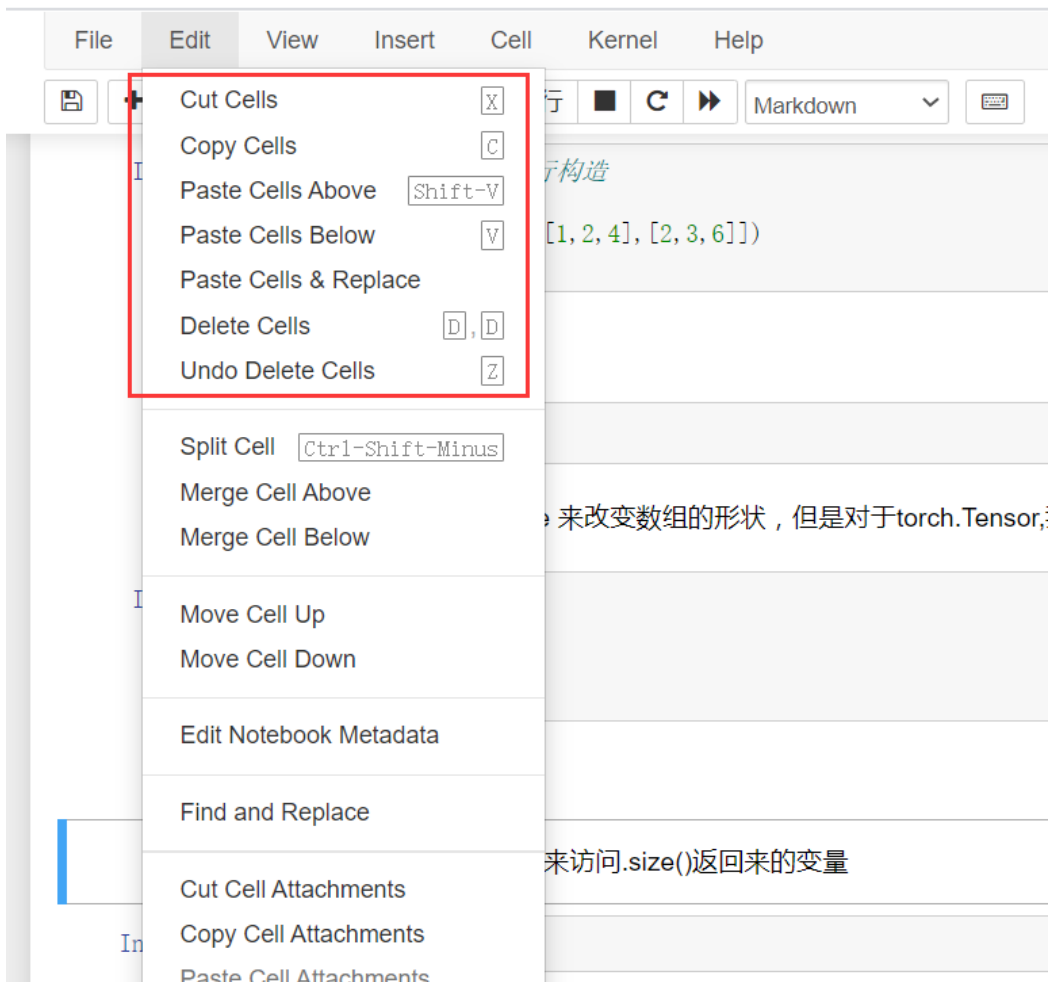
numpy 中通常使用.shape 来改变数组的形状，但是对于torch.Tensor,我们使用的是.size()

In [ ]:

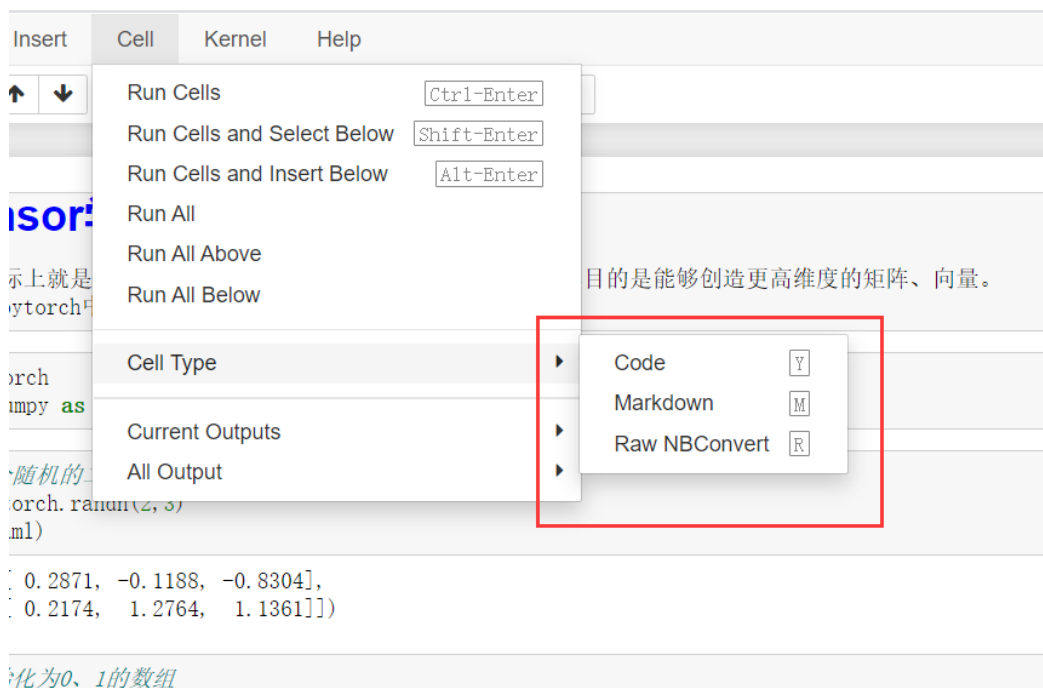
In [ ]:

In [8]: shape = exam4.size()

print(shape)
```



改变块的种类快捷键：Y M R。y是代码块，M是标签，R是块。



按H会出现快捷键栏

命令行模式(按 **Esc** 生效)

编辑快捷键

|                                      |  |
|--------------------------------------|--|
| <b>F</b> : 查找并且替换                    | <b>Shift-J</b> : 扩展下面选择的代码块                |
| <b>Ctrl-Shift-F</b> : 打开命令配置         | <b>Ctrl-A</b> : select all cells           |
| <b>Ctrl-Shift-P</b> : 打开命令配置         | <b>A</b> : 在上面插入代码块                        |
| <b>Enter</b> : 进入编辑模式                | <b>B</b> : 在下面插入代码块                        |
| <b>P</b> : 打开命令配置                    | <b>X</b> : 剪切选择的代码块                        |
| <b>Shift-Enter</b> : 运行代码块, 选择下面的代码块 | <b>C</b> : 复制选择的代码块                        |
| <b>Ctrl-Enter</b> : 运行选中的代码块         | <b>Shift-V</b> : 粘贴到上面                     |
| <b>Alt-Enter</b> : 运行代码块并且插入下面       | <b>V</b> : 粘贴到下面                           |
| <b>Y</b> : 把代码块变成代码                  | <b>Z</b> : 撤销删除                            |
| <b>M</b> : 把代码块变成标签                  | <b>D, D</b> : 删除选中单元格                      |
| <b>R</b> : 清除代码块格式                   | <b>Shift-M</b> : 合并选中单元格, 如果只有一个单<br>元格被选中 |
| <b>1</b> : 把代码块变成heading 1           | <b>Ctrl-S</b> : 保存并检查                      |
| <b>2</b> : 把代码块变成heading 2           | <b>S</b> : 保存并检查                           |
| <b>3</b> : 把代码块变成heading 3           | <b>L</b> : 切换行号                            |
| <b>4</b> : 把代码块变成heading 4           | <b>O</b> : 选择单元格的输出                        |
| <b>5</b> : 把代码块变成heading 5           | <b>Shift-O</b> : 切换选定单元的输出滚动               |
| <b>6</b> : 把代码块变成heading 6           |  |

关闭

## 快捷键

|                             |   |
|-----------------------------|---|
| <b>K</b> : 选择上面的代码块         | <b>H</b> : 显示快捷键                        |
| <b>上</b> : 选择上面的代码块         | <b>I, I</b> : 中断服务                      |
| <b>下</b> : 选择下面的代码块         | <b>O, O</b> : 重启服务(带窗口)                 |
| <b>J</b> : 选择下面的代码块         | <b>Esc</b> : 关闭页面                       |
| <b>Shift-K</b> : 扩展上面选择的代码块 | <b>Q</b> : 关闭页面                         |
| <b>Shift-上</b> : 扩展上面选择的代码块 | <b>Shift-L</b> : 在所有单元格中切换行号, 并保持<br>设置 |
| <b>Shift-下</b> : 扩展下面选择的代码块 | <b>Shift-空格</b> : 向上滚动                  |
|                             | <b>空格</b> : 向下滚动                        |

编辑模式(按 **Enter** 生效)

|                         |                              |
|-------------------------|------------------------------|
| <b>Tab</b> : 代码完成或缩进    | <b>Ctrl-右</b> : 跳到单词右边       |
| <b>Shift-Tab</b> : 工具提示 | <b>Ctrl-删除</b> : 删除前面的单词     |
| <b>Ctrl-J</b> : 缩进      | <b>Ctrl-Delete</b> : 删除后面的单词 |
| <b>Ctrl-I</b> : 取消缩进    | <b>Ctrl-Y</b> : 重做           |
| <b>Ctrl-A</b> : 全选      | <b>Alt-U</b> : 重新选择          |
| <b>Ctrl-Z</b> : 撤销      | <b>Ctrl-M</b> : 进入命令行模式      |
| <b>Ctrl-/</b> : 评论      | <b>Ctrl-Shift-F</b> : 打开命令配置 |
| <b>Ctrl-D</b> : 删除整行    | <b>Ctrl-Shift-P</b> : 打开命令配置 |
| <b>Ctrl-U</b> : 撤销选择    | <b>Esc</b> : 进入命令行模式         |

编辑模式(按 `Enter` 生效)

`Tab`: 代码完成或缩进

`Shift-Tab`: 工具提示

`Ctrl-]`: 缩进

`Ctrl-[`: 取消缩进

`Ctrl-A`: 全选

`Ctrl-Z`: 撤销

`Ctrl-/`: 评论

`Ctrl-D`: 删除整行

`Ctrl-U`: 撤销选择

`Insert`: 切换 重写标志

`Ctrl-Home`: 跳到单元格起始处

`Ctrl-上`: 跳到单元格起始处

`Ctrl-End`: 跳到单元格最后

`Ctrl-下`: 跳到单元格最后

`Ctrl-左`: 跳到单词左边

`Ctrl-右`: 跳到单词右边

`Ctrl-删除`: 删除前面的单词

`Ctrl-Delete`: 删除后面的单词

`Ctrl-Y`: 重做

`Alt-U`: 重新选择

`Ctrl-M`: 进入命令行模式

`Ctrl-Shift-F`: 打开命令配置

`Ctrl-Shift-P`: 打开命令配置

`Esc`: 进入命令行模式

`Shift-Enter`: 运行代码块, 选择下面的代码块

`Ctrl-Enter`: 运行选中的代码块

`Alt-Enter`: 运行代码块并且插入下面

`Ctrl-Shift-Minus`: split cell at cursor(s)

`Ctrl-S`: 保存并检查

`下`: 光标下移

`上`: 光标上移

空格: 向下滚动

关闭

## 改变jupyter风格

参考: <https://www.cnblogs.com/6b7b5fc3/p/13488264.html>

在虚拟环境中。在conda中进入到相应的虚拟环境中。

conda activate pytorch

进入jupyter: jupyter notebook

使用下列代码安装jupyter\_themes(如果是控制台, 则将conda换成pip)

```
conda install jupyterthemes
```

使用以下命令可以查看所有的主题:

```
jt -l
```



```
(pytorch) C:\Users\Administrator>jt -l
Available Themes:
  chesterish
  grade3
  gruvboxd
  gruvboxl
  monokai
  oceans16
  onedork
  solarizedd
  solarizedl
```

使用以下命令选择要使用的主题：

```
jt -t 主题名称
```

```
(pytorch) C:\Users\Administrator>jt -l
Available Themes:
  chesterish
  grade3
  gruvboxd
  gruvboxl
  monokai
  oceans16
  onedork
  solarizedd
  solarizedl

(pytorch) C:\Users\Administrator>jt -t chesterish

(pytorch) C:\Users\Administrator>
```

chesterish: 适合夜间

File Edit View Insert Cell Kernel Help

pytorch-切片和索引

In [1]: 1 import torch

In [2]: 1 a = torch.rand(4, 3, 28, 28)

In [5]: 1 a[0].shape  
torch.Size([3, 28, 28])

In [6]: 1 a[0, 0].shape  
torch.Size([28, 28])

In [7]: 1 #返回的是标量  
2 a[0, 0, 2, 4]

grade3: 主题为灰色

File Edit View Insert Cell Kernel Help 可信的 Python 3

pytorch-切片和索引

In [1]: 1 import torch

In [2]: 1 a = torch.rand(4, 3, 28, 28)

In [5]: 1 a[0].shape  
torch.Size([3, 28, 28])

In [6]: 1 a[0, 0].shape  
torch.Size([28, 28])

In [7]: 1 #返回的是标量  
2 a[0, 0, 2, 4]  
tensor(0.4782)

In [8]: 1 a.shape

gruvboxd:主题为黄色

File Edit View Insert Cell Kernel Help 可信的 Python 3

pytorch-切片和索引

In [1]: 1 import torch

In [2]: 1 a = torch.rand(4, 3, 28, 28)

In [5]: 1 a[0].shape  
  
torch.Size([3, 28, 28])

In [6]: 1 a[0, 0].shape  
  
torch.Size([28, 28])

In [7]: 1 #返回的是标量  
2 a[0, 0, 2, 4]  
  
tensor(0.4782)

gruvboxl: 暖黄色

File Edit View Insert Cell Kernel Help 可信的 Python 3

pytorch-切片和索引

1 这是一篇关于pytorch张量切片与索引的笔记

In [1]: 1 import torch

In [2]: 1 a = torch.rand(4, 3, 28, 28)

In [5]: 1 a[0].shape  
  
torch.Size([3, 28, 28])

In [6]: 1 a[0, 0].shape  
  
torch.Size([28, 28])

In [7]: 1 #返回的是标量  
2 a[0, 0, 2, 4]  
  
tensor(0.4782)

monokai: 主题篇荧光色

File Edit View Insert Cell Kernel Help 可信的 Python 3

pytorch-切片和索引

这是一篇关于pytorch张量切片与索引的笔记

In [1]: 1 import torch

In [2]: 1 a = torch.rand(4, 3, 28, 28)

In [5]: 1 a[0].shape  
  
torch.Size([3, 28, 28])

In [6]: 1 a[0, 0].shape  
  
torch.Size([28, 28])

ocean 16

FileEditViewInsertCellKernelHelp可信的Python 3

pytorch-切片和索引

这是一篇关于pytorch张量切片与索引的笔记

In [1]:

1 import torch

In [2]:

1 a = torch.rand(4, 3, 28, 28)

In [5]:

1 a[0].shape

torch.Size([3, 28, 28])

In [6]:

1 a[0,0].shape

torch.Size([28, 28])

In [7]:

1 #返回的是标量

2 a[0,0,2,4]

tensor(0.4782)

onedork

FileEditViewInsertCellKernelHelp可信的Python 3

pytorch-切片和索引

这是一篇关于pytorch张量切片与索引的笔记

In [1]:

1 import torch

In [2]:

1 a = torch.rand(4, 3, 28, 28)

In [5]:

1 a[0].shape

torch.Size([3, 28, 28])

In [6]:

1 a[0,0].shape

torch.Size([28, 28])

In [7]:

1 #返回的是标量

2 a[0,0,2,4]

tensor(0.4782)

solarizedd: 比较模糊

FileEditViewInsertCellKernelHelp可信的Python

pytorch-切片和索引

这是一篇关于pytorch张量切片与索引的笔记

In [1]:

1 import torch

In [2]:

1 a = torch.rand(4, 3, 28, 28)

In [5]:

1 a[0].shape

torch.Size([3, 28, 28])

In [6]:

1 a[0, 0].shape

torch.Size([28, 28])

In [7]:

1 #返回的是标量

2 a[0, 0, 2, 4]

tensor(0.4782)

In [8]:

solarizedl

FileEditViewInsertCellKernelHelp可信的Python 3

pytorch-切片和索引

这是一篇关于pytorch张量切片与索引的笔记

In [1]:

1 import torch

In [2]:

1 a = torch.rand(4, 3, 28, 28)

In [5]:

1 a[0].shape

torch.Size([3, 28, 28])

In [6]:

1 a[0, 0].shape

torch.Size([28, 28])

In [7]:

1 #返回的是标量

2 a[0, 0, 2, 4]

tensor(0.4782)

In [8]: