



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Python 基础

04/19/2021

为什么选择Python

- 最热门的编程语言
- 适合初学者：语法简单，语句清晰，编程简单
- 应用广泛：数据处理、搭建网站、爬虫等等
- 人工智能

学习资料

廖雪峰Python教程：

<https://www.liaoxuefeng.com/wiki/1016959663602400/>

Python Tutorial for Beginners (For Absolute Beginners):

<https://www.youtube.com/playlist?list=PLS1QulWo1RIaJECMeUT4LFwJ-ghgoSH6n>

Google's Python Class:

<https://developers.google.com/edu/python/introduction>

Python安装

- 推荐使用一下两款软件：

Anaconda：轻松实现python中各种包的管理。

PyCharm：帮助用户在使用Python语言开发时提高其效率。

- 网上有具体安装教程：

- <https://blog.csdn.net/yingduo8217/article/details/107057895/>

- <https://www.cnblogs.com/pejsidney/p/9216470.html>

Anaconda

- Anaconda是一个基于Python的**数据处理和科学计算平台**，它已经内置了许多非常有用的第三方库，装上Anaconda，就相当于把Python和一些如Numpy、Pandas、Matplotlib等**常用的库自动安装好了**，使得安装比常规python安装要容易。如果没有安装Anaconda,而是安装了python，那么还需要pip install一个一个安装各种库，安装起来比较痛苦，还需要考虑兼容性。因而建议直接安装推荐直接使用Anaconda。

Anaconda

1) Anaconda 附带了一大批常用数据科学包，它附带了 conda、Python 和 150 多个科学包及其依赖项。

2) 管理包

Anaconda 是在 conda（一个包管理器和环境管理器）上发展出来的。

在数据分析中，你会用到很多第三方的包，而conda（包管理器）可以很好的帮助你在计算机上**安装、卸载和更新包**。

3) 管理环境

同时安装两个python版本可能会造成许多混乱和错误。这时候 conda就可以帮助你**为不同的项目建立不同的运行环境**。

还有很多项目使用的包版本不同，比如不同的pandas版本，不可能同时安装两个 Numpy 版本，你要做的应该是，为每个 Numpy 版本创建一个环境，然后项目的对应环境中工作。这时候conda就可以帮你做到。

PyCharm

- PyCharm 是一种 Python IDE，其实就是创建快捷方式调用python 解释器运行代码，增加了一些友好的编辑代码的功能。可以帮助程序员节约时间，提高生产效率。IDE(Integrated Development Environment)，译为集成开发环境，其缩写形式IDE同时也代指“电子集成驱动器”。是一种辅助程序开发人员开发软件的应用软件。IDE通常包括编程语言编辑器、自动建立工具、通常还包括调试器。

安装过程：Anaconda

官网：<https://www.anaconda.com/products/individual>

教程：<https://blog.csdn.net/Wonz5130/article/details/82258603>

Anaconda Installers

Windows

Python 3.8

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (397 MB)

MacOS

Python 3.8

64-Bit Graphical Installer (462 MB)

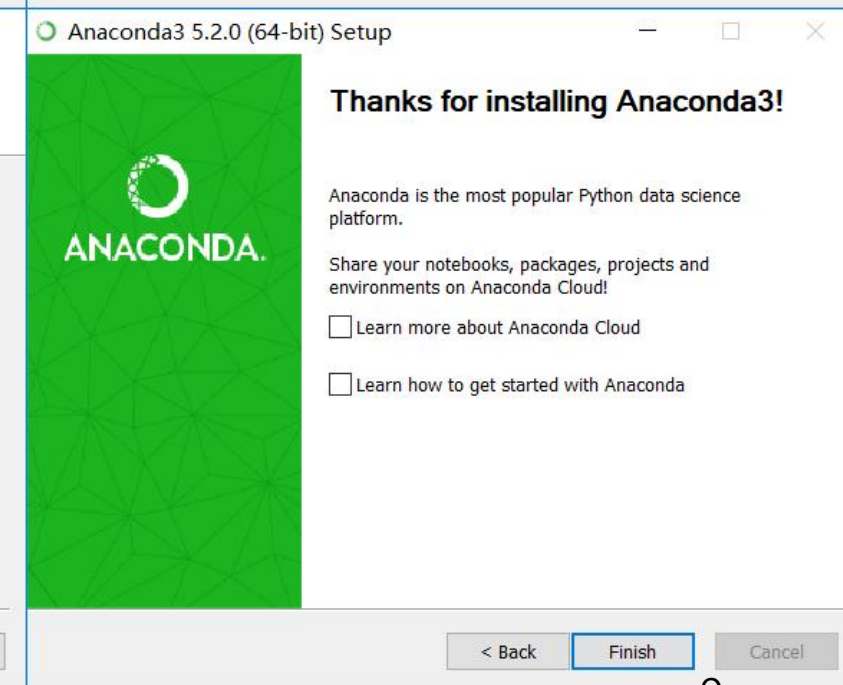
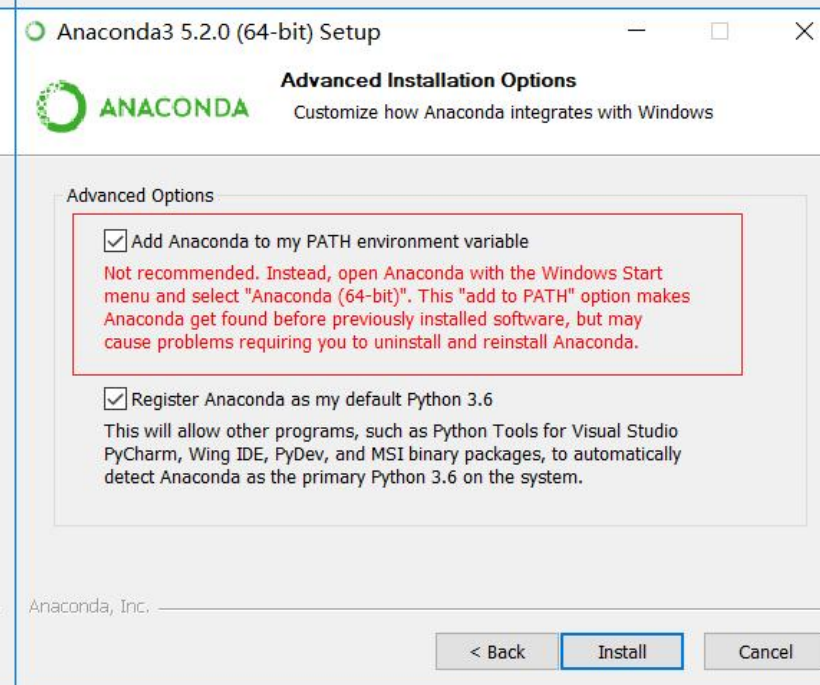
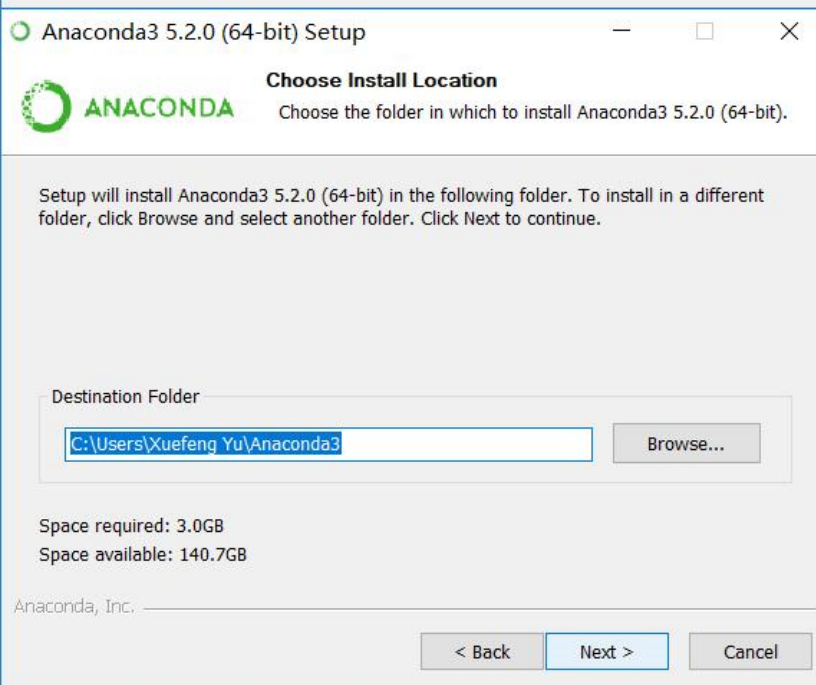
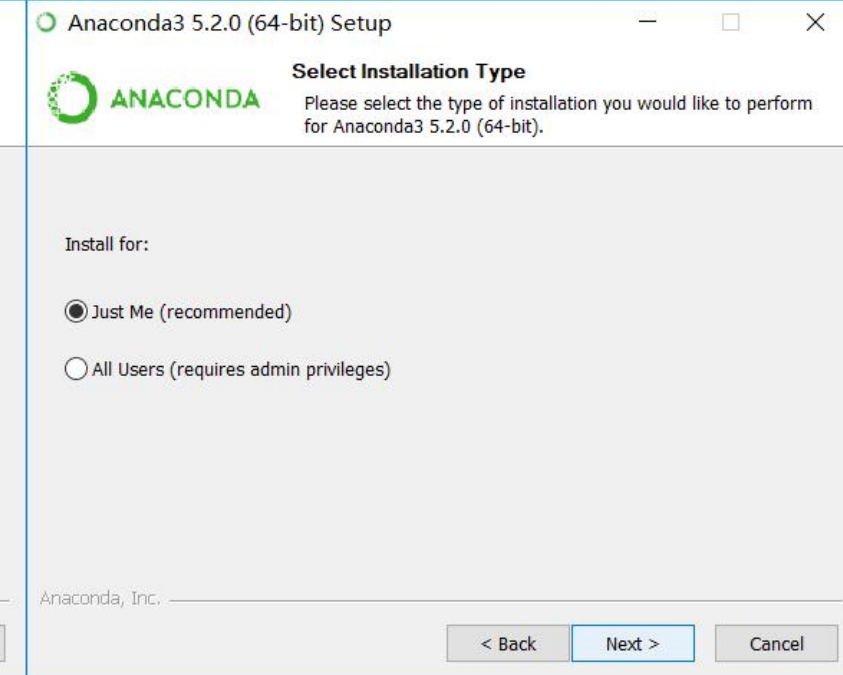
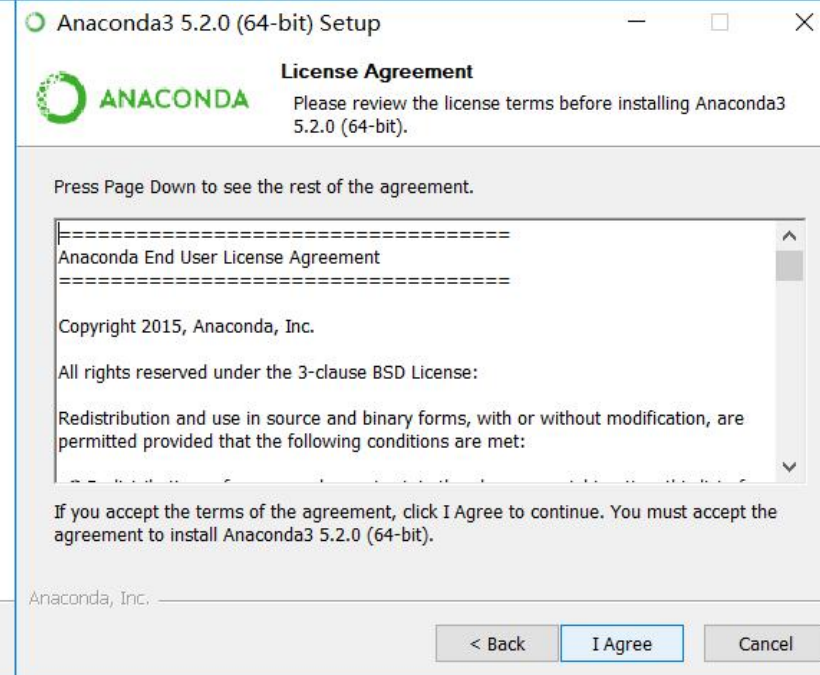
64-Bit Command Line Installer (454 MB)

Linux

Python 3.8

64-Bit (x86) Installer (550 MB)

64-Bit (Power8 and Power9) Installer (290 MB)



安装过程：PyCharm

<https://www.jetbrains.com/pycharm/download/#section=windows>



Version: 2020.2.1
Build: 202.6948.78
26 August 2020

[System requirements](#)

[Installation Instructions](#)

[Other versions](#)

Download PyCharm

[Windows](#)

[Mac](#)

[Linux](#)

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

Download

Free trial

Community

For pure Python development

Download

Free, open-source



TensorFlow

- TensorFlow 是谷歌的开发者创造的一款开源的深度学习框架，于 2015 年发布。
- TensorFlow 现已被公司、企业与创业公司广泛用于自动化工作任务和开发新系统，其在分布式训练支持、可扩展的生产和部署选项、多种设备（比如安卓）支持方面备受好评。



PyTorch

- PyTorch 是最新的深度学习框架之一，由 Facebook 的团队开发，并于 2017 年在 GitHub 上开源。
- PyTorch 很简洁、易于使用、支持动态计算图而且内存使用很高效，因此越来越受欢迎。
- PyTorch 因其简单易上手，而被广大 Researcher 所使用。



创建TensorFlow环境

- 创建conda环境
- 安装TensorFlow
- PyCharm添加对应环境



创建名为TensorFlow的conda环境

conda create --name tensorflow python=3.6

conda create -n tensorflow python=3.7

```
C:\Users\Administrator>conda create --name tensorflow python=3.6
Collecting package metadata (current_repodata.json): done
Solving environment: /
Warning: >10 possible package resolutions (only showing differing package
- https://mirrors.ustc.edu.cn/anaconda/pkgs/free::certifi-2016.2.28-py3
free::vs2015_runtime-14.0.25420-0, https://mirrors.ustc.edu.cn/anaconda/p
ustc.edu.cn/anaconda/pkgs/free::wincertstore-0.2-py36_0
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free::certifi-2016
nda/pkgs/free::vs2015_runtime-14.0.25420-0, https://mirrors.ustc.edu.cn/a
/mirrors.ustc.edu.cn/anaconda/pkgs/free::wincertstore-0.2-py36_0
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free::wheel-0.29.0
kgs/free::certifi-2016.2.28-py36_0, https://mirrors.ustc.edu.cn/anaconda/
/mirrors.ustc.edu.cn/anaconda/pkgs/free::wincertstore-0.2-py36_0
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free::certifi-2016
```


##

```
To activate this environment, use

$ conda activate tensorflow

To deactivate an active environment, use

$ conda deactivate
```

这样的显示表明 安装成功

<https://blog.csdn.net/PursueWin>

conda activate tensorflow

这样显示表明添加成功

```
C:\Users\Administrator>activate tensorflow
(tensorflow) C:\Users\Administrator>conda info --envs
```



pip install tensorflow
pip install tensorflow==1.14.0

```
(tensorflow) C:\Users\Administrator>pip install --upgrade --ignore-installed tensorflow
Collecting tensorflow
  Using cached https://files.pythonhosted.org/packages/bf/4a/5c86ed8b245aa48f9f819b13a0a9039e9126ba19fdd0c7e0b8026c12315a/tensorflow-1.14.0-cp36-cp36m-win_amd64.whl
Collecting absl-py>=0.7.0 (from tensorflow)
Collecting numpy<2.0, >=1.14.5 (from tensorflow)
  Using cached https://files.pythonhosted.org/packages/b7/c1/a58630a439aa10a285169b4a122bc9f7a9a4392e4ec39602f0a60b2693db/numpy-1.17.0-cp36-cp36m-win_amd64.whl
Collecting protobuf>=3.6.1 (from tensorflow)
  Using cached https://files.pythonhosted.org/packages/74/74/44ec96740ed10ae6d0508efc083c6b7e605c509bc32136e9aea840d09daf/protobuf-3.9.0-cp36-cp36m-win_amd64.whl
Collecting wrapt>=1.11.1 (from tensorflow)
Collecting keras-applications>=1.0.6 (from tensorflow)
```

安装TensorFlow

<https://blog.csdn.net/PursueWin>

PyCharm中添加TensorFlow (环境)

PyCharm [E:\Pycharm] - PyCharm (Administrat

File Edit View Navigate Code Refactor R

New Project...
New... Alt+Insert
New Scratch File Ctrl+Alt+Shift+Insert
Open...
Open URL...
Save As...
Open Recent
Close Project
Rename Project...
Settings... Ctrl+Alt+S
Other Settings
Import Settings...
Export Settings...
Settings Repository...
Sync Settings to JetBrains Account...
Save All Ctrl+S
Synchronize Ctrl+Alt+Y
Invalidate Caches / Restart...
Print...
Associate with File Type...
Power Save Mode
Exit

Settings

Project: Pycharm > Project Interpreter For current project

Project Interpreter: Python 3.6 (tensorflow) D:\SoftwareInstall\AnacondaAndTensorFlow\envs\tensorflow\python.exe

Package Version Latest version

certifi	2016.2.28	2019.6.16
cycler	0.10.0	0.10.0
gast	0.2.2	0.2.2
google-pasta	0.1.7	0.1.7
grpcio	1.22.0	1.16.0
h5py	2.9.0	2.9.0
joblib	0.13.2	0.13.2
keras-applications	1.0.8	1.0.7
keras-preprocessing	1.1.0	1.1.0
kiwisolver	1.1.0	1.1.0
markdown	3.1.1	3.1.1
matplotlib	3.1.1	3.1.1
numpy	1.17.0	1.17.0
pip	19.2.1	19.2.1
protobuf	3.9.0	3.9.1
pyparsing	2.4.1.1	2.4.2
python	3.6.2	3.7.3
python-dateutil	2.8.0	2.8.0
scikit-learn	0.21.1	0.21.3
scipy	1.3.0	0.19.1
setuptools	41.0.1	41.0.1
six	1.12.0	1.12.0
tensorboard	1.14.0	1.14.0
tensorflow	1.14.0	1.13.1

按照你安装的位置寻找在TensorFlow下的Python

1

2

https://blog.csdn.net/PursueWin

TensorFlow 测试

```
import tensorflow as tf

hello = tf.constant('hello, tf')
sess = tf.compat.v1.Session()
print(sess.run(hello))
```

```
b'hello, tf'
```

```
Process finished with exit code 0
```

Python简单图解

脚本文件一般用 .py 后缀

单行注释

导入其它代码模块

注意! Python最好也最个性的语法:
使用缩进来代替其它语句块声明;
一般建议每个层级用4个空格来缩进。

变量得先实例化
才可进一步计算

单行的语句块, 其实可以不用换行的,
但是, 建议清晰起见, 规范点:
- 另起一行
- 缩进一级

中文用户一定得先用这行来声明编码, 同时文件本身也得存储成UTF-8编码!

模块名, 其实导入了 os.py

函数名 "main" 在这儿并不是必须的, 调用在这段脚本的最后部分;

声明单行字符串, 使用双/单引号都成,
注意对字符串中的引号进行逃逸处理!

函数调用, 声明在后续代码;

字符可乘, 等于: '====='

调用了 os 模块中的函数

连接字符串

内置的列表类型对象, 其实可以包含不同类型数据,
甚至可以包含其它列表对象;

在循环中, i 指代了列表中按顺序的每个 "food"

range() 内置函数, 返回类似
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
的数字列表, 注意 for in 循环语句使用冒号结束声明!

在M\$中很好的支持UTF-8的编辑器不多,
跨平台又支持Py特性的更少;
好在我们有 Limodou 贡献的 UliPad
这一编辑器本身就是Py实现的!
(基于wxPython)

```
1 # -*- coding: utf-8 -*-
2 # Quick Python Script Explanation for Progeammers
3 # 给程序员的超快速Py脚本解说
4 import os
5
6 def main():
7     print 'Hello World!'
8
9     print "这是Alice\'的问候."
10    print '这是Bob\'的问候.'
11
12    foo(5, 10)
13
14    print '=' * 10
15    print '这将直接执行'+os.getcwd()
16
17    counter = 0
18    counter += 1
19
20    food = ['苹果', '杏子', '李子', '梨']
21    for i in food:
22        print '俺就爱整只:'+i
23
24    print '数到10'
25    for i in range(10):
26        print i
```

关于
UliPad 4.0
作者: Limodou (limodou@gmail.com)
如果你有任何问题请与我联系。
[The UliPad project homepage](#)
[The UliPad maillist](#)
[The UliPad Snippets Site](#)
[My Blog](#)
[Contact me](#)
确定

函式声明,
注意使用冒号结束声明

```
27 L
28 def foo(param1, secondParam):
29     res = param1+secondParam
30     print '%s 加 %s 等于 %s'%(param1, secondParam, res)
31     if res < 50:
32         print '这个'
33     elif (res>=50) and ((param1==42) or (secondParam==24)):
34         print '那个'
35     else:
36         print '嗯....'
37     return res # 这是单行注释
38     '''这是多
39     行注释.....'''
40
41 if __name__ == '__main__':
42     main()
```

字串的格式化输出基本类似C语言的

判定式也基本和C语言的相同

用冒号来结束判断句,
在 if elif else 行最后

多行注释的内容不用遵守当前缩进
只要开始的''' 缩进正确就成!

逻辑运算符,不使用 && 和 ||,使用直观的E文单词

每级语法块不用}之类的括号引领!
直接回车+4空格
(当然,要在当前缩进基础上)

这是单行注释

这都是合法注释

一般在脚本最后调用主函式 main();而且使用 内置的运行脚本名来判定;
当且仅当我们直接运行当前脚本时, __name__ 才为 __main__
这样当脚本被当作模块进行 import 导入时,并不运行 main()
所以,一般这里是进行测试代码安置的...

第一个Python程序

```
print ("Hello, Python!")
```

输出结果：

Hello, Python!

Python变量

- 数字类型:

int

long (在 Python3.X 版本中 long 类型被移除, 使用 int 替代)

float

complex (复数)

int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j

Python变量

- 字符串或串(String)是由数字、字母、下划线组成的一串字符。

$s = "a_1a_2\cdots a_n"$

```
str = 'Hello World!'
```

```
print str           # 输出完整字符串
print str[0]        # 输出字符串中的第一个字符
print str[2:5]      # 输出字符串中第三个至第六个之间的字符串
print str[2:]       # 输出从第三个字符开始的字符串
print str * 2       # 输出字符串两次
print str + "TEST"  # 输出连接的字符串
```

以上实例输出结果:

```
Hello World!
H
llo
llo World!
Hello World!Hello World!
Hello World!TEST
```

Python变量

List（列表）是 Python 中使用最频繁的数据类型。

```
list = [ 'runoob', 786 , 2.23, 'john', 70.2 ]  
tinylist = [123, 'john']
```

```
print list           # 输出完整列表  
print list[0]        # 输出列表的第一个元素  
print list[1:3]       # 输出第二个至第三个元素  
print list[2:]        # 输出从第三个开始至列表末尾的所有元素  
print tinylist * 2    # 输出列表两次  
print list + tinylist # 打印组合的列表
```

```
['runoob', 786, 2.23, 'john', 70.2]
```

```
runoob
```

```
[786, 2.23]
```

```
[2.23, 'john', 70.2]
```

```
[123, 'john', 123, 'john']
```

```
['runoob', 786, 2.23, 'john', 70.2, 123, 'john']
```

Python运算符

- 举个简单的例子 $4 + 5 = 9$ 。例子中，4 和 5 被称为操作数， "+" 称为运算符。
- 算术运算符
- 比较（关系）运算符
- 赋值运算符
- 逻辑运算符

Python算数运算符

以下假设变量： **a=10**, **b=20**：

运算符	描述	实例
+	加 - 两个对象相加	a + b 输出结果 30
-	减 - 得到负数或是一个数减去另一个数	a - b 输出结果 -10
*	乘 - 两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果 200
/	除 - x除以y	b / a 输出结果 2
%	取模 - 返回除法的余数	b % a 输出结果 0
**	幂 - 返回x的y次幂	a**b 为10的20次方， 输出结果 100000000000000000000
//	取整除 - 返回商的整数部分（向下取整）	<pre>>>> 9//2 4 >>> -9//2 -5</pre>

Python算数运算符

注意：Python2.x 里，整数除整数，只能得出整数。如果要得到小数部分，把其中一个数改成浮点数即可。

```
>>> 1/2
0
>>> 1.0/2
0.5
>>> 1/float(2)
0.5
```

- 在 Python 2.x 中 / 除法就跟我们熟悉的大多数语言，比如 Java 和 C，整数相除的结果是一个整数，把小数部分完全忽略掉，浮点数除法会保留小数点的部分得到一个浮点数的结果。
- 在 Python 3.x 中 / 除法不再这么做了，对于整数之间的相除，结果也会是浮点数。

Python比较运算符

以下假设变量a为10，变量b为20：

运算符	描述	实例
==	等于 - 比较对象是否相等	(a == b) 返回 False。
!=	不等于 - 比较两个对象是否不相等	(a != b) 返回 true。
<>	不等于 - 比较两个对象是否不相等。 python3 已废弃。	(a <> b) 返回 true。这个运算符类似 != 。
>	大于 - 返回x是否大于y	(a > b) 返回 False。
<	小于 - 返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。	(a < b) 返回 true。
>=	大于等于 - 返回x是否大于等于y。	(a >= b) 返回 False。
<=	小于等于 - 返回x是否小于等于y。	(a <= b) 返回 true。

Python赋值运算符

以下假设变量a为10，变量b为20：

运算符	描述	实例
=	简单的赋值运算符	<code>c = a + b</code> 将 <code>a + b</code> 的运算结果赋值为 <code>c</code>
+=	加法赋值运算符	<code>c += a</code> 等效于 <code>c = c + a</code>
-=	减法赋值运算符	<code>c -= a</code> 等效于 <code>c = c - a</code>
*=	乘法赋值运算符	<code>c *= a</code> 等效于 <code>c = c * a</code>
/=	除法赋值运算符	<code>c /= a</code> 等效于 <code>c = c / a</code>
%=	取模赋值运算符	<code>c %= a</code> 等效于 <code>c = c % a</code>
**=	幂赋值运算符	<code>c **= a</code> 等效于 <code>c = c ** a</code>
//=	取整除赋值运算符	<code>c //= a</code> 等效于 <code>c = c // a</code>

Python逻辑运算符

Python语言支持逻辑运算符，以下假设变量 a 为 10, b为 20:

运算符	逻辑表达式	描述	实例
and	x and y	布尔"与" - 如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。	(a and b) 返回 20。
or	x or y	布尔"或" - 如果 x 是非 0，它返回 x 的计算值，否则它返回 y 的计算值。	(a or b) 返回 10。
not	not x	布尔"非" - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。	not(a and b) 返回 False

Python条件语句

- Python条件语句是通过一条或多条语句的执行结果（True或者False）来决定执行的代码块。
- Python 编程中 if 语句用于控制程序的执行，基本形式为：

```
if 判断条件:  
    执行语句.....  
else:  
    执行语句.....
```

- 当判断条件为多个值时，可以使用以下形式：

```
if 判断条件1:  
    执行语句1.....  
elif 判断条件2:  
    执行语句2.....  
elif 判断条件3:  
    执行语句3.....  
else:  
    执行语句4.....
```

Python条件语句

```
flag = False
name = 'luren'
if name == 'python':
    flag = True
    print 'welcome boss'
else:
    print name
```

判断变量是否为 python
条件成立时设置标志为真
并输出欢迎信息
条件不成立时输出变量名称

输出结果为：

luren

输出结果

Python循环语句

- 循环语句允许我们执行一个语句或语句组多次。

循环类型	描述
<u>while 循环</u>	在给定的判断条件为 true 时执行循环体，否则退出循环体。
<u>for 循环</u>	重复执行语句

While循环语句

Python 编程中 while 语句用于循环执行程序，即在某条件下，循环执行某段程序，以处理需要重复处理的相同任务。其基本形式为：

```
while 判断条件(condition):  
    执行语句(statements).....
```

While循环语句

```
count = 0
while (count < 9):
    print 'The count is:', count
    count = count + 1

print "Good bye!"
```

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
Good bye!
```

For循环语句

- Python for循环可以遍历任何序列的项目， 如一个列表或者一个字符串。
- for循环的语法格式如下：

```
for iterating_var in sequence:  
    statements(s)
```

For循环语句

```
for letter in 'Python':    # 第一个实例
    print '当前字母 :', letter

fruits = ['banana', 'apple', 'mango']
for fruit in fruits:       # 第二个实例
    print '当前水果 :', fruit

print "Good bye!"
```

```
当前字母 : P
当前字母 : y
当前字母 : t
当前字母 : h
当前字母 : o
当前字母 : n
当前水果 : banana
当前水果 : apple
当前水果 : mango
Good bye!
```

For循环语句

常用的通过序列索引迭代的方式：

```
fruits = ['banana', 'apple', 'mango']  
for index in range(len(fruits)):  
    print '当前水果 :', fruits[index]  
  
print "Good bye!"
```

```
当前水果 : banana  
当前水果 : apple  
当前水果 : mango  
Good bye!
```

Python函数

- 函数是组织好的，可重复使用的，用来实现单一，或相关联功能的代码段。
- 函数能提高应用的模块性，和代码的重复利用率。你已经知道Python提供了许多内建函数，比如print()。但你也可以自己创建函数，这被叫做用户自定义函数。

如何定义函数

你可以定义一个由自己想要功能的函数，以下是简单的规则：

- 函数代码块以 `def` 关键词开头，后接函数标识符名称和圆括号`()`。
- 任何传入参数和自变量必须放在圆括号中间。圆括号之间可以用于定义参数。
- 函数内容以冒号起始，并且缩进。
- `return [表达式]` 结束函数，选择性地返回一个值给调用方。不带表达式的`return`相当于返回 `None`。

Python函数

调用函数时，默认参数的值如果没有传入，则被认为是默认值。

#可写函数说明

```
def printinfo( name, age = 35 ):
    "打印任何传入的字符串"
    print "Name: ", name
    print "Age ", age
    return
```

#调用printinfo函数

```
printinfo( age=50, name="miki" )
printinfo( name="miki" )
```

Name: miki

Age 50

Name: miki

Age 35

Python函数

- return语句[表达式]退出函数，选择性地向调用方返回一个表达式。
不带参数值的return语句返回None。

可写函数说明

```
def sum( arg1, arg2 ):  
    # 返回2个参数的和."  
    total = arg1 + arg2  
    print "函数内 : ", total  
    return total
```

调用sum函数

```
total = sum( 10, 20 )
```

```
函数内 : 30
```

Python函数

- 定义在函数内部的变量拥有一个局部作用域，定义在函数外的拥有全局作用域。
- 局部变量只能在其被声明的函数内部访问，而全局变量可以在整个程序范围内访问。调用函数时，所有在函数内声明的变量名称都将被加入到作用域中。

```
total = 0 # 这是一个全局变量
# 可写函数说明
def sum( arg1, arg2 ):
    #返回2个参数的和."
    total = arg1 + arg2 # total在这里是局部变量.
    print "函数内是局部变量 :", total
    return total

#调用sum函数
sum( 10, 20 )
print "函数外是全局变量 :", total
```

函数内是局部变量 : 30

函数外是全局变量 : 0

NumPy

- NumPy(Numerical Python) 是 Python 语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。
- NumPy 的前身 Numeric 最早是由 Jim Hugunin 与其它协作者共同开发，2005 年，Travis Oliphant 在 Numeric 中结合了另一个同性质的程序库 Numarray 的特色，并加入了其它扩展而开发了 NumPy。NumPy 为开放源代码并且由许多协作者共同维护开发。
- 教程：<https://www.runoob.com/numpy/numpy-tutorial.html>
- NumPy 官网 <http://www.numpy.org/>
- NumPy 源代码：<https://github.com/numpy/numpy>

SciPy

- SciPy 官网: <https://www.scipy.org/>
- SciPy 源代码: <https://github.com/scipy/scipy>
- Scipy依赖于Numpy
- Scipy包含的功能: 最优化、线性代数、积分、插值、拟合、特殊函数、快速傅里叶变换、信号处理、图像处理、常微分方程求解器等
- 应用场景: Scipy是高端科学计算工具包, 用于数学、科学、工程学等领域

Matplotlib

- Matplotlib 官网: <https://matplotlib.org/>
- Matplotlib 源代码: <https://github.com/matplotlib/matplotlib>
- Matplotlib 是一个 Python 的 2D 绘图库, 它以各种硬拷贝格式和跨平台的交互式环境生成出版质量级别的图形。