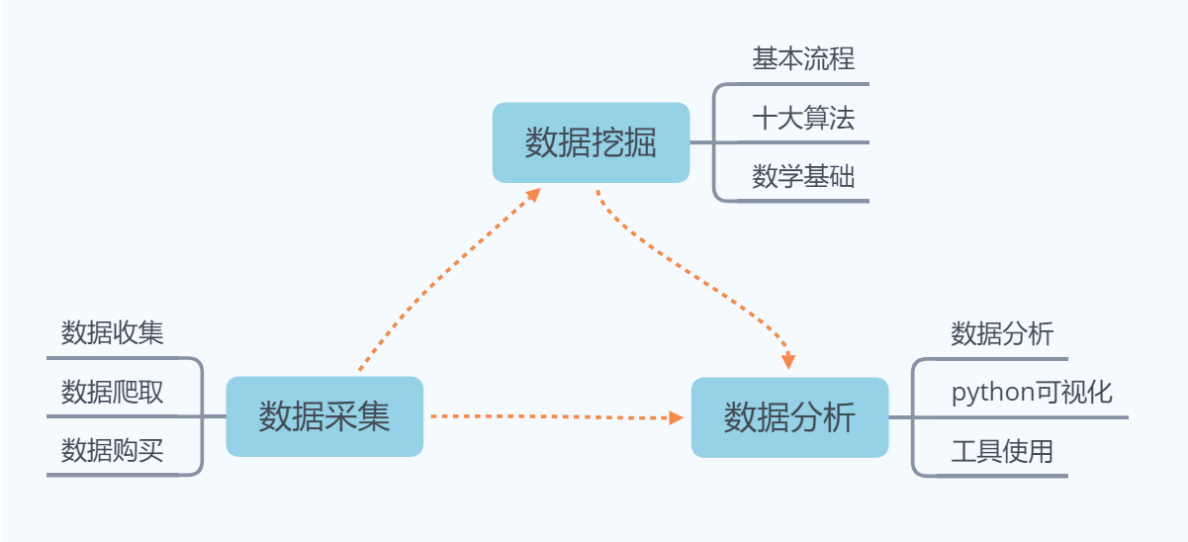
01 数据分析基础

一、数据分析介绍

当我们谈论数据分析的时候，都在讲些什么呢？

这里我可以把数据分析分成三个重要的组成部分。

1. **数据采集。**它是数据分析的原材料，也是最“**接地气**”的部分，因为任何分析都要有数据源。
2. **数据分析。**就是针对现有的数据，进行专业的技术进行分析，并得到一份分析报告。
3. **数据可视化。**它可以说是数据领域中万金油的技能，可以让我们直观地了解到数据分析的结果。
4. **数据挖掘。**它最核心的部分，也是整个商业价值所在。之所以要进行数据分析，就是要找到其中的规律，来指导我们的业务，或者是给领导做出决策提供参考信息。因此**数据挖掘的核心是挖掘数据的商业价值，也就是我们所谈的商业智能BI。**



1、1 生活中的数据分析

1、1、1 无处不在的数据

> 生活中存在各式各样的数据，那么基于这些数据，我们又能做哪些分析呢？

最近几年大数据这个词是火的不行，确实随着社会科技水平的提高，我们使用电子设备的时间越来越长，现在数据的增长量真的非常非常快，这些数据来自各个领域，比如：

- 社交：微信，微博，知乎，豆瓣什么的
- 交通：出租车，公交车等类的数据，比如：滴滴出行
- 金融：股票历史交易信息，公司财报，新闻媒体的态度等等
- 医疗：在数据收集和存储上还有很长的路要走

典型的数据分析应用：

- 竞选预测：特朗普和拜登
- 拥堵预测：交通出行，地图导航
- 信誉评估：信用额度贷款等
- 辅助诊断：医疗影像等方面，比如 ： CT图像中的肿瘤

1、1、2 数据分析的作用

数据分析是指用适当的统计分析方法对收集来的大量数据进行分析，提取有用信息和形成结论而对数据加以详细研究和概括总结的过程。

数据分析的目的有多种，概括起来有三种：

- 现状分析:告诉你过去发生了什么。　　探索型数据分析
- 原因分析:告诉你某一现状为什么发生。　　验证型数据分析
- 预测分析:预测未来会发生什么。　　预测型数据分析

1、1、3 为什么需要数据分析能力?

举例：

1. 通过数据分析，我们可以更好地了解用户画像，为企业做留存率、流失率等指标分析，进而精细化产品运营。
2. 面对生活中遇到的种种麻烦，数据分析也可以提供解决方案，比如信用卡反欺诈，自动屏蔽垃圾邮件等。
3. 如果你关注比特币，数据分析可以帮助你预测比特币的走势。

案例分析

故事内容这个故事发生于20世纪90年代的美国超市中，超市管理人员分析销售数据时发现了一个令人难于理解的现象：在某些特定的情况下，“啤酒”与“尿布”两件看上去毫无关系的商品会经常出现在同一个购物篮中，这种独特的销售现象引起了管理人员的注意，经过后续调查发现，这种现象出现在年轻的父亲身上。

故事起因在美国有婴儿的家庭中，一般是母亲在家中照看婴儿，年轻的父亲前去超市购买尿布。父亲在购买尿布的同时，往往会顺便为自己购买啤酒，这样就会出现啤酒与尿布这两件看上去不相干的商品经常会出现在同一个购物篮的现象。

如果这个年轻的父亲在卖场只能买到两件商品之一，则他很有可能会放弃购物而到另一家商店，直到可以一次同时买到啤酒与尿布为止。

超市发现了这一独特的现象，开始在卖场尝试将啤酒与尿布摆放在相同的区域，让年轻的父亲可以同时找到这两件商品，并很快地完成购物。

故事小结这个故事是因为有数据分析的结果支持才会获得成功并得到广泛传播，通过分析购物篮中的商品集合数据，找出商品之间的关联关系，发现客户的购买行为，从而获得更多的商品销售收入

> 经典数据挖掘案例：https://www.sohu.com/a/243626833_99923499

二、环境搭建

2、1 Anaconda：

`Anaconda（水蟒）`是一个捆绑了`Python`、`conda`、其他相关依赖包的一个软件。包含了180多个科学计算包及其依赖。`Anaconda3`是集成了`Python3`的环境，`Anaconda2`是集成了`Python2`的环境。`Anaconda`默认集成的包，是属于内置的`Python`的包。并且支持绝大部分操作系统（比如：Windows、Mac、Linux等）。下载地址如下：`https://www.anaconda.com/distribution/`（如果官网下载太慢，可以在清华大学开源软件站中下载：`https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/`）。根据自己的操作系统，下载相

应的版本，因为`Anaconda`内置了许多的包，所以安装过程需要耗费相当长的时间，大家在安装的时候需要耐心等待。在安装完成后，会有以下几个模块：`Anaconda prompt`、`Anaconda Navigator`、`Spyder`、`jupyter notebook`，以下分别做一些介绍。

2、1、1 Anaconda prompt：

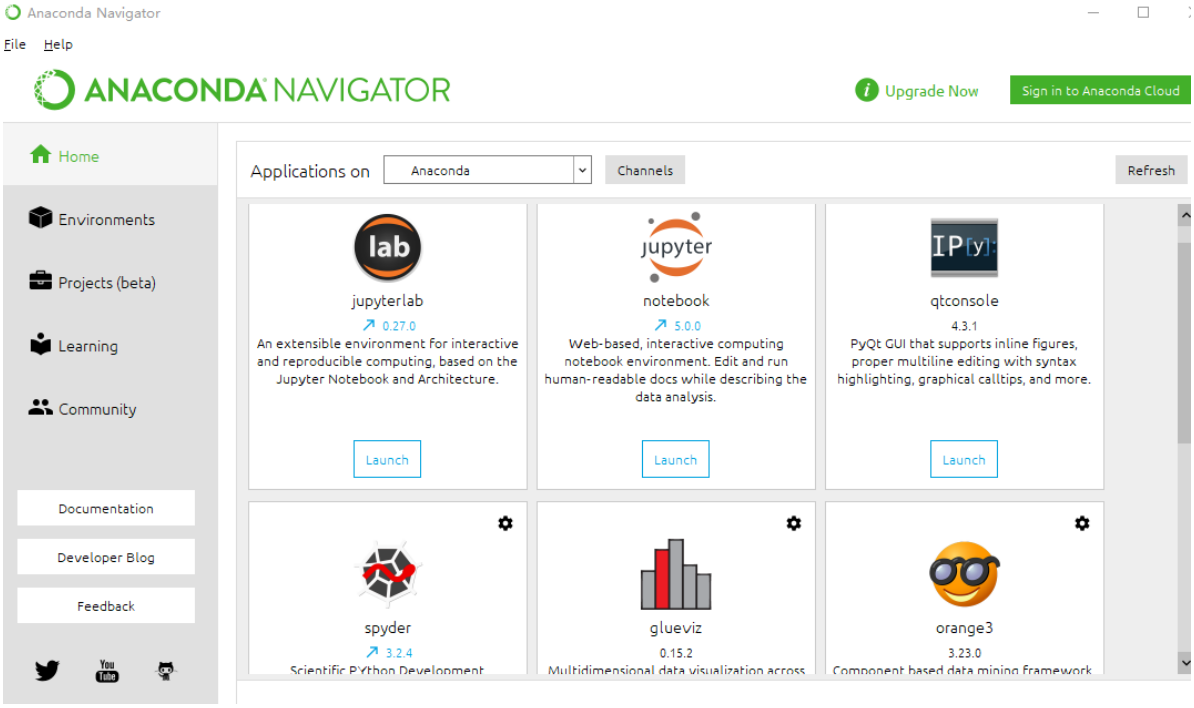
`Anaconda prompt`是专门用来操作`anaconda`的终端。如果你安装完`Anaconda`后没有在环境变量的`PATH`中添加相关的环境变量，那么以后你想在终端使用`anaconda`相关的命令，则必须要在`Anaconda prompt`中完成。

2、1、2 Anaconda Navigator：

这个相当于是一个导航面板，上面组织了`Anaconda`相关的软件。

2、1、3 Spyder：

一个专门开发`Python`的软件，熟悉`MATLAB`的同学会比较有亲切感，但在后期的学习过程中，我们将不会使用这个工具写代码，因为还有更好的可替代的工具。



2、2 Jupyter 介绍

Jupyter项目是一个非盈利的开源项目，源于2014年的ipython项目，因为它逐渐发展为支持跨所有编程语言的交互式数据科学和科学计算

- Jupyter Notebook，原名IPython Notbook，是IPython的加强网页版，一个开源Web应用程序
- 名字源自Julia、Python 和 R（数据科学的三种开源语言）
- 是一款程序员和科学工作者的**编程/文档/笔记/展示**软件
- **.ipynb**文件格式是用于计算型叙述的**JSON文档格式**的正式规范

Jupyter项目旨在开发跨几十种编程语言的开源软件，开放标准和用于交互式计算的服务。

2、2、1 为什么使用 Jupyter Notebook？

- 传统软件开发： 工程 / 目标明确
  - 需求分析，设计架构，开发模块，测试
- 数据挖掘：艺术 / 目标不明确
  - 目的是具体的洞察目标，而不是机械的完成任务
  - 通过执行代码来理解问题
  - 迭代式地改进代码来改进解决方法

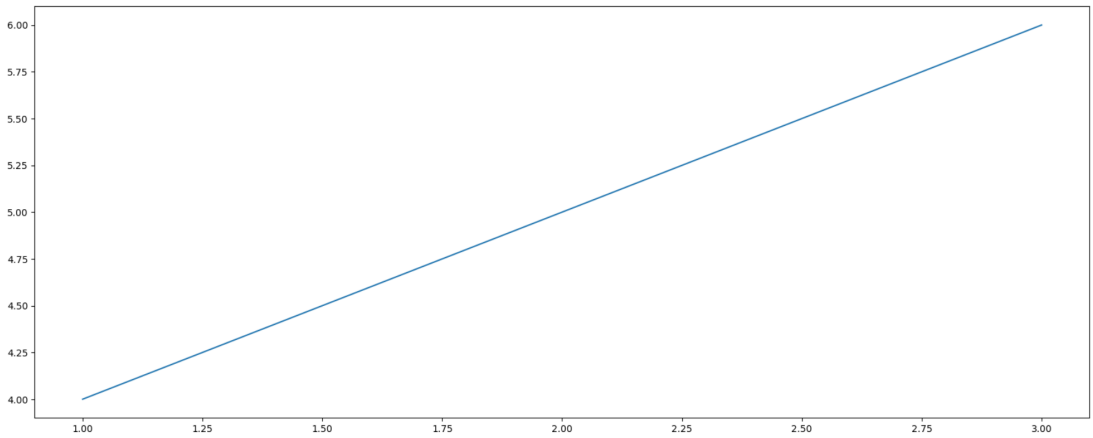
实时运行的代码、叙事性的文本和可视化被整合在一起，方便使用代码和数据来讲述故事

一个Python编辑环境，可以实时的查看代码的运行效果。

2、2、2 对比 Jupyter Notebook 和 Pycharm

画图

```
In [1]: import matplotlib.pyplot as plt
        %matplotlib inline
        plt.figure(figsize=(20, 8), dpi=100)
        plt.plot([1,2,3], [4,5,6])
        plt.show()
```



数据展示

```
In [2]: import pandas as pd
        stock_day = pd.read_csv("./stock_day/stock_day.csv")
```

```
In [3]: stock_day
```

Out[3]:

	open	high	close	low	volume	price_change	p_change	ma5	ma10	ma20	v_ma5	v_ma10	v_ma20	turnover
2018-02-27	23.53	25.88	24.16	23.53	95578.03	0.63	2.68	22.942	22.142	22.875	53782.64	46738.65	55576.11	2.39
2018-02-26	22.80	23.78	23.53	22.80	60985.11	0.69	3.02	22.406	21.955	22.942	40827.52	42736.34	56007.50	1.53
2018-02-23	22.88	23.37	22.82	22.71	52914.01	0.54	2.42	21.938	21.929	23.022	35119.58	41871.97	56372.85	1.32
2018-02-22	22.25	22.76	22.28	22.02	36105.01	0.36	1.64	21.446	21.909	23.137	35397.58	39904.78	60149.60	0.90
2018-02-14	21.49	21.99	21.92	21.48	23331.04	0.44	2.05	21.366	21.923	23.253	33590.21	42935.74	61716.11	0.58
2018-02-13	21.40	21.90	21.48	21.31	30802.45	0.28	1.32	21.342	22.103	23.387	39694.65	45518.14	65161.68	0.77
2018-02-12	20.70	21.40	21.19	20.63	32445.39	0.82	4.03	21.504	22.338	23.533	44645.16	45679.94	68686.33	0.81
2018-02-09	21.20	21.46	20.36	20.19	54304.01	-1.50	-6.86	21.920	22.596	23.645	48624.36	48982.38	70552.47	1.36
2018-02-08	21.79	22.09	21.88	21.75	27068.16	0.09	0.41	22.372	23.009	23.839	44411.98	48612.16	73852.45	0.68
2018-02-07	22.69	23.11	21.80	21.29	53853.25	-0.50	-2.24	22.480	23.258	23.929	52281.28	56315.11	74925.33	1.35
2018-02-06	22.80	23.55	22.29	22.20	55555.00	-0.97	-4.17	22.864	23.607	24.029	51341.63	64413.58	75738.95	1.39
2018-02-05	22.45	23.39	23.27	22.25	52341.39	0.65	2.87	23.172	23.928	24.112	46714.72	69278.66	77070.00	1.31

总结：Jupyter Notebook 相比 Pycharm 在画图和数据展示方面更有优势。

2、3 Jupyter Notebook 的使用

2、3、1 jupyter 安装与适配

- 首先在终端下安装 jupyter，Anaconda环境下默认自带会安装

```
pip install jupyter
```

- 安装jupyter拓展库

```
pip install autopep8  # 安装pep8代码规范的模块
pip install jupyter_contrib_nbextensions  # 安装 jupyter 拓展包
pip install yapf  # 安装拓展包依赖的第三方功能模块
```

- 拓展包安装后需要执行命令适配

```
"""拓展包适配"""
jupyter contrib nbextension install --user
jupyter nbextension enable code_prettify/code_prettify
```

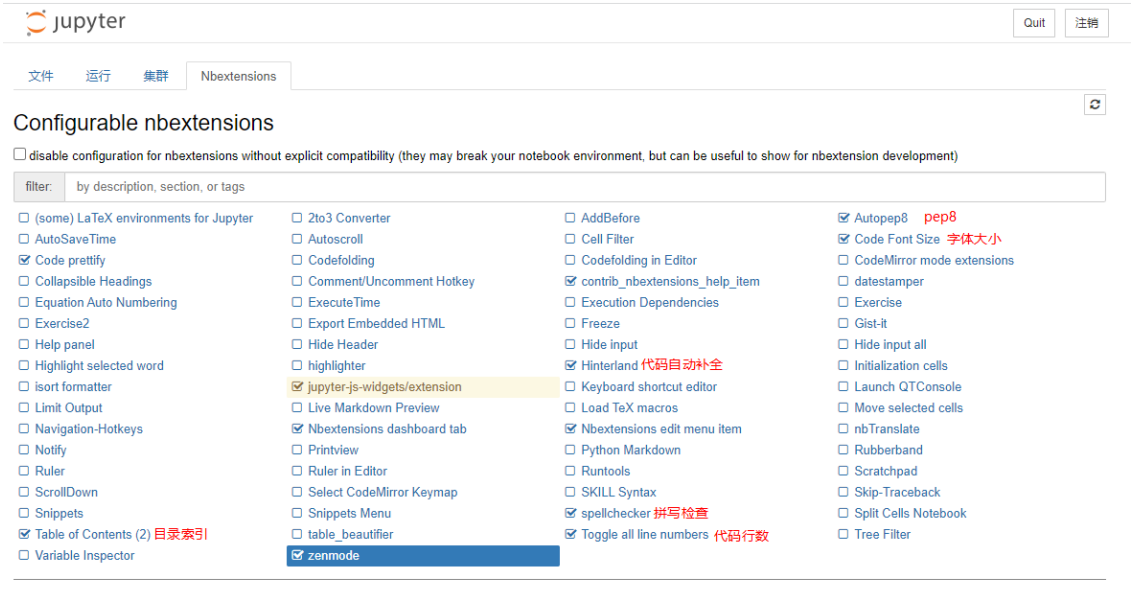
- 环境搭建好后，在命令行下输入 `jupyter notebook` 命令，会自动弹出浏览器窗口打开 Jupyter Notebook

```
# 输入命令
jupyter notebook
```

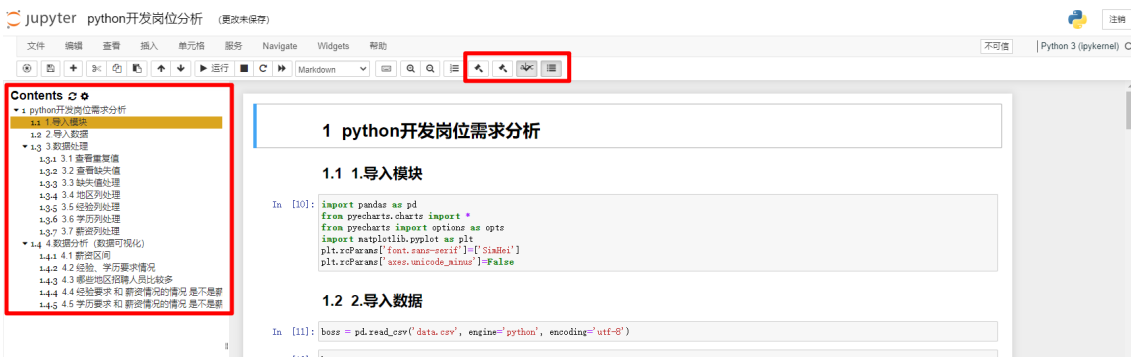
本地notebook的默认URL为：http://localhost:8888

> 想让notebook打开指定目录，只要进入此目录后执行命令即可

- 打开`jupyter notebook`以后，在`Nbextensions`选项下勾选配置选项，如下图所示：



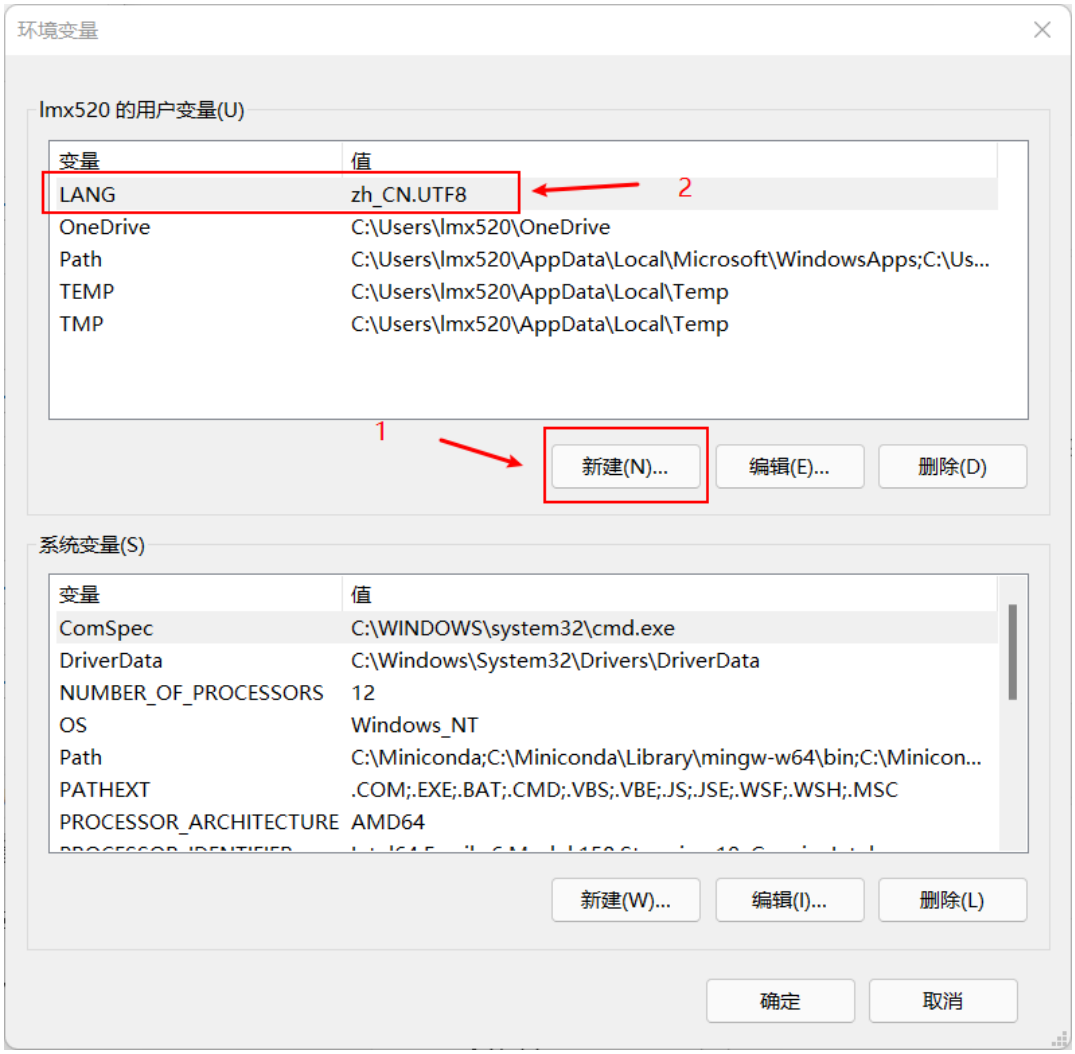
- 添加拓展配置以后的效果



- `Jupyter notebook`汉化，更改为中文界面
  - 首先桌面-此电脑-右击 属性，选择**高级系统设置**

  - 弹出的选项卡中，选择 高级-环境变量

- 选择上面的用户变量，是用户变量，点击新建，新建用户变量

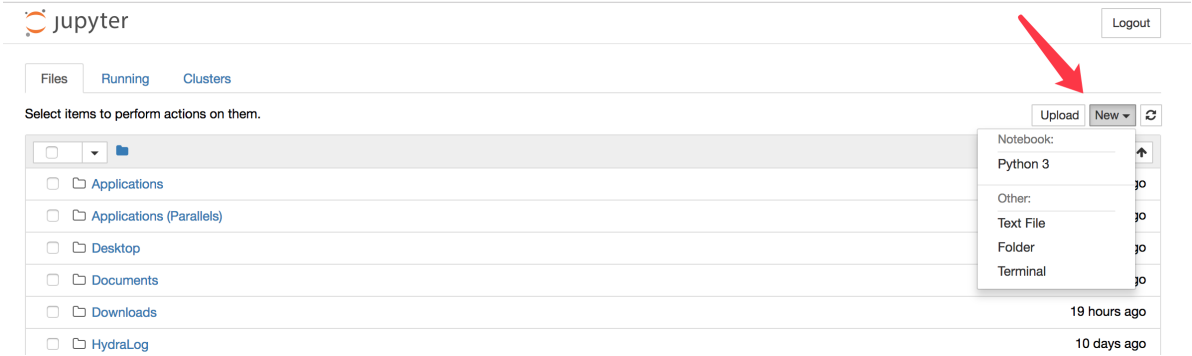


变量名：LANG
变量值：zh_CN.UTF8

- 重要：配置好后需要关闭cmd终端，重新启动`jupyter`才会生效。

2、3、2 创建文件

- 新建notebook文档



- 内容界面操作

标题栏：点击标题（如Untitled）修改文档名

编辑栏：

```
In [1]: print("hello world")
        hello world

In [2]: a = 1
        b = 2
        c = a + b

In [3]: a
Out[3]: 1

In [4]: b
Out[4]: 2

In [5]: c
Out[5]: 3

In [6]: import random
        random.random()
Out[6]: 0.9748235910569898
```

2、3、3 cell 单元格操作

- 什么是cell?
  - **cell**：一对In Out会话被视作一个代码单元，称为cell
  - cell行号前的 * ，表示代码正在运行

Jupyter支持两种模式：

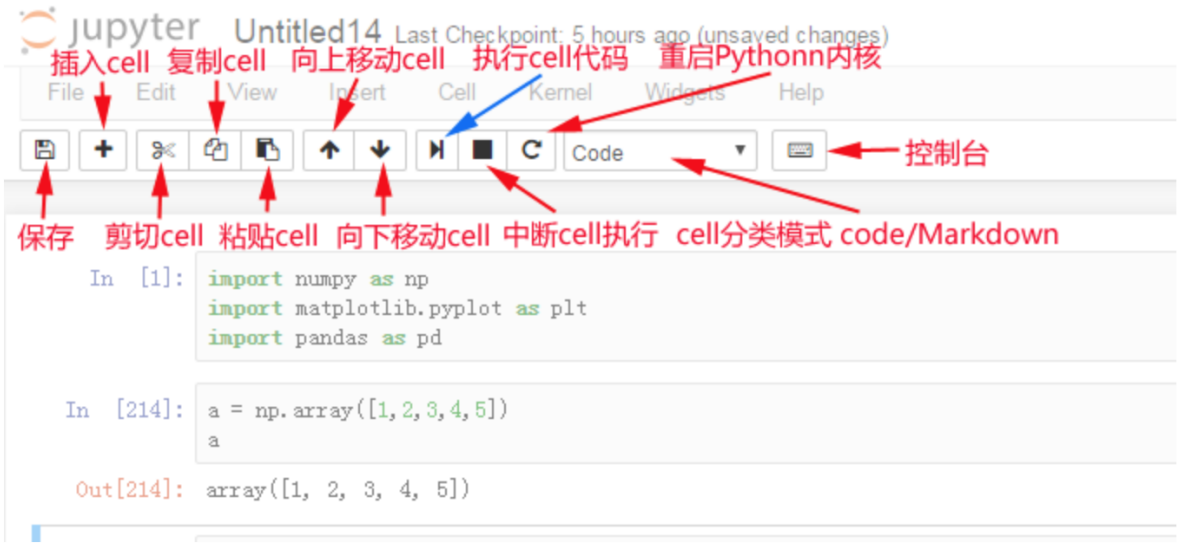
- 编辑模式（Enter）
  - 命令模式下`回车Enter`或`鼠标双击`cell进入编辑模式
  - 可以**操作cell内文本**或代码，剪切 / 复制 / 粘贴移动等操作
- 命令模式（Esc）
  - 按`Esc`退出编辑，进入命令模式
  - 可以**操作cell单元本身**进行剪切 / 复制 / 粘贴 / 移动等操作

快捷键操作

- 两种模式通用快捷键
  - `Shift+Enter`，**执行本单元代码，并跳转到下一单元**
  - `Ctrl+Enter`，**执行本单元代码，留在本单元**
- **命令模式**：按ESC进入
  - Y： 在命令模式下转入代码状态
  - M： 在命令模式下切换到 Markdown
  - R： 普通文本，运行不会输出结果

    - L：为当前cell加上行号
    - A：在该单元格的上方插入新单元格
    - B：在该单元格的下方插入新单元格
    - X：剪切选中的单元
    - C：复制选中的单元
    - V：粘贴到下方单元
    - DD：删除选中的单元（敲两个D）
- 其他(了解)
    - `双击D`：删除当前cell
    - `Z`，回退
    - 快速跳转到首个cell，`Crtl+Home`
    - 快速跳转到最后一个cell，`Crtl+End` -->
- **编辑模式**：按Enter进入
    - 补全代码：变量、方法后跟`Tab键`
    - 为一行或多行代码添加/取消注释：`Ctrl+/`（Mac:CMD+/）
- 其他(了解)：
    - 多光标操作：`Ctrl键点击鼠标`（Mac:CMD+点击鼠标）
    - 回退：`Ctrl+Z`（Mac:CMD+Z）
    - 重做：`Ctrl+Y`（Mac:CMD+Y）

鼠标操作



markdown演示

掌握标题和缩进即可

其他操作

> 函数名 + ? 查看源码

```
In [7]: print?
```

```
Docstring:
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file:  a file-like object (stream); defaults to the current sys.stdout.
sep:   string inserted between values, default a space.
end:   string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
Type:      builtin_function_or_method
```

> Tab 自动补全：如果敲代码没有自动补全，可以用**Tab**查看代码提示

三、小结及注意事项

- 是什么
    - 是一个 ipython 的 web 加强版
- 为什么要使用jupyter
    - 用于数据探索过程
- 怎么用
    - 1.通过 jupyter notebook 就可以使用
    - 2.保存文件是.ipynb
    - 3.每个内容,都对应的是一个cell
- 快捷键
    - Shift+Enter，执行本单元代码，并跳转到下一单元
    - Ctrl+Enter，执行本单元代码，留在本单元

注意事项

`jupyter notebook` 每一个 `cell` 运行完后都会把这个 `cell` 中的变量保存到内存中，如果在一个 `cell` 中修改了之前的变量，再此运行这个 `cell` 的时候可能会导致一些问题产生。比如以下代码：

```
# 第一个cell中的代码
a = 10
b = 20

# 第二个cell中的代码
c = a/b
b = 0
```

因为第二个`cell`修改了`b`变量，此时在整个环境中`b`都是等于0的，所以以后再运行这个`cell`的时候，`a/b`这个就会出问题了。这时候可以使用`Kernel->Restart&Run All`来重新运行整个项目。

附录：Jupyter 常用快捷键

命令模式（按Esc键）：

1. Enter：转入编辑模式
2. Shift-Enter：运行本单元，选中下个单元
3. Ctrl-Enter：运行本单元
4. Alt-Enter：运行本单元，在其下插入新单元
5. Y：单元转入代码状态
6. M：单元转入markdown状态
7. R：单元转入raw状态
8. 1：设定 1 级标题
9. 2：设定 2 级标题
10. 3：设定 3 级标题
11. 4：设定 4 级标题
12. 5：设定 5 级标题
13. 6：设定 6 级标题
14. Up：选中上方单元
15. K：选中上方单元
16. Down：选中下方单元
17. J：选中下方单元
18. Shift-K：扩大选中上方单元
19. Shift-J：扩大选中下方单元
20. A：在上方插入新单元
21. B：在下方插入新单元
22. X：剪切选中的单元
23. C：复制选中的单元
24. Shift-V：粘贴到上方单元
25. V：粘贴到下方单元
26. Z：恢复删除的最后一个单元
27. D,D：删除选中的单元
28. Shift-M：合并选中的单元
29. Ctrl-S：文件存盘
30. S：文件存盘
31. L：转换行号
32. O：转换输出
33. Shift-O：转换输出滚动
34. Esc：关闭页面
35. Q：关闭页面

36. H：显示快捷键帮助
37. I,I：中断Notebook内核
38. 0,0：重启Notebook内核
39. Shift：忽略
40. Shift-Space：向上滚动
41. Space：向下滚动

编辑模式：

1. Tab : 代码补全或缩进
2. Shift-Tab : 提示
3. Ctrl-] : 缩进
4. Ctrl-[: 解除缩进
5. Ctrl-A : 全选
6. Ctrl-Z : 复原
7. Ctrl-Shift-Z : 再做
8. Ctrl-Y : 再做
9. Ctrl-Home : 跳到单元开头
10. Ctrl-Up : 跳到单元开头
11. Ctrl-End : 跳到单元末尾
12. Ctrl-Down : 跳到单元末尾
13. Ctrl-Left : 跳到左边一个字首
14. Ctrl-Right : 跳到右边一个字首
15. Ctrl-Backspace : 删除前面一个字
16. Ctrl-Delete : 删除后面一个字
17. Esc : 进入命令模式
18. Ctrl-M : 进入命令模式
19. Shift-Enter : 运行本单元，选中下一单元
20. Ctrl-Enter : 运行本单元
21. Alt-Enter : 运行本单元，在下面插入一单元
22. Ctrl-Shift-- : 分割单元
23. Ctrl-Shift-Subtract : 分割单元
24. Ctrl-S : 文件存盘
25. Shift : 忽略
26. Up : 光标上移或转入上一单元
27. Down :光标下移或转入下一单元

附录：conda基本使用

`conda` 伴随着 `Anaconda` 安装而自动安装的。 `conda` 可以跟 `virtualenv` 一样管理不同的环境，也可以跟 `pip` 一样管理某个环境下的包。以下来看看两个功能的用法。

环境管理：

`conda` 能跟 `virtualenv` 一样管理不同的 `Python` 环境，不同的环境之间是互相隔离，互不影响的。为什么需要创建不同的环境呢？原因是有时候项目比较多，但是项目依赖的包不一样，比如 `A` 项目用的是 `Python2` 开发的，而 `B` 项目用的是 `Python3` 开发的，那么我们在同一台电脑上就需要两套不同的环境来支撑他们运行了。创建环境的基本命令如下：

```
# conda create --name [环境名称] 比如以下：
conda create --name da-env
```

这样将创建一个叫做 `da-env` 的环境，这个环境的 `python` 解释器根据 `anaconda` 来，如果 `anaconda` 为 `3.7`，那么将默认使用 `3.7` 的环境，如果 `anaconda` 内置的是 `2.7`，那么将默认使用 `2.7` 的环境。然后你就可以使用 `conda install numpy` 的方式来安装包了，并且这样安装进来的包，只会安装在当前环境中。有的同学可能有想问，如果想要装一个 `Python2.7` 的环境， `anaconda` 中没有内置 `Python2.7`，那么该怎么实现呢？。实际上，我们只需要在安装的时候指定 `python` 的版本，如果这个版本现在不存在，那么 `anaconda` 会自动的给我们下载。所以安装 `Python2.7` 的环境，使用以下代码即可实现：

```
conda create --name xxx python=2.7
```

以下再列出 `conda` 管理环境的其他命令：

1. 创建的时候指定需要安装的包：

   ```
   conda create --name xxx numpy pandas
   ```

2. 创建的时候既需要指定包，也需要指定python环境：

   ```
   conda create --name xxx python=3.6 numpy pandas
   ```

3. 进入到某个环境

   ```
   windows: activate xxx
   mac/linux: source activate xxx
   ```

4. 退出环境：

   ```
   deactivate
   ```

5. 列出当前所有的环境：

   ```
   conda env list
   ```

6. 移除某个环境：

   ```
   conda remove --name xxx --all
   ```

7. 环境下的包导出和导入：
   - 导出：`conda env export > environment.yml`。
   - 导入：`conda env create --name xxx -f environment.yml`。

包管理：

`conda` 也可以用来管理包。比如我们创建完一个新的环境后，想要在这个环境中安装包（比如numpy），那么可以通过以下代码来实现：

```
activate xxx
conda install numpy
```

以下再介绍一些包管理常用的命令：

1. 在不进入某个环境下直接给这个环境安装包：

```
conda install [包名] -n [环境名]
```

2. 列出该环境下所有的包：

```
 conda list
```

3. 卸载某个包：

```
 conda remove [包名]
```

4. 设置安装包的源：

```
 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
```