



Python课件一

CREATED BY：罗亚雄

Python课件一

1 简述计算机硬件的发展历史

第一代计算机——电子管计算机（时间1946~1957）

第二代计算机——晶体管计算机（时间1957~1964）

第三代计算机——中小规模集成电路计算机（时间1964~1971）

第四代计算机——大规模和超大规模集成电路计算机（时间1971~至今）

第五代计算机——泛指具有人工智能的计算机（至今~未来）

2 简述计算机软件的发展历史

编程语言的发展

高级语言的发展

B语言与Unix

C语言

C语言和Unix

类C语言起源、历史

3 为什么我们学的是Python

Python有哪些优点
Python有哪些作用
4 Python和anaconda
接下来就给大家演示如何使用Python解释器
方法一：直接在命令行编写Python代码并且运行
方法二：文本编程
Anaconda
启用 Jupyter notebook 的两种方式
1.点击安装时生成的快捷方式（方便，但不推荐使用）
2.在CMD中执行: jupyter notebook。（推荐使用）
最后说明一下
简要说明一下计算机的编码

1 简述计算机硬件的发展历史

第一代计算机——电子管计算机（时间1946~1957）

无论如何，一项技术的突破必然伴随着其他行业的突破，简而言之，电子计算机的出现，前提必须有电子技术的进步，否则一切都是空谈！

时间	事件	
1906年	美国的Lee De Forest 发明了电子管。在这之前造出数字电子计算机是不可能的。这为电子计算机的发展奠定了基础	
1924年2月	一个具有划时代意义的公司成立，IBM	
1935年	IBM推出IBM 601机。这是一台能在一秒钟算出乘法的穿孔卡片计算机。这台机器无论在自然科学还是在商业意义上都具有重要的地位。大约造了1500台。	
1937年	英国剑桥大学的Alan M. Turing (1912-1954)出版了他的论文，并提出了被后人称之为"图灵机"的数学模型。	
1937年	美国贝尔试验室的George Stibitz展示了用继电器表示二进制的装置。尽管仅仅是个展示品，但却是世界上第一台二进制电子计算机。	
1941年	Atanasoff和学生Berry完成了能解线性代数方程的计算机，取名叫"ABC"（Atanasoff-Berry Computer），用电容作存储器，用穿孔卡片作辅助存储器，那些孔实际上是"烧"上的。时钟频率是60HZ，完成一次加法运算用时一秒。这就是ABC计算机。	
1946	美国宾夕法尼亚大学，第一台通用电子计算机ENIAC (Electronic Numerical Integrator 和 Computer)诞生。总工程师埃克特在当时年仅25岁。	

这时的计算机的基本线路是采用电子管结构，程序从人工手编的 机器指令程序（0 1），过渡到符号语言（汇编），电子管计算机是计算工具革命性发展的开始，它所采用的进位制与程序存贮等基本技术思想，奠定了现代电子计算机技术基础。以冯·诺依曼为代表。

第二代计算机——晶体管计算机（时间1957~1964）

电子管时代的计算机尽管已经步入了现代计算机的范畴，但其体积之大、能耗之高、故障之多、价格之贵大大制约了它的普及应用。直到晶体管被发明出来，电子计算机才找到了腾飞的起点，一发而不可收.....

20世纪50年代中期，晶体管的出现使计算机生产技术得到了根本性的发展，由晶体管代替电子管作为计算机的基础器件，用磁芯或磁鼓作存储器，在整体性能上，比第一代计算机有了很大的提高。

第三代计算机——中小规模集成电路计算机（时间1964~1971）

20世纪60年代中期，计算机发展历程随着半导体工艺的发展，成功制造了集成电路。中小规模集成电路成为计算机的主要部件，主存储器也渐渐过渡到半导体存储器，使计算机的体积更小，大大降低了计算机计算时的功耗，由于减少了焊点和接插件，进一步提高了计算机的可靠性。

第四代计算机——大规模和超大规模集成电路计算机（时间1971~至今）

随着大规模集成电路的成功制作并用于计算机硬件生产过程，计算机的体积进一步缩小，性能进一步提高。集成更高的大容量半导体存储器作为内存储器，发展了并行技术和多机系统，出现了精简指令集计算机（RISC），软件系统工程化、理论化，程序设计自动化。微型计算机在社会上的应用范围进一步扩大，几乎所有领域都能看到计算机的“身影”。

第五代计算机——泛指具有人工智能的计算机（至今~未来）

目前还没有明确地定义

2 简述计算机软件的发展历史

编程语言的发展

计算机软件系统的发展，也伴随着编程语言的发展。计算机程序设计语言的发展，经历了从**机器语言**、**汇编语言**到**高级语言**的历程。

机器语言：简单点说，机器本身也只认识0和1，电路无非就只有通和断两种状态，对应的二进制就是二进制的1和1。

汇编语言：汇编语言只是把一些特殊的二进制用特殊的符号表示，例如，机器要传送一个数据，假设“传送”这个指令对应的机器码是000101，则人们把000101用一个特殊符号，比如mov来表示，当人们要用这个指令时用mov就行，但是mov的本质还是000101，没有脱离硬件的范围，有可能这个指令不能在其他机器上用。

高级语言：高级语言完全脱离了硬件范畴，所有的语法更贴近人类的自然语言，人们只需要清楚高级语言的语法，写出程序就行了，剩下的交给编译器或者解释器去编译或者解释成机器语言就行了，看，这样就完全脱离了硬件的范畴，大大提高了程序的开发效率。接下来我们就来看看高级语言的发展，高级语言非常多，我们主要看看比较经典的几个。

高级语言的发展

B语言与Unix

20世纪60年代，贝尔实验室的研究员Ken Thompson（肯·汤普森）发明了B语言，并使用B编了个游戏 - Space Travel，他想玩自己这个游戏，所以他背着老板找到了台空闲的机器 - PDP-7，但是这台机器没有操作系统，于是Thompson着手为PDP-7开发操作系统，后来这个OS被命名为 - UNIX。

C语言

1971年，Ken Thompson（肯·汤普森）的同事D.M.Ritchie（DM里奇），也很想玩Space Travel，所以加入了Ken Thompson，合作开发UNIX，他的主要工作是改进Thompson的B语言。最终，在1972年这个新语言被称为C，取BCPL的第二个字母，也是B的下一个字母。

C语言和Unix

1973年，C主体完成。Ken Thompson和D.M.Ritchie迫不及待的开始用C语言完全重写了UNIX。此时编程的乐趣已经使他们完全忘记了那个“Space Travel”，一门心思的投入到了UNIX和C语言的开发中。自此，C语言和UNIX相辅相成的发展至今。



Ken Thompson (left) 和D.M.Ritchie (right)

类C语言起源、历史

C++ (C plus plus Programming Language) - 1983

还是贝尔实验室的人，Bjarne Stroustrup（本贾尼·斯特劳斯特卢普）在C语言的基础上推出了C++，它扩充和完善了C语言，特别是在**面向对象编程**方面。一定程度上克服了C语言编写大型程序时的不足。

Python (Python Programming Language) --1991

1989年圣诞节期间，Guido van Rossum 在阿姆斯特丹，Guido van Rossum为了打发圣诞节的无趣，决心开发一个新的脚本解释程序，做为ABC语言的一种继承。之所以选中Python（大蟒蛇的意思）作为该编程语言的名字，是因为他是一个叫Monty Python的喜剧团体的爱好者。第一个Python的版本发布于1991年。

Java (Java Programming Language) - 1995

Sun公司的Patrick Naughton的工作小组研发了Java语言，主要成员是James Gosling（詹姆斯·高斯林）

C# (C Sharp Programming Language) - 2000

Microsoft公司的Anders Hejlsberg（安德斯·海尔斯伯格）发明了C#，他也是Delphi语言之父。

当然现在还有一些新语言，比如2009年Google的go语言，以及麻省理工的julia等。

3 为什么我们学的是Python

Python有哪些优点

1 语法简单 漂亮：我们可以说Python是简约的语言，非常易于读写。在遇到问题时，我们可以把更多的注意力放在问题本身上，而不用花费太多精力在程序语言、语法上。

2 丰富而免费的库：Python社区创造了各种各样的Python库。在他们的帮助下，你可以管理文档，执行单元测试、数据库、web浏览器、电子邮件、密码学、图形用户界面和更多的东西。所有东西包括在标准库，然而，除了它，还有很多其他的库。

3 开源：Python是免费开源的。这意味着我们不用花钱，就可以共享、复制和交换它，这也帮助Python形成了丰富的社区资源，使其更加完善，技术发展更快。

4 Python既支持面向过程，也支持面向对象编程。在面向过程编程中，程序员复用代码，在面向对象编程中，使用基于数据和函数的对象。尽管面向对象的程序语言通常十分复杂，Python却设法保持简洁。

5 Python兼容众多平台，所以开发者不会遇到使用其他语言时常会遇到的困扰。**Linux原装Python。**

Python有哪些作用

Python是什么都能做，但是我们学的是数据分析，我们看看在数据分析领域Python能做什么。

数据采集：以Scrapy 为代表的各类方式的爬虫

数据链接：Python有大量各类数据库的第三方包，方便快速的实现增删改查

数据清洗：Numpy、Pandas，结构化和非结构化的数据清洗及数据规整化的利器

数据分析：Scikit-Learn、Scipy，统计分析，科学计算、建模等

数据可视化：Matplotlib、Seaborn等等大量各类可视化的库

4 Python和anaconda

Python我们说过了是一门编程语言，ok，这个没问题，同时我们也已经说过了计算机只认识0 1 这种机器码，这个也没有问题。那么我写的Python代码到底是通过谁变成了了0 1 这种计算机能识别的机器码呢？

我们平常所说的下载一个Python，其实下载的是一个Python解释器，就是利用它，把我们写的Python代码变成了计算机能识别的东西。

Python解释器的官方网站是：<https://www.python.org/>

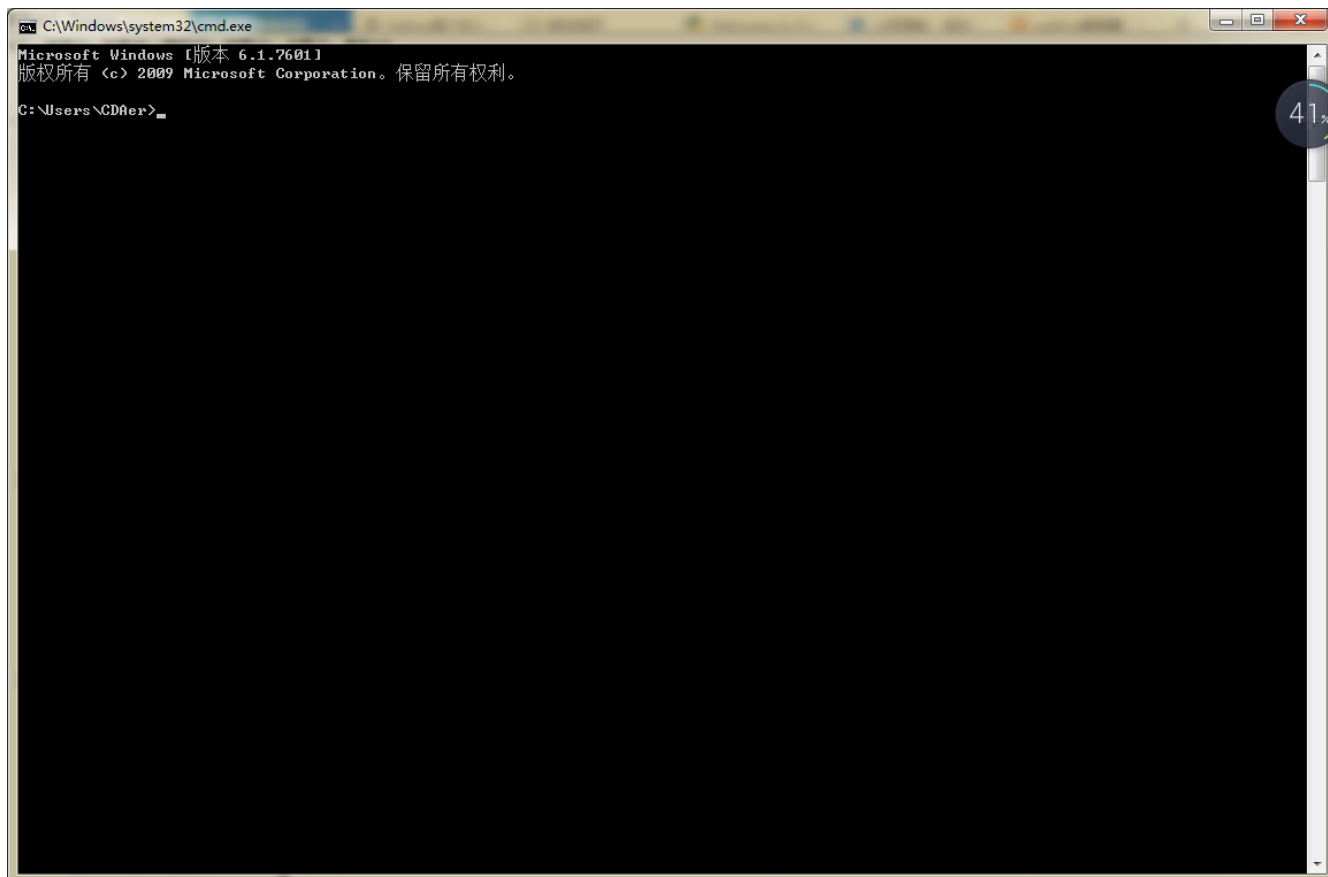
需要注意的是，我们想想这个问题，我们下载下来的Python解释器归根结底也是一个软件，那它又是用什么语言编写的呢？注意下，我们平常说的Python解释器是用c语言编写的，可以称作CPython，原因在于c语言的稳定性和运行速度在高级语言里首屈一指。除此之外还有其他语言编写的。比如：Java语言变得JPython，c#编写的IronPython等等。但是官方，也就是我们用的还是c语言编写的CPython。

接下来就给大家演示如何使用Python解释器

方法一：直接在命令行编写Python代码并且运行

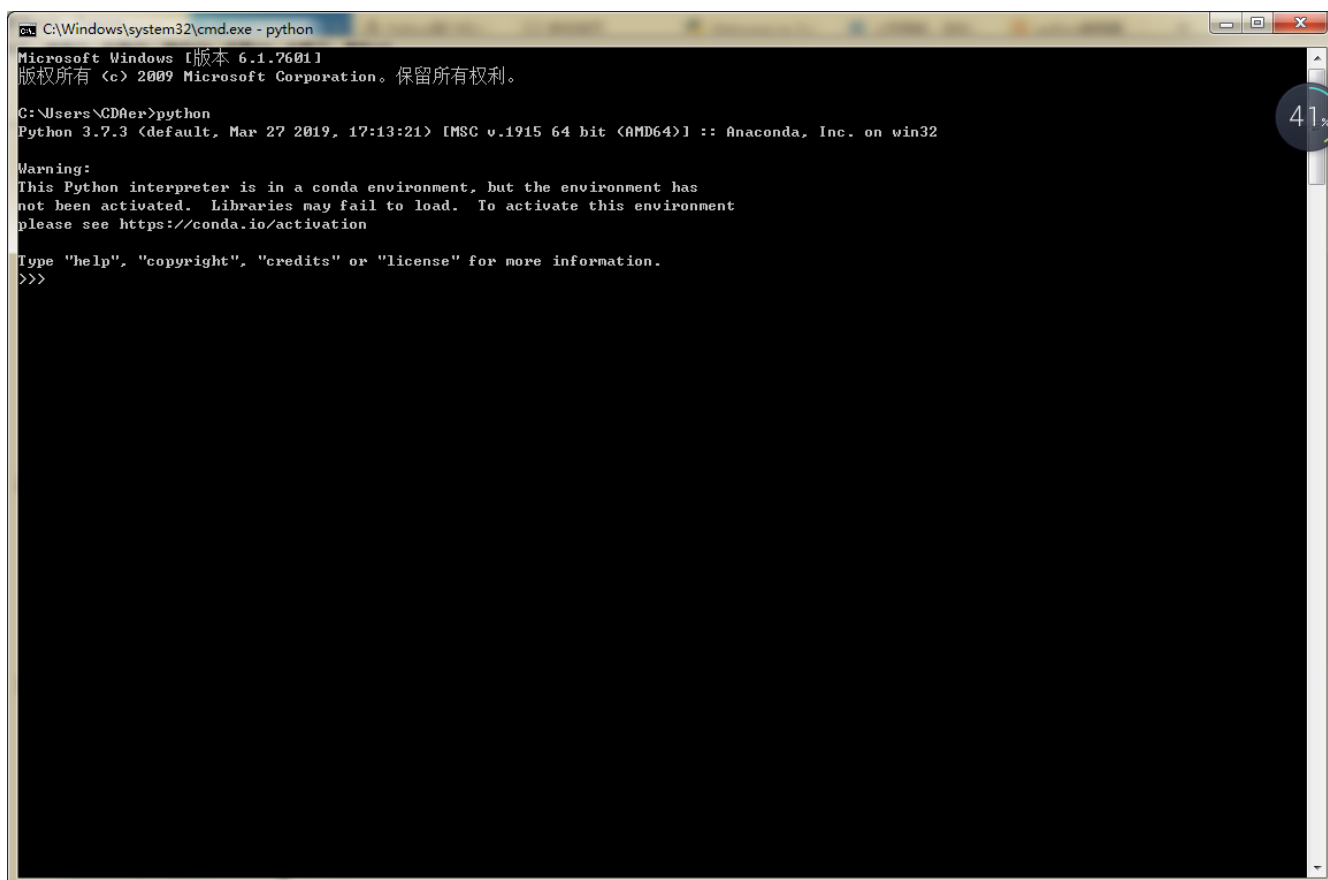
当大家从Python解释器的官方网站下载后，安装并且**配置好环境变量后**，我们可以直接在命令行打开Python环境，进行编程和运行。

win+R---cmd打开命令行窗口



```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。
C:\Users\CD\er>
```

直接往里面输入Python，然后回车（如果说找不到路径，那就是没有配好环境变量），没有问题，如下



```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。
C:\Users\CD\er>python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>>
```

可以看出，我这里的Python是3.7.3，就可以在这个环境下编写Python代码了，比如 输出 hello world

```

C:\Windows\system32\cmd.exe - python
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\CDaer>python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>>

```

也可以直接计算，比如计算2的13次方

```

C:\Windows\system32\cmd.exe - python
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\CDaer>python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>> print(2**13)
8192
>>>

```

也可以使用列表推导式 `print([x**2+1 for x in range(1,101)])`

```

C:\Windows\system32\cmd.exe - python
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\CDaer>python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>> print(2**13)
8192
>>> print([x**2+1 for x in range(1,101)])
[2, 5, 10, 17, 26, 37, 50, 65, 82, 101, 122, 145, 170, 197, 226, 257, 290, 325, 362, 401, 442, 485, 530, 577, 626, 677, 730, 785, 842, 901,
962, 1025, 1090, 1157, 1226, 1297, 1370, 1445, 1522, 1601, 1682, 1765, 1850, 1937, 2026, 2117, 2210, 2305, 2402, 2501, 2602, 2705, 2810, 2917,
3026, 3137, 3250, 3365, 3482, 3601, 3722, 3845, 3970, 4097, 4226, 4357, 4490, 4625, 4762, 4901, 5042, 5185, 5330, 5477, 5626, 5777, 5930,
6085, 6242, 6401, 6562, 6725, 6890, 7057, 7226, 7397, 7570, 7745, 7922, 8101, 8282, 8465, 8650, 8837, 9026, 9217, 9410, 9605, 9802, 10001]
>>>

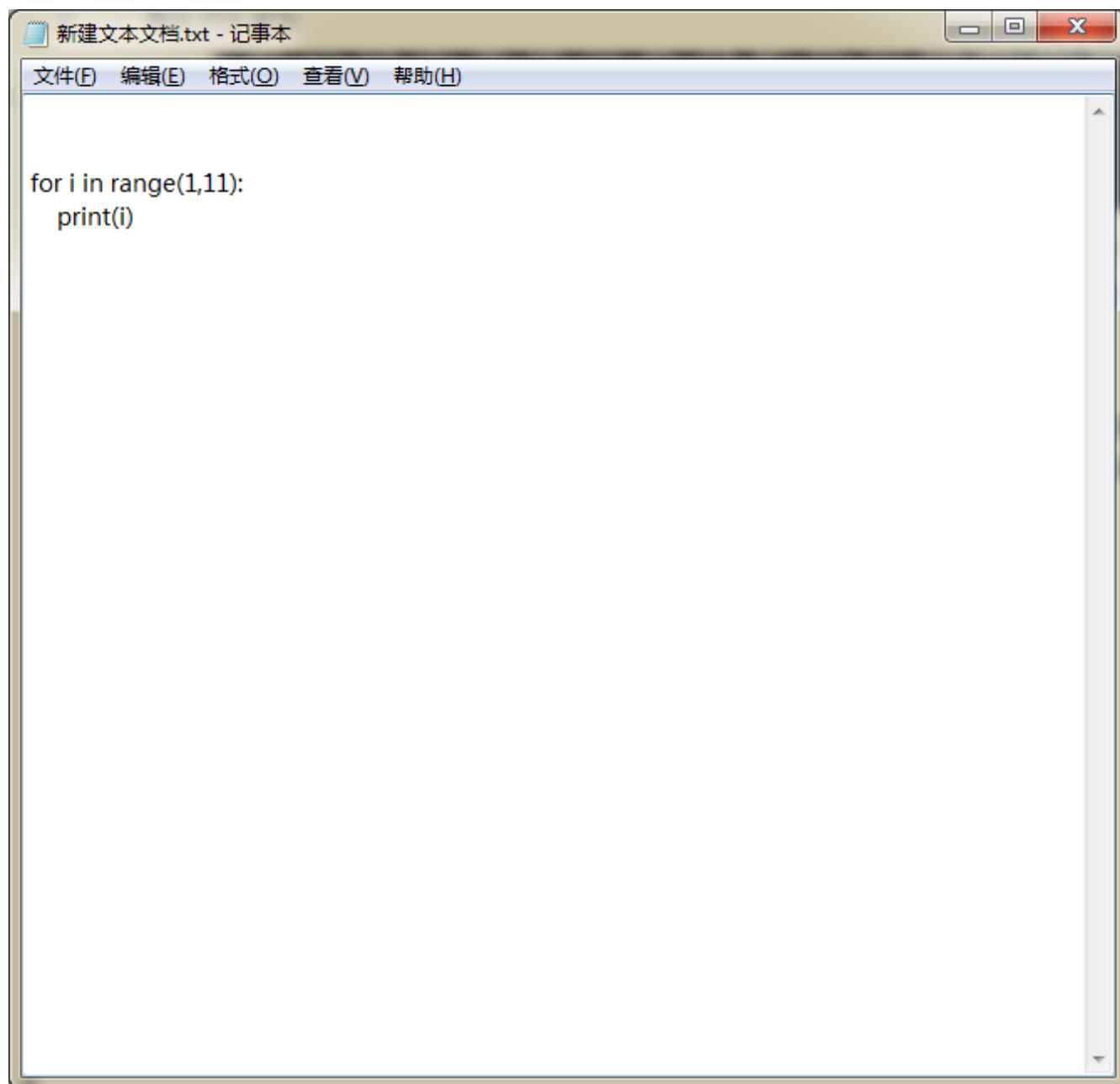
```

总之，我们可以在这个环境里进行Python代码的编写和运行。但是这里有一个大问题，我写的代码怎么保存呢，要是不小心把命令窗口关闭了，那就什么都没有了....于是我们再介绍下文本编程。

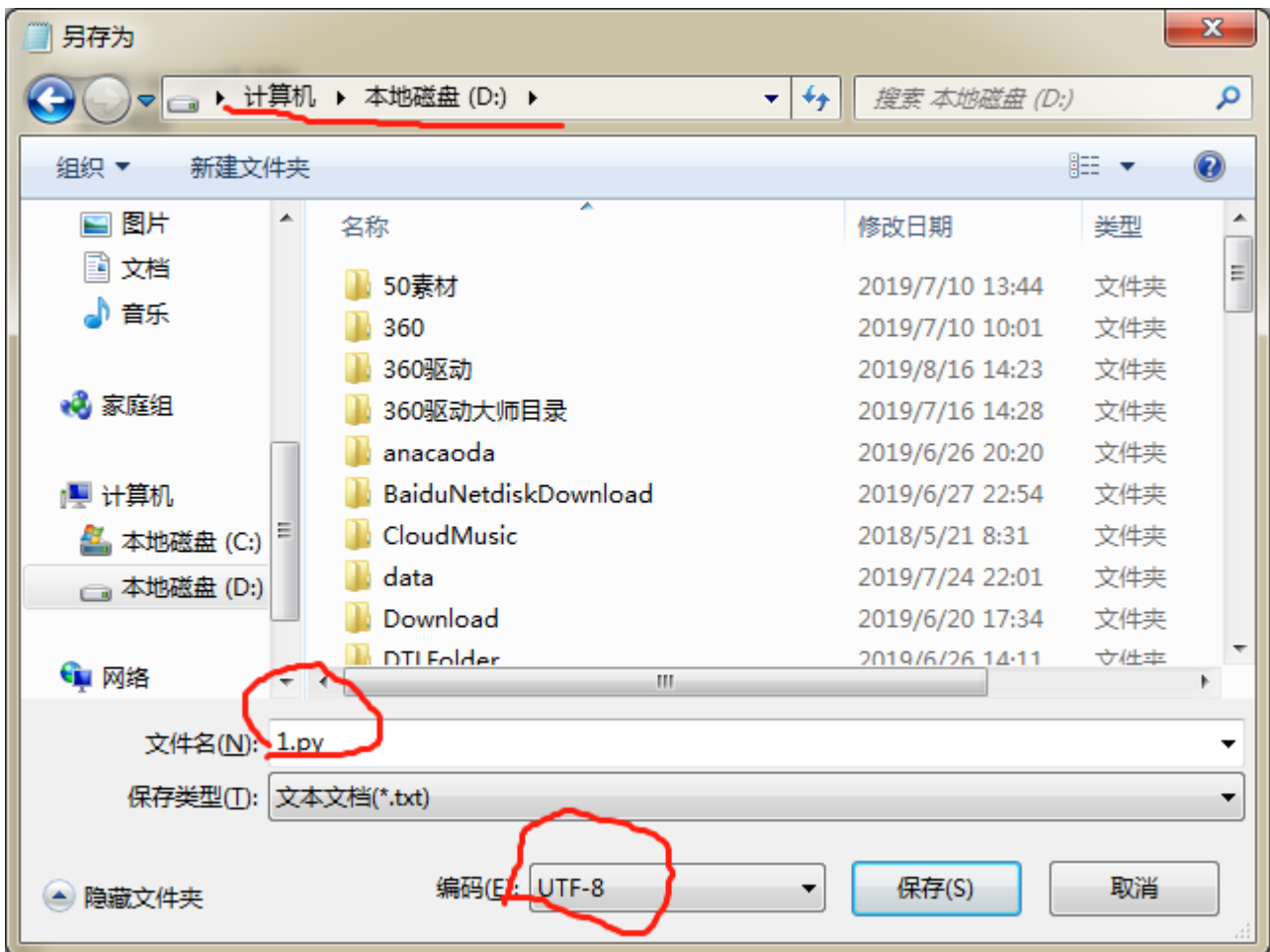
方法二：文本编程

刚才我们说我们在命令行编写的代码无法保存，那么我们可不可以不在文本里编写代码，然后执行呢？答案是可以的。ok，我们新建一个记事本文档(txt),然后往里面写入我们要写的Python代码，比如我们就写一个循环，输出1-10吧。

右键--新建--文本文档

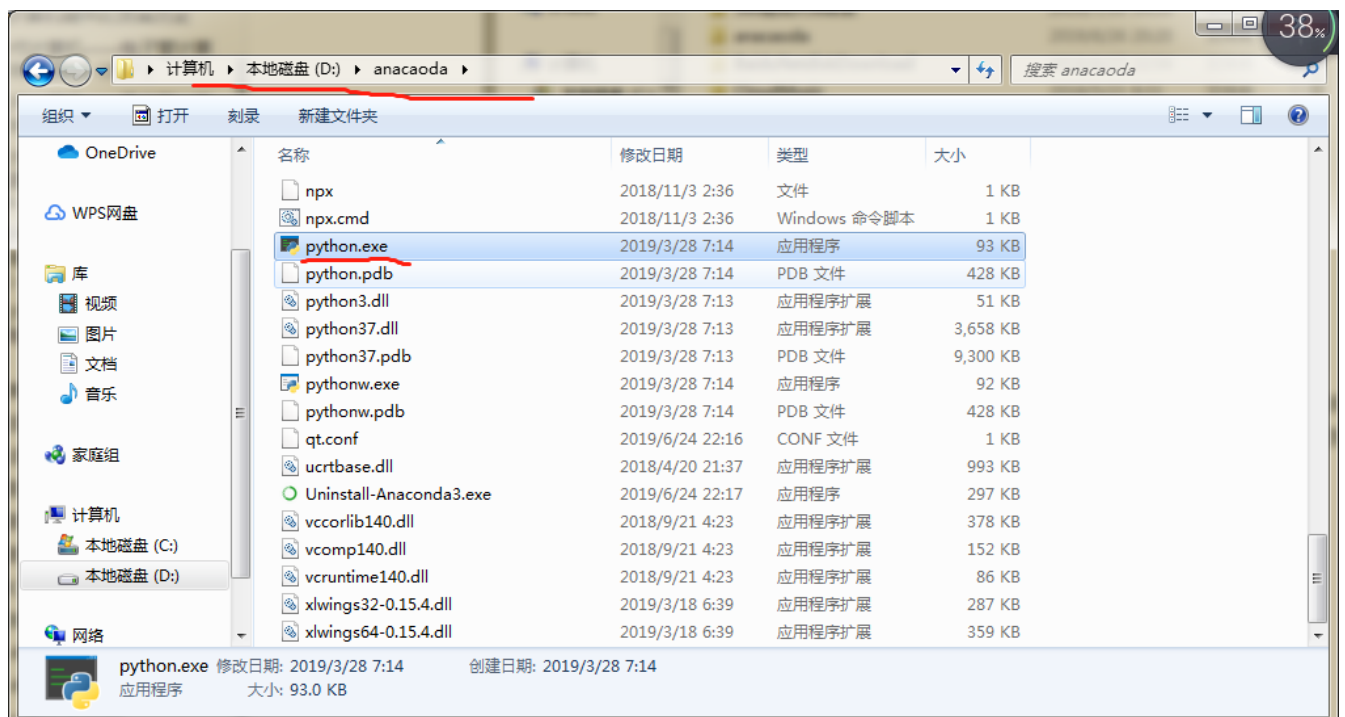


然后文件--另存为---放到一个很容易找到的目录，比如我这里就放到**D盘的根目录下**，命名为**1.py**(这是Python执行文件的文件后缀)，还要重点注意的是**编码改成utf-8**。

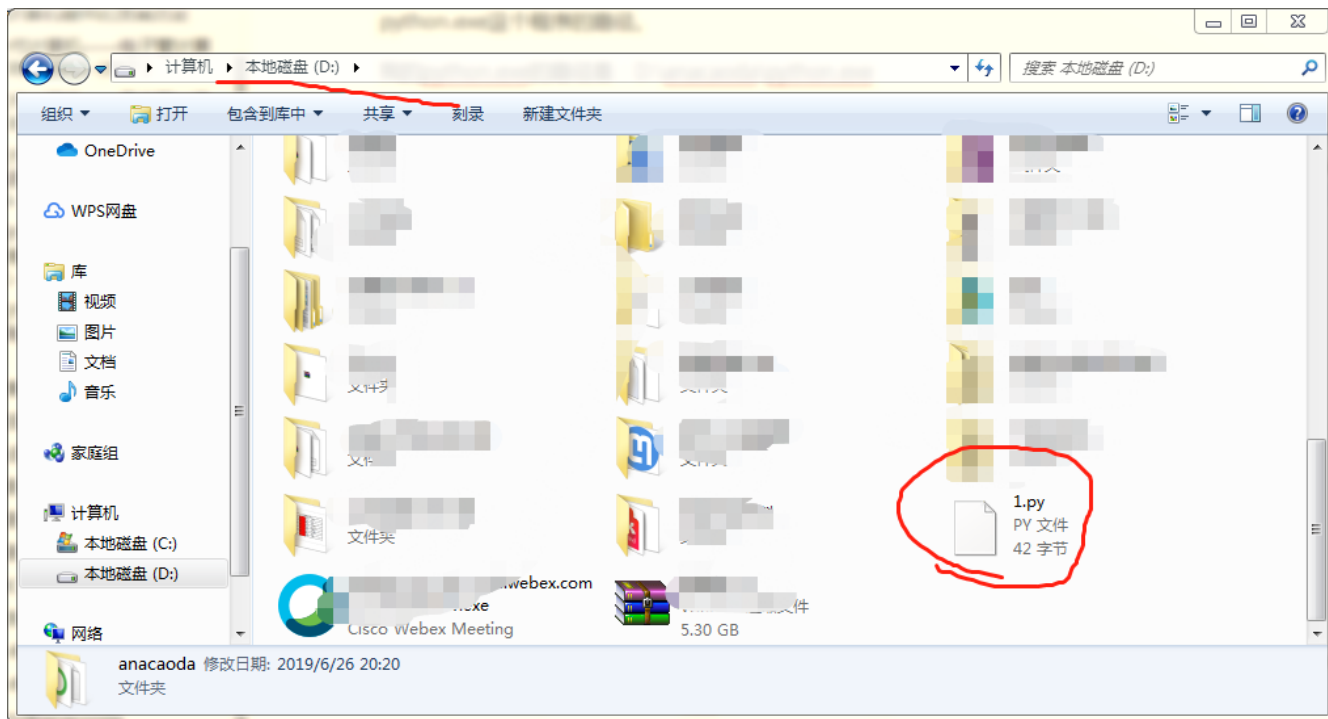


现在在呢，我们自然需要调用我们的Python解释器去执行我们刚才写的1.py这个文件，我这里就直接调用anaconda内置的Python解释器，当然如果你另外安装了其他的Python解释器也可以。那么首先呢的找到python.exe这个程序的路径。

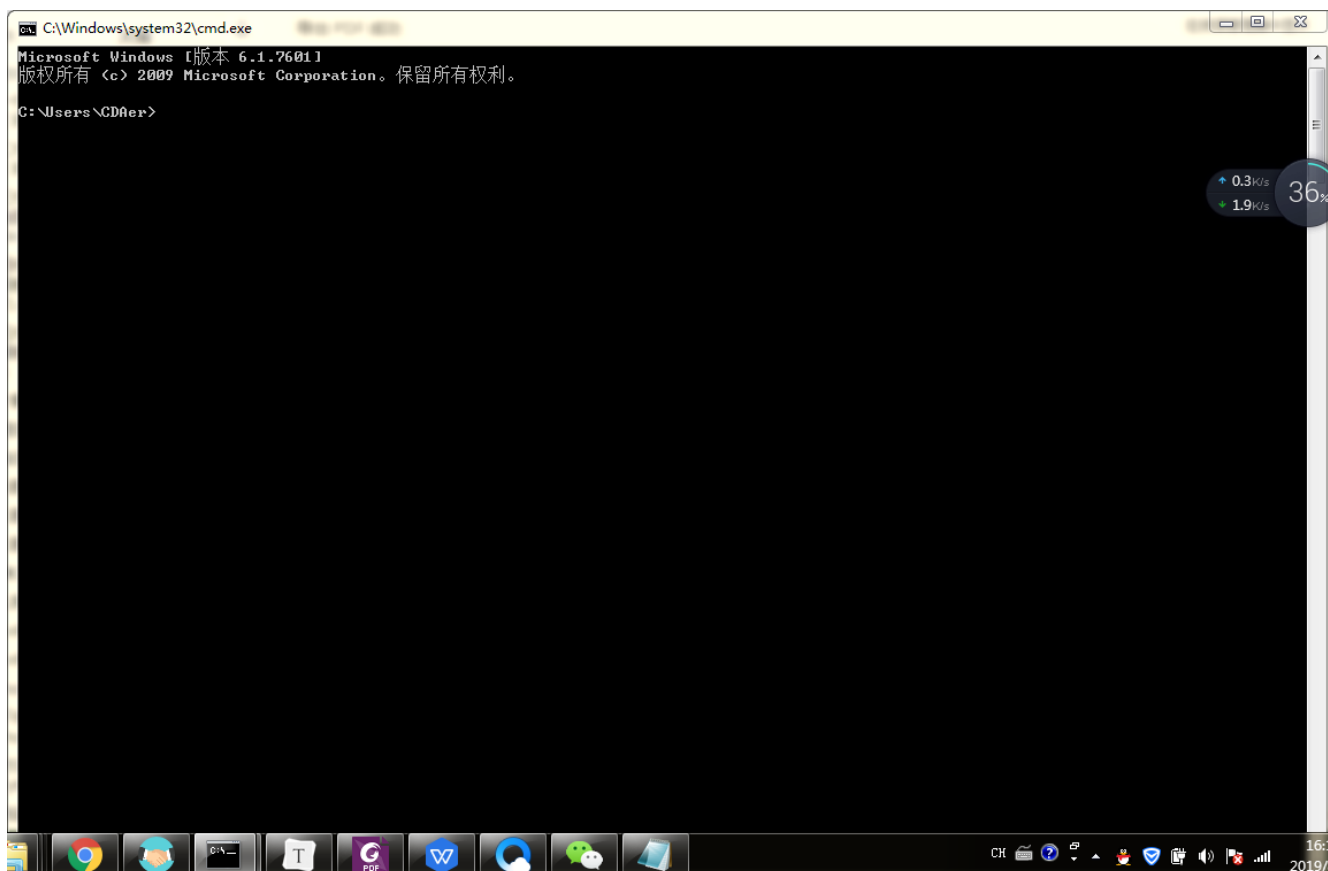
我的python.exe的路径是：D:\anacoda\python.exe （打错了，我之前文件命名写成anacoda....）



然后需要执行的python代码文件1.py的路径是：D:\1.py



那么现在呢，运行命令窗口：win+R---cmd打开命令行窗口



先要找到python.exe,它的路径是：D:\anacoda\python.exe，所以先进入D盘

输入 d: 回车 然后就进入了D盘，如下

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\CDAer>d:

D:\>_
```

然后我们进入anacaoda 输入：cd anacaoda

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\CDAer>d:

D:\>cd anacaoda

D:\anacaoda>
```

然后找到python.exe 后面接上 d:\1.py 然后回车 就执行了1.py这个文件 结果如下

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\CDAer>d:

D:\>cd anacaoda

D:\anacaoda>python.exe d:\1.py
1
2
3
4
5
6
7
8
9
10

D:\anacaoda>
```

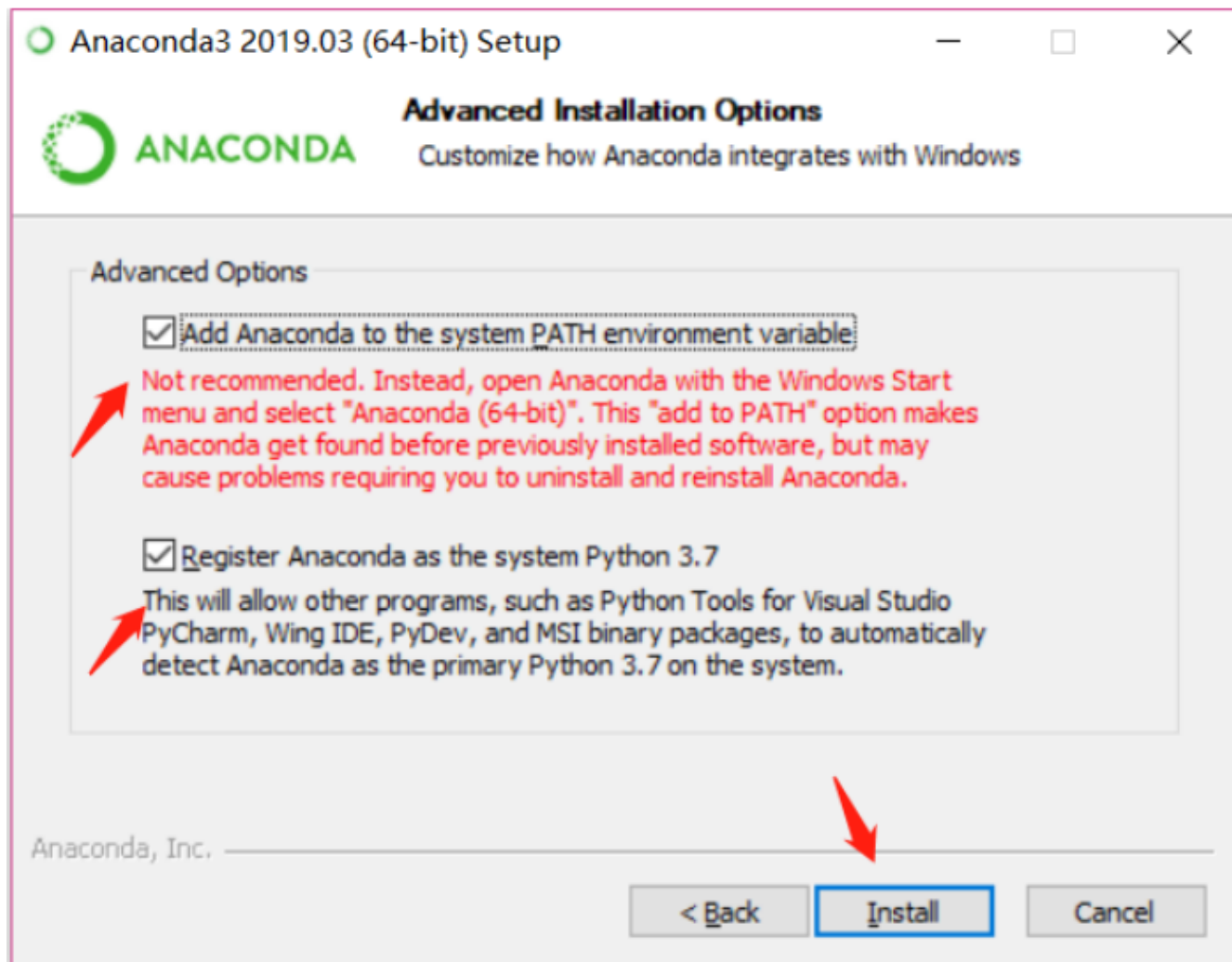
那么，现在我们就完成了在文档里编程，并且执行这个文件，确实比第一种要好，至少能保存我们写的代码了，但是，大家有没有感觉还是很繁琐，要注意编码，要注意文件路径等等，那么接下来呢，给大家介绍一下更加强大的编程工具，我们称之为**集成开发环境（IDE）**。

Anaconda

Anaconda,我相信助教老师已经发了安装包和安装教程，我这里就简单说一下。

Anaconda官方网站：<https://www.anaconda.com/>

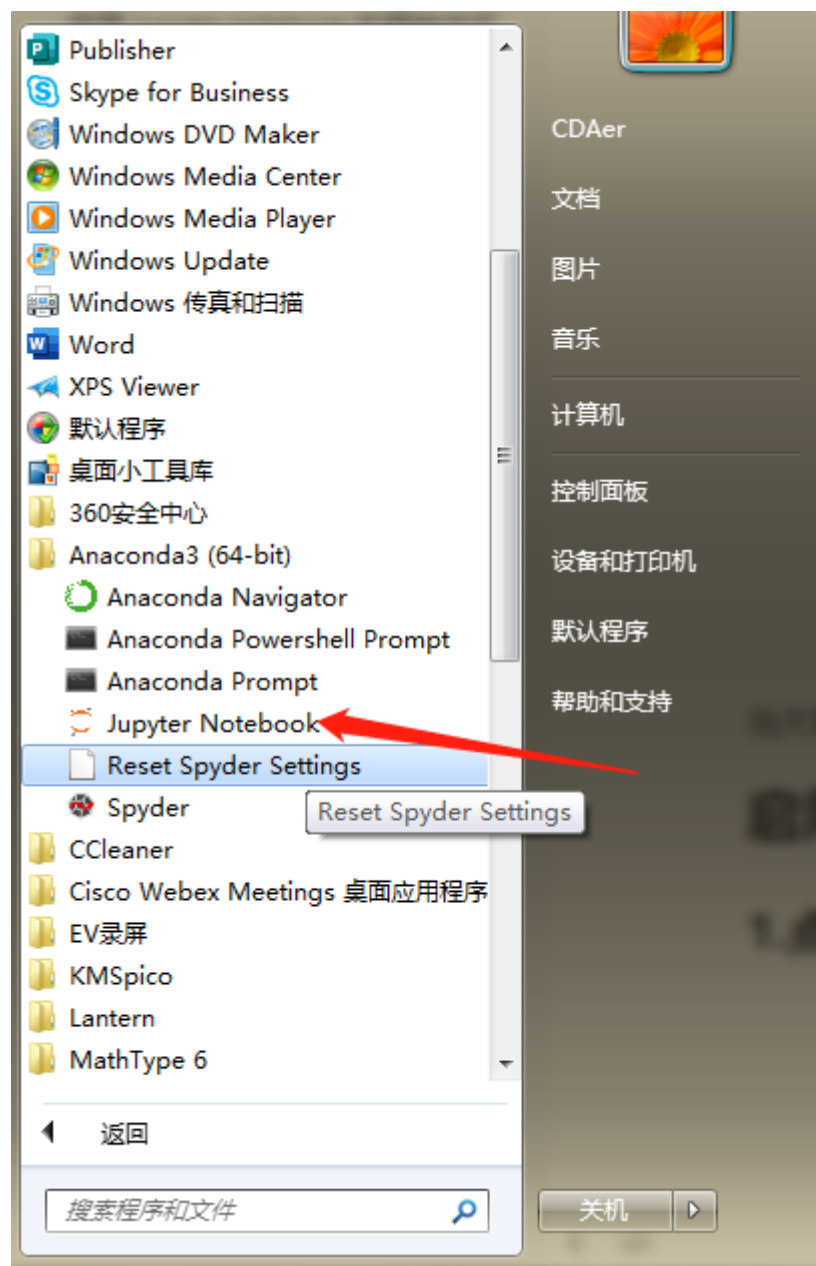
Anaconda非常好安装，相信安装过的同学是有体会的，就是默认-下一步，但是有一步非常重要，就是帮我们配置Python的环境变量。如下图：



当大家安装好了之后呢，我们就要启用Anaconda中的Jupyter notebook

启用 Jupyter notebook 的两种方式

1. 点击安装时生成的快捷方式（方便，但不推荐使用）



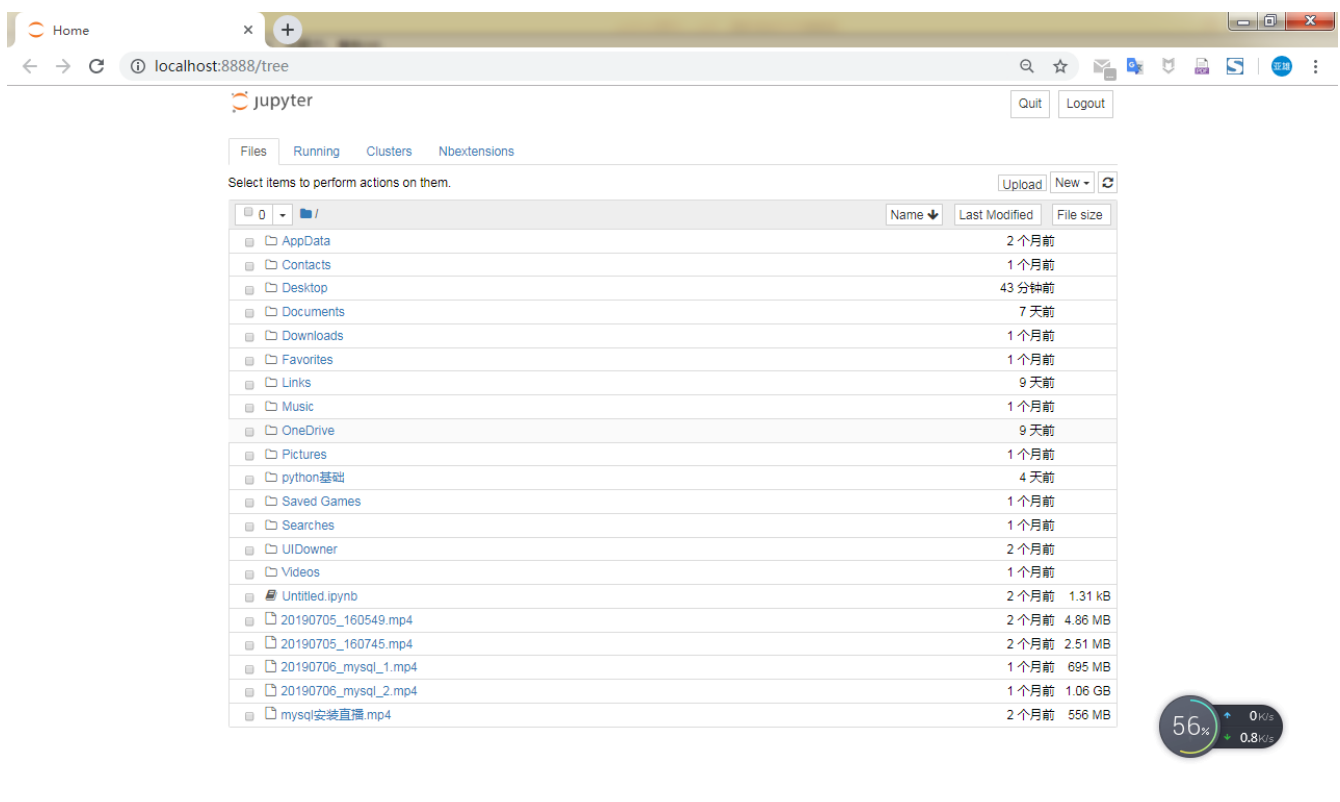
单击 Jupyter notebook,会弹出一个命令行窗口

```
Jupyter Notebook

[I 10:58:39.331 NotebookApp] [jupyter_nbextensions_configurator] enabled 0.4.1
[I 10:58:39.390 NotebookApp] JupyterLab extension loaded from D:\anacoda\lib\site-packages\jupyterlab
[I 10:58:39.391 NotebookApp] JupyterLab application directory is D:\anacoda\share\jupyter\lab
[I 10:58:39.393 NotebookApp] Serving notebooks from local directory: C:\Users\CD\Aer
[I 10:58:39.393 NotebookApp] The Jupyter Notebook is running at:
[I 10:58:39.394 NotebookApp] http://localhost:8888/?token=8145627c2480c6bb26cda6fb6caec5671134ed97314e84b1
[I 10:58:39.394 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 10:58:39.440 NotebookApp]

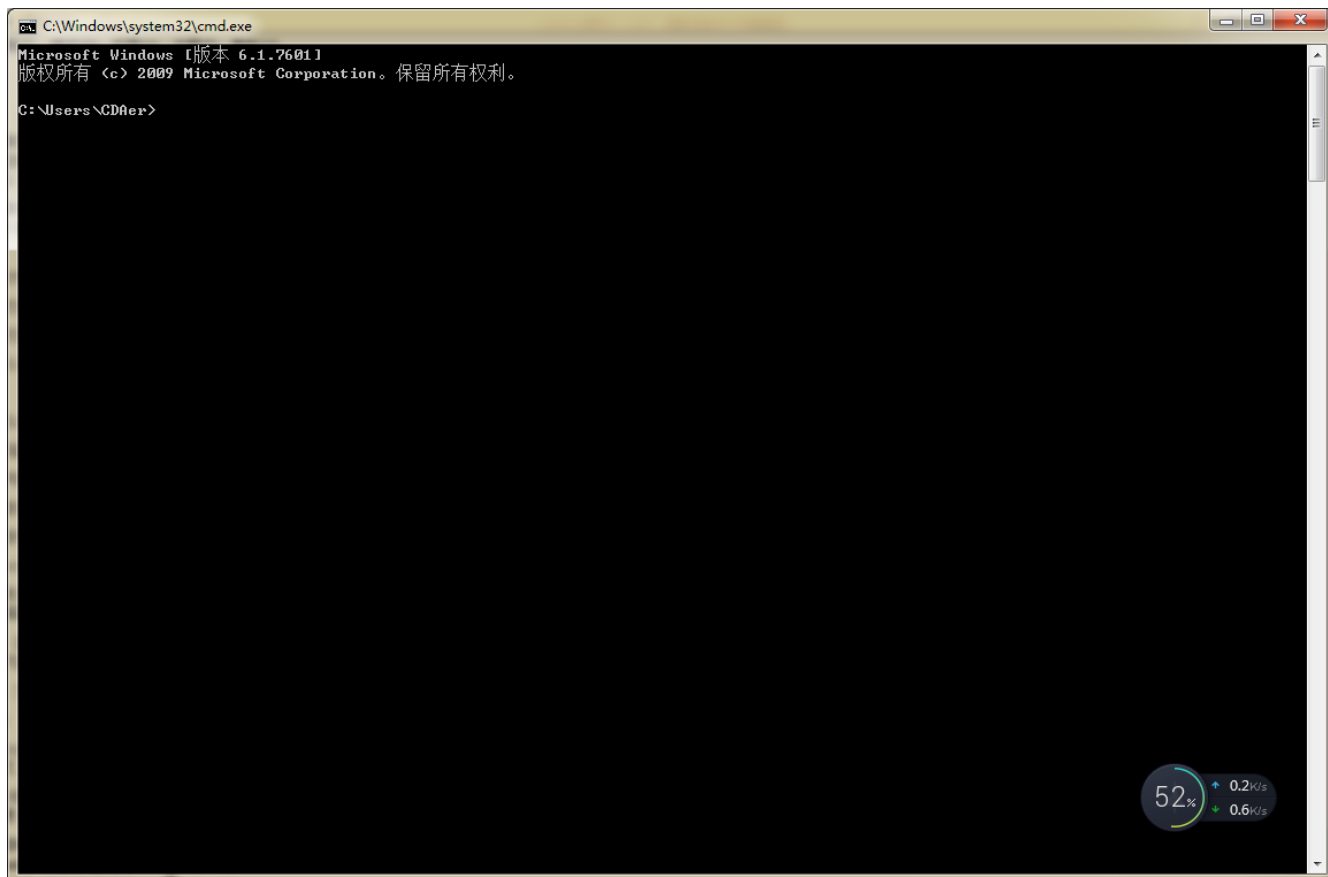
To access the notebook, open this file in a browser:
    file:///C:/Users/CD/Aer/AppData/Roaming/jupyter/runtime/nbserver-10164-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=8145627c2480c6bb26cda6fb6caec5671134ed97314e84b1
```

接着会打开电脑的 默认浏览器（建议下载一个谷歌浏览器，并将其设置为默认浏览器），弹出以下画面

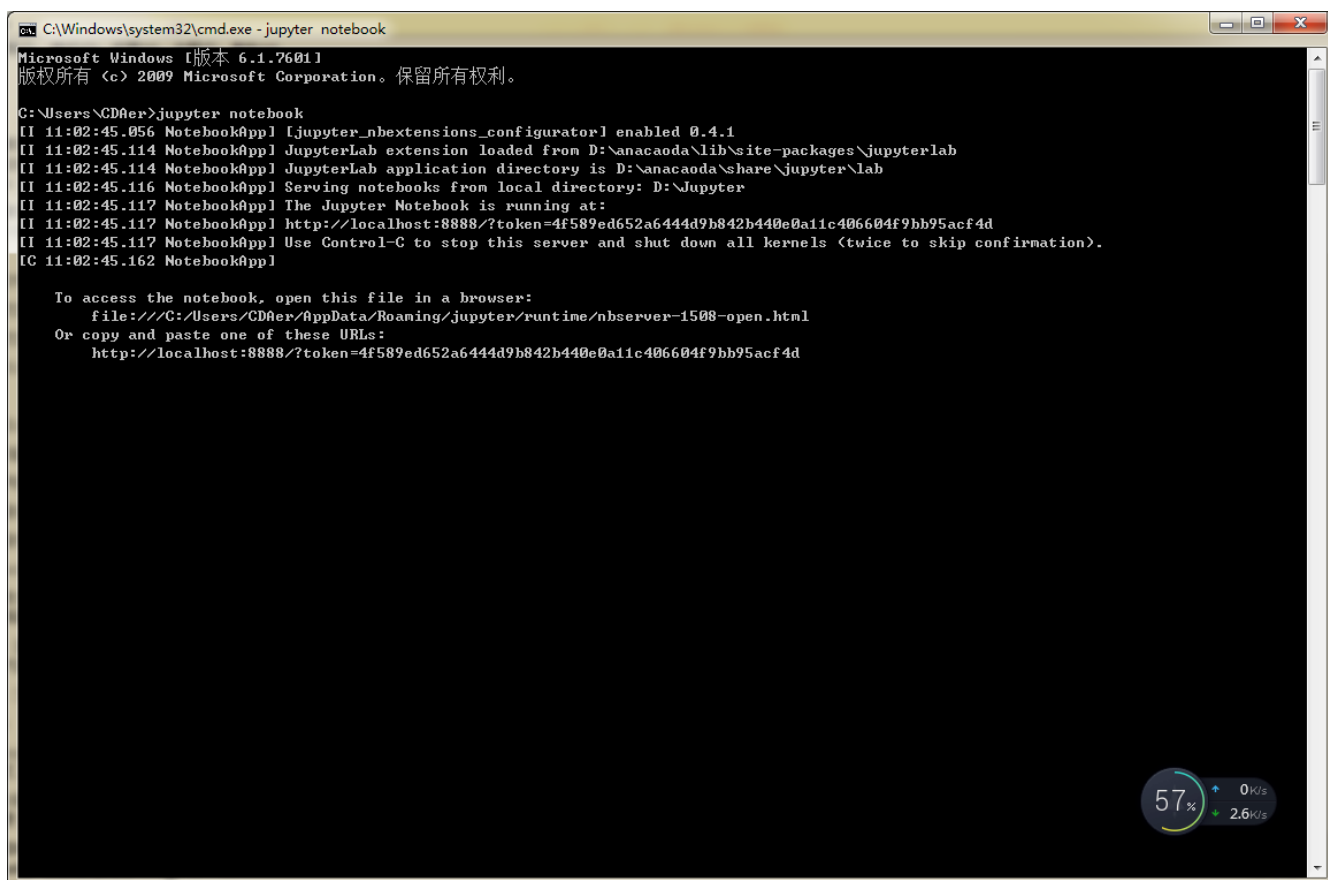


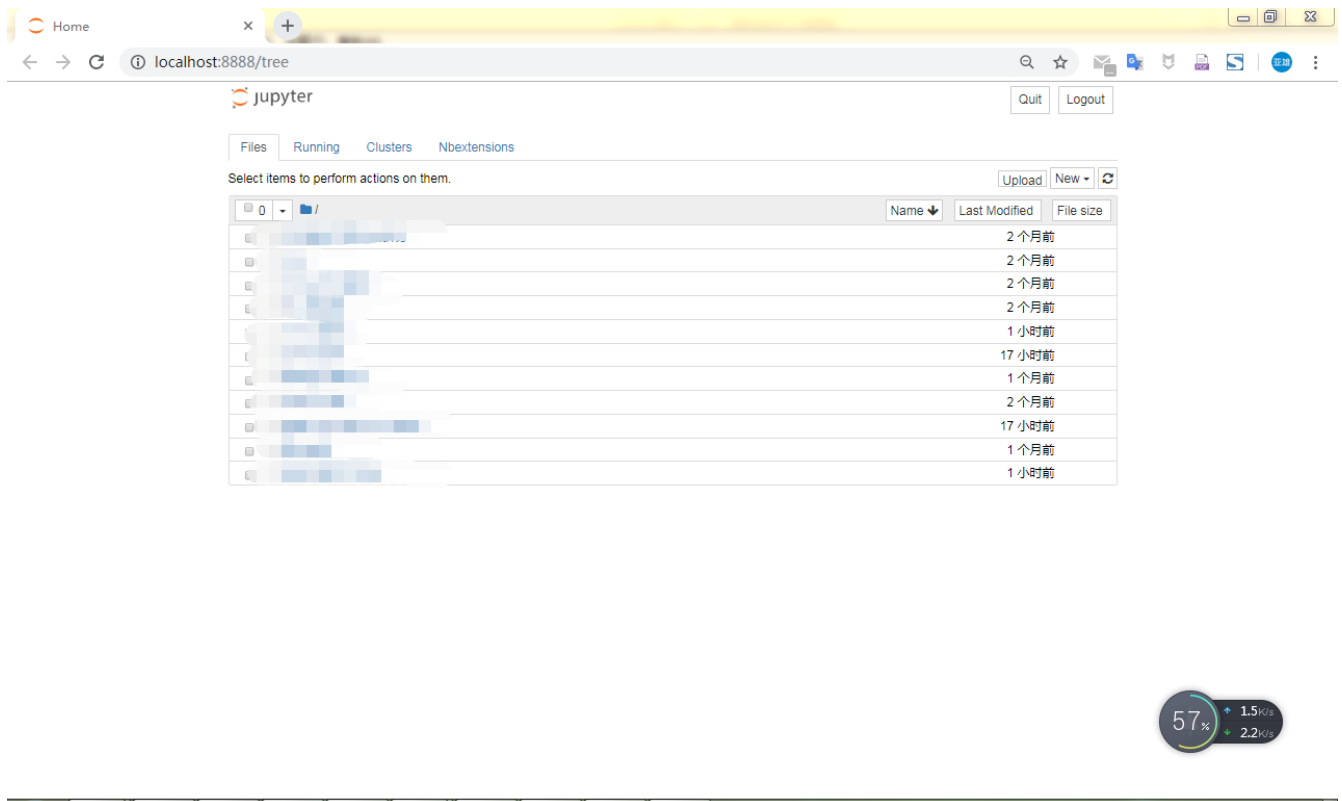
2.在CMD中执行: jupyter notebook。（推荐使用）

win+R,输入cmd,打开 命令行窗口



输入 Jupyter notebook 然后 回车





由于我更这里呢更改过工作路径，内容可能不太一样，但是没有关系，打开这个界面就没问题了。

最后说明一下

python的集成开发环境有很多，但是对于我们数据分析领域而言，我们选择Anaconda。这里额外推荐大家去了解下一个python的集成开发环境pycharm,对于大型程序的编写，非常好用。

我们为什么选择anaconda作为我们数据分析的利器？主要原因就是以下几点：

- 1 anaconda内置python解释器，我们可以直接在anaconda里面的进行python代码的编写和运行，非常方便。
- 2 不仅能够进行python代码的编写和执行，还能够把每次代码执行结果显示并保留下来，包括图片。
- 3 自带100多个常用的python的包，方便我们处理包与包之间的关系。
- 4 可以利用markdown语法进行额外说明。

当然，其优点远远不止这几点。

简要说明一下计算机的编码

前段时间大家在学习Excel老是碰到什么**ANSI**, 学习MySQL的时候老是碰到什么**utf-8**，不知道这些是什么就简要说下。计算机只是一台机器，底层的硬件部分只认识有没有电，什么意思，刚开始的计算机是电子管计算机，通过电路的通断来表示不同的信息，通电用1表示，断开用0表示，就是我们熟悉的二进制。也就是说，计算机本质上只认识0 1这种机器码。那么如何让计算机认识'a',认识'A',这就要涉及编码。我们可以用一串0 1的组合来表示某个特定的字符。比如我们可以用0110 0001来表示'a',用0100 0001表示'A'，这样计算机就能识别'a',或者'A'。于是呢，诞生了经典的**ASCII码**。

ASCII (American Standard Code for Information Interchange，美国标准信息交换代码) 是基于拉丁字母的一套电脑编码系统，主要用于显示现代英语和其他西欧语言，其最多只能用 8 位来表示（一个字节），即： $2^8 = 256$ ，所以，**ASCII**码最多只能表示 256 个符号。

那中文怎么办？日文怎么办？于是有的国家自己推出了自己的编码方式，像我国的**GB2313 GBK**等等，但是每个国家都有自己的编码，如何进行信息交换呢？用**GB2313**进行编码的中文在日本自己的计算机编码里显示出来就是乱码，于是为了解决不同国家的信息交换问题，诞生了**unicode**,你可以称之为万国码 国际码等等,几乎包含所有国家的文字的编码，于是如果大家都用**Unicode**，就不会乱码了。

但是呢，**Unicode**的储存方式是用2-4个字节，也就是任何一个字符的编码至少需要2个字节来储存，比如"Python"这个字符串，一共6个字符，那么如果用Unicode编码至少需要12个字节，一看，好像也没有多大，是的，但是如果我们的考虑是网络传输时，英文字符用**ASCII**码就一个字节，而Unicode需要的是至少两个字节，至少多了一倍，对于网络的传输效率来说，太低了。于是就诞生了**utf-8**。

utf-8进行了分类，使用1、2、3、4个字节表示所有字符；优先使用1个字节、无法满足则使增加一个字节，最多4个字节。英文占1个字节、欧洲语系占2个、东亚占3个，其它及特殊字符占4个，这样，相比**Unicode**传输效率上就提高了很多。

当然还有其他编码，这里不过多阐述了，有兴趣的，百度一下即可。

ok ,接下来就给大家讲解Python的语法，同时再讲解anaconda的使用。