Python安装及基础

目 录

- 一. 为什么是Python？
- 二. Python的安装
- 三. PyCharm的安装
- 四. Python的基础
- 五. Tensorflow的安装

一. 为什么是Python？

自身特点

- 优雅而健壮的编程语言
- 注重如何解决问题
- 简单易学，功能丰富
- 高级
- 面向对象
- 可移植性
- 可扩展性
- 可嵌入性
- 健壮性
- 解释性
- 易学易读易用

一. 为什么是Python？

深度学习上的应用

（A）基于Python的深度学习库、深度学习方向、机器学习方向、自然语言处理方向的一些网站基本都是通过Python来实现的；

（B）机器学习，尤其是现火爆的深度学习，其工具框架大都提供Python接口；

（C）Python在科学计算领域一直有着较好的声誉，其简洁清晰的语法以及丰富的计算工具，深受此领域开发者喜爱。

（D）早在深度学习以及Tensorflow等框架流行之前，Python中即有scikit-learn，能够很方便地完成几乎所有机器学习模型，从经典数据集下载到构建模型只需要简单的几行代码。配合Pandas、matplotlib等工具，能很简单地进行调整。

一. 为什么是Python？

深度学习上的应用

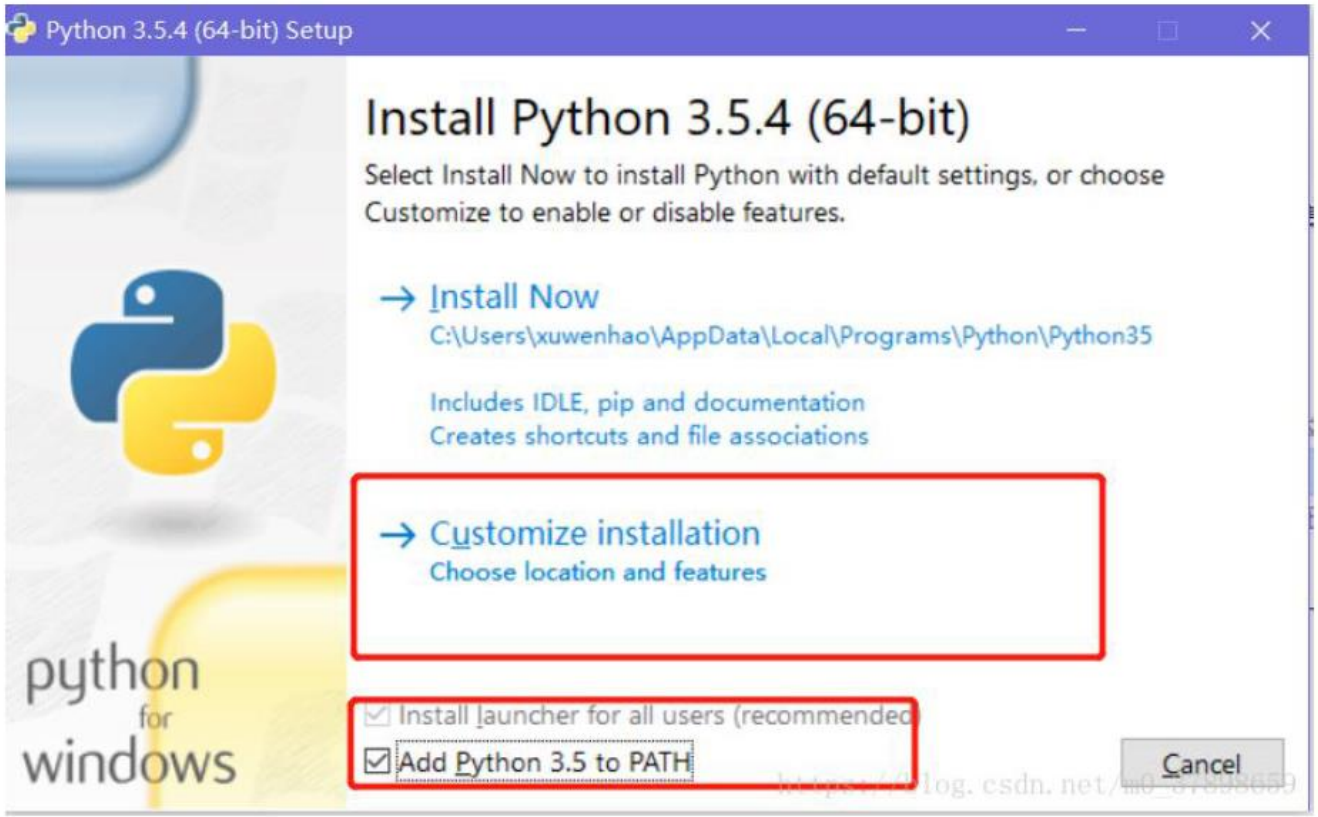
（E）Tensorflow、PyTorch、MXNet、Keras等深度学习框架更是极大地拓展了机器学习的可能。使用TF编写一个手写数字识别的深度学习网络仅仅需要寥寥数十行代码，即可借助底层实现，方便地调用包括GPU在内的大量资源完成工作。

（F）无论什么框架，Python只是作为前端描述用的语言，实际计算则是通过底层的C/C++实现。由于Python能很方便地引入和使用C/C++项目和库，从而实现功能和性能上的扩展，这样的大规模计算中，让开发者更关注逻辑于数据本身，而从内存分配等繁杂工作中解放出来，是Python被广泛应用到机器学习领域的重要原因。

二. Python的安装

Python在Windows的安装

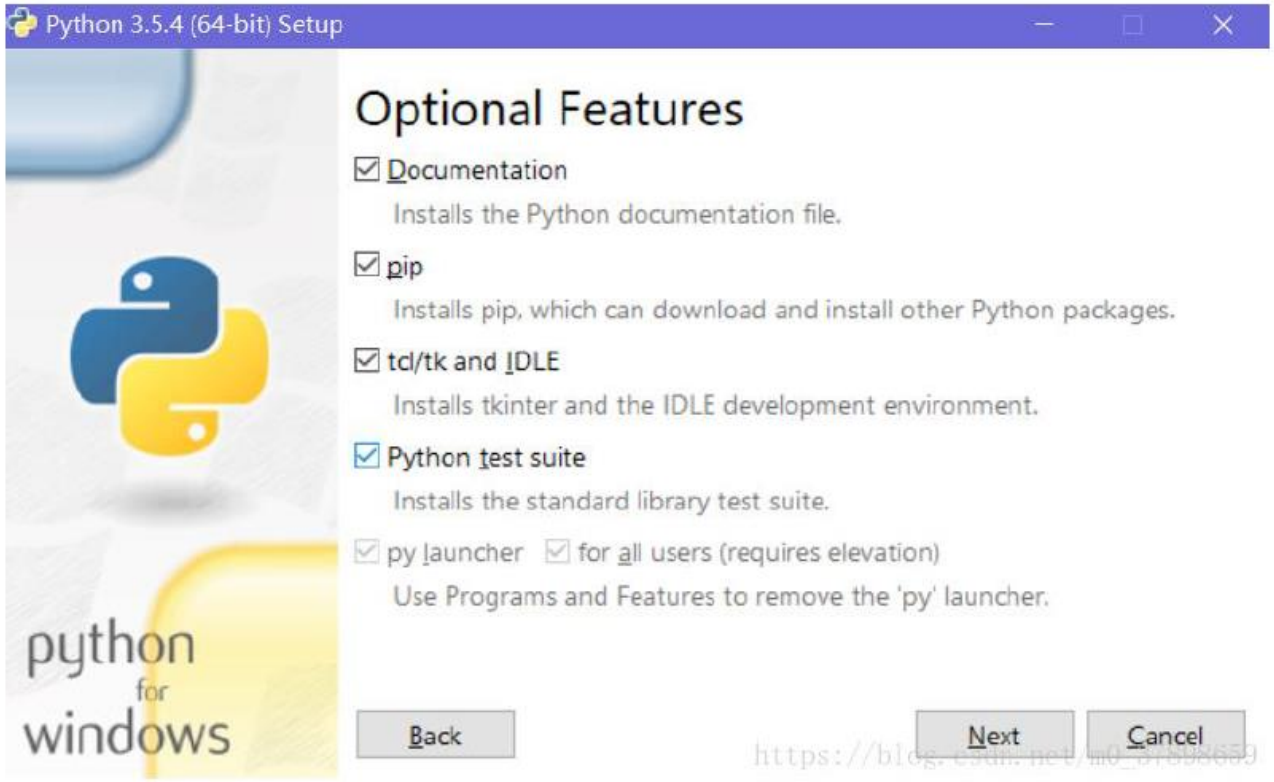
1、注意，一定要选择“Add Python 3.5 to PATH”，这个是直接自动添加到环境变量中。



二. Python的安装

Python在Windows的安装

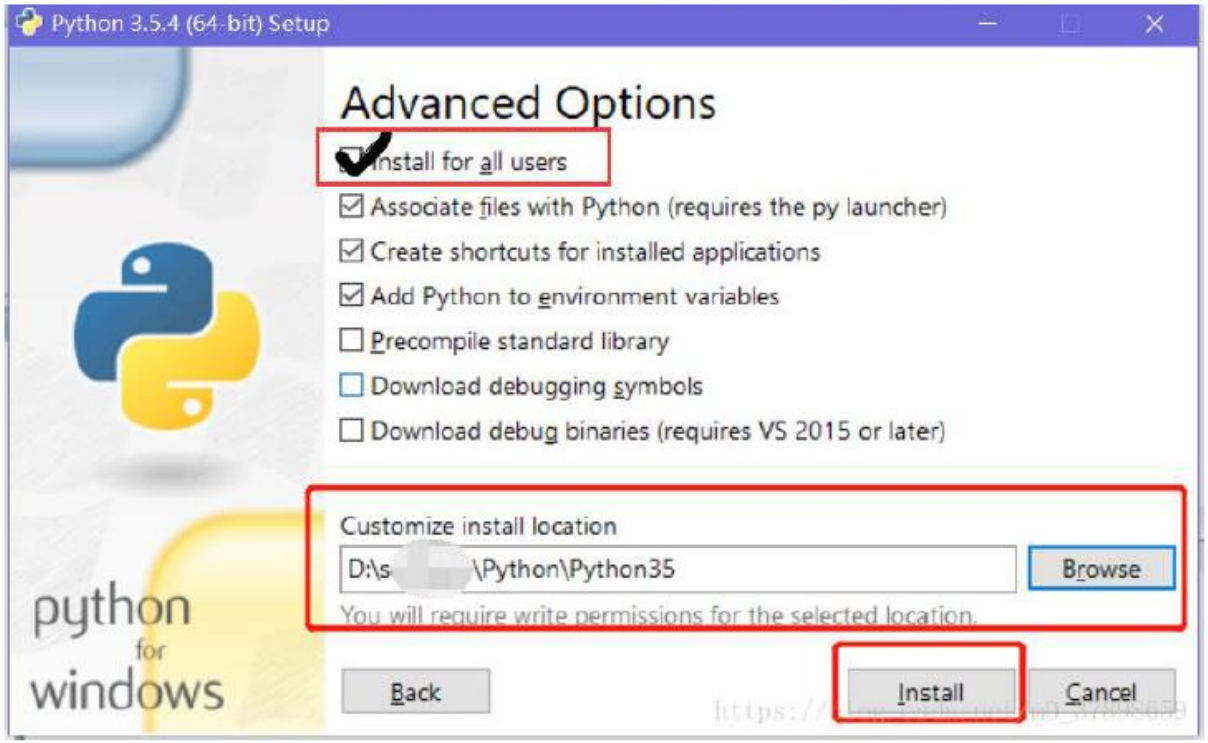
2、这一步全选就好了，然后 Next



二. Python的安装

Python在Windows的安装

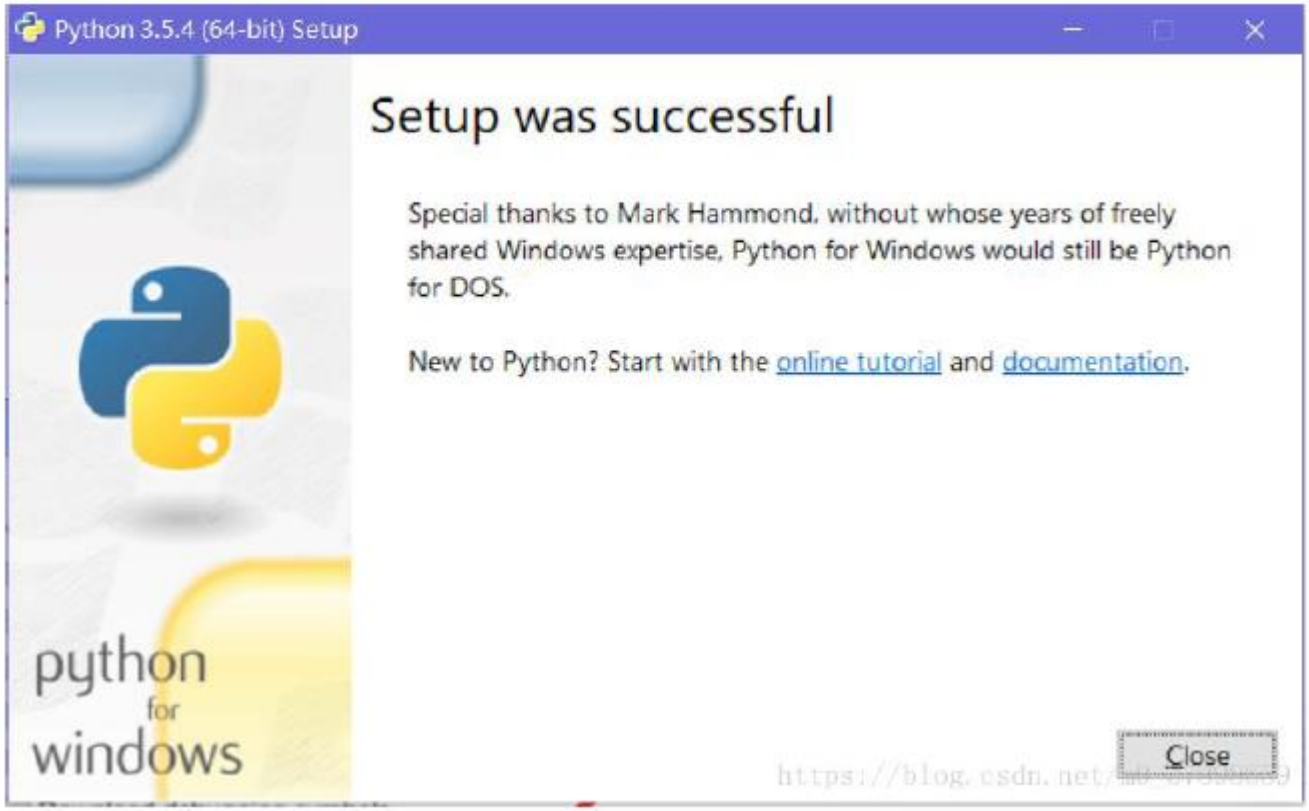
3、这一步勾选 “install for all users”， 可以改变下载路径，然后点击安装。



二. Python的安装

Python在Windows的安装

4、安装成功的页面。



二. Python的安装

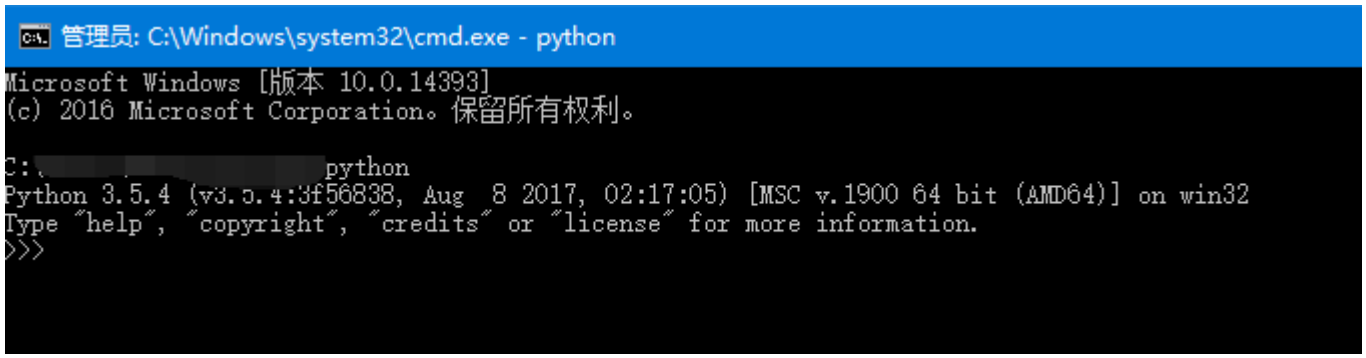
Python在Windows的安装

5、验证是否安装成功。

打开 cmd（win+R），直接输入 python

有版本信息出来，证明成功！

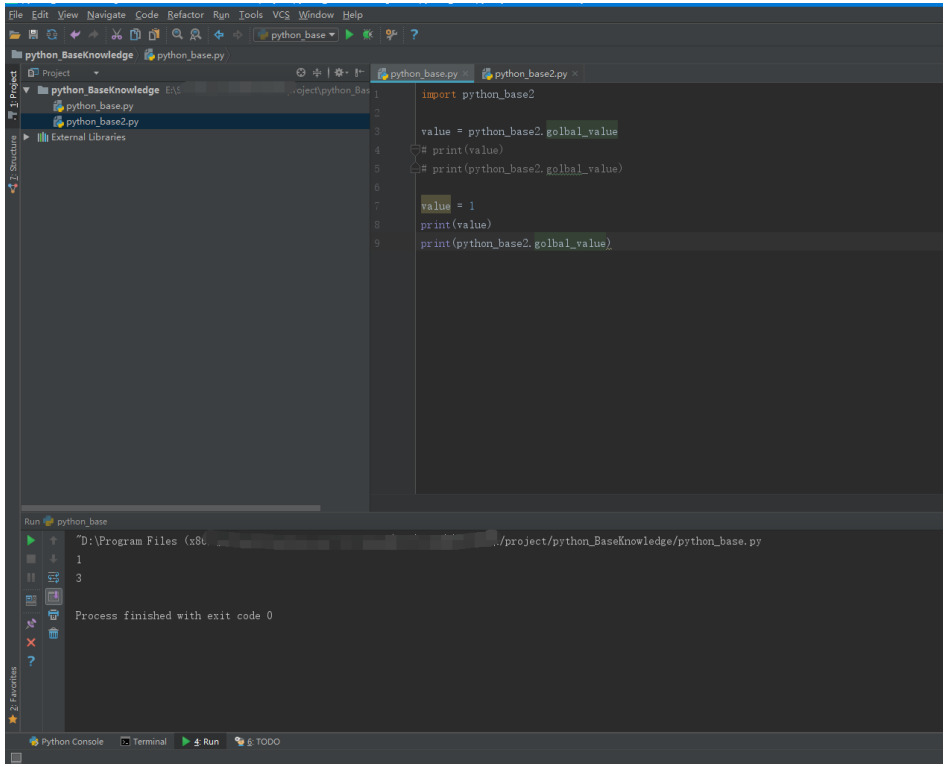
然后输入“quit()”退出 python 环境（或者直接关闭 cmd）



三. PyCharm的安装

PyCharm是什么？

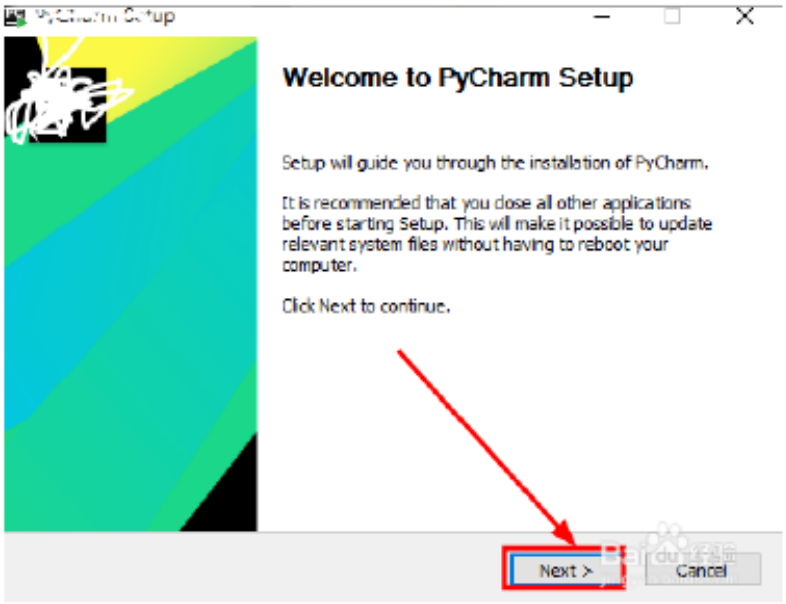
PyCharm是一种Python IDE，带有一整套可以帮助用户在使用Python语言开发时提高其效率的工具，比如调试、语法高亮、Project管理、代码跳转、智能提示、自动完成、单元测试、版本控制。



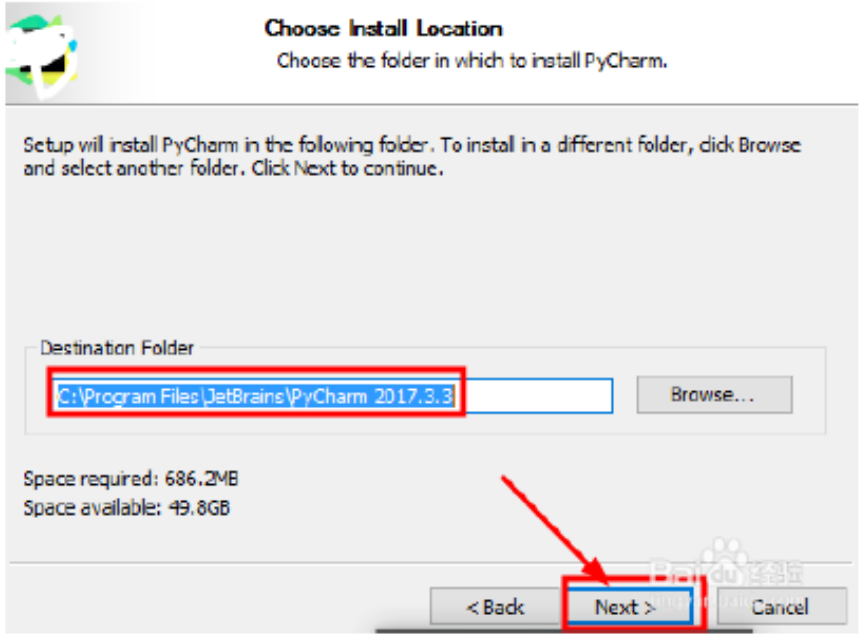
三. PyCharm的安装

PyCharm在Windows的安装

1、首先双击应用程序打开如图所示的安装向导界面，点击 Next 按钮。



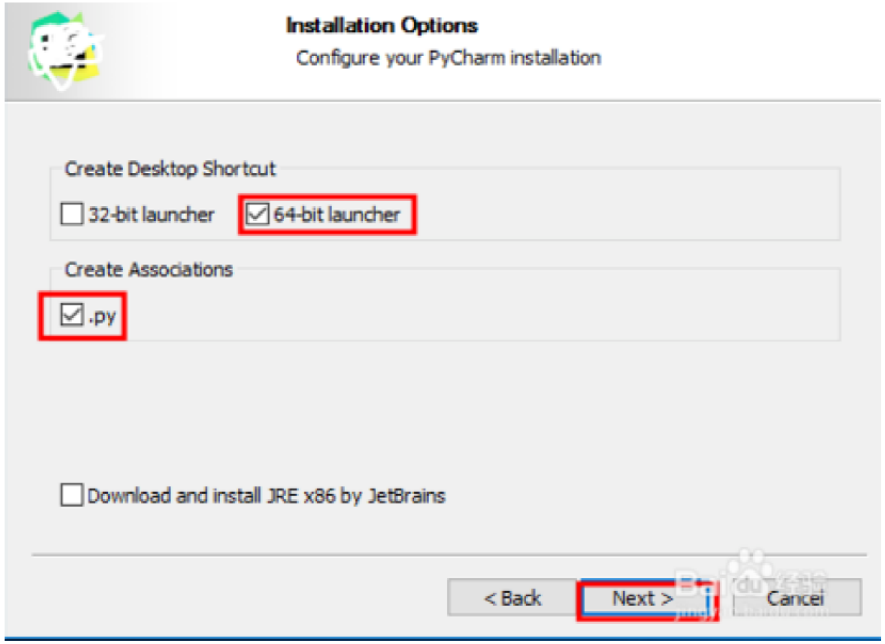
2、然后选择安装位置界面，可以修改默认的安装位置。点击 Next 按钮



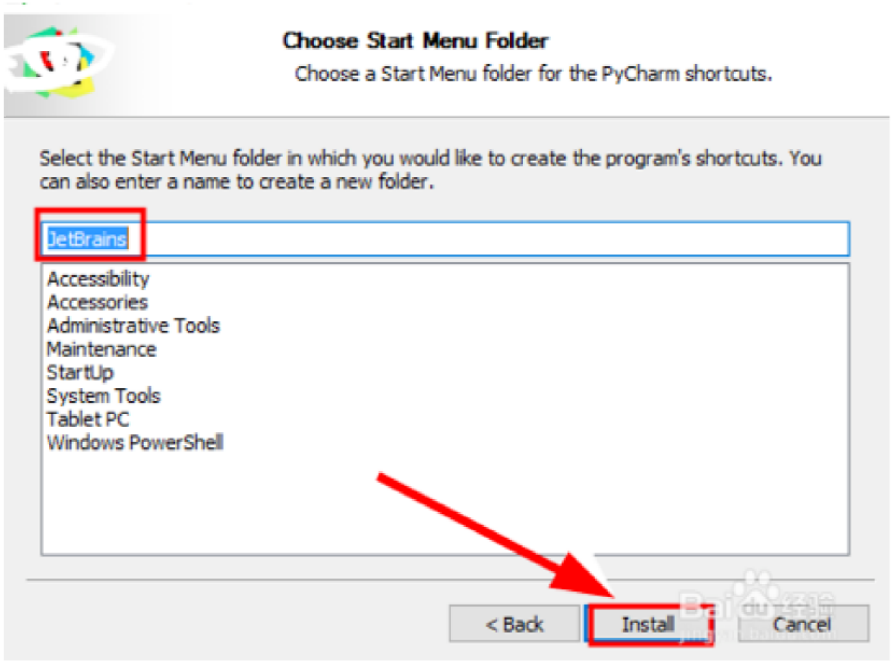
三. PyCharm的安装

PyCharm在Windows的安装

3、选择安装的系统版本，根据你的系统版本来，其他的可以根据个人情况勾选。



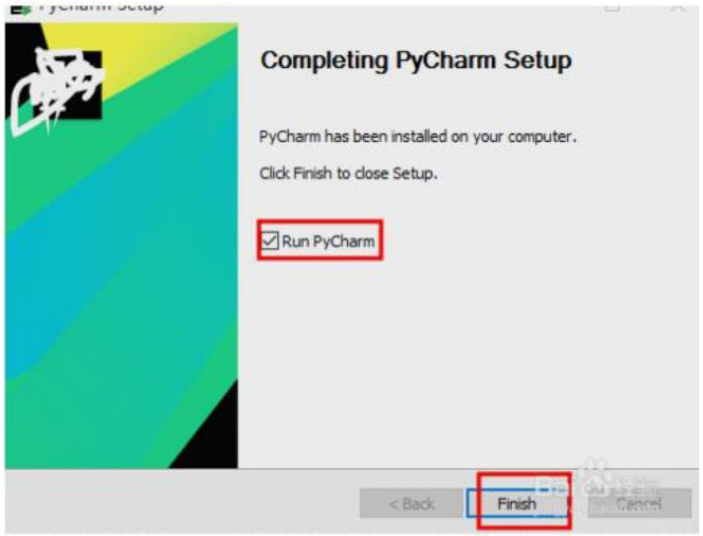
4、选择要创建的开始菜单选项的文件夹名称，然后点击 Install 按钮。



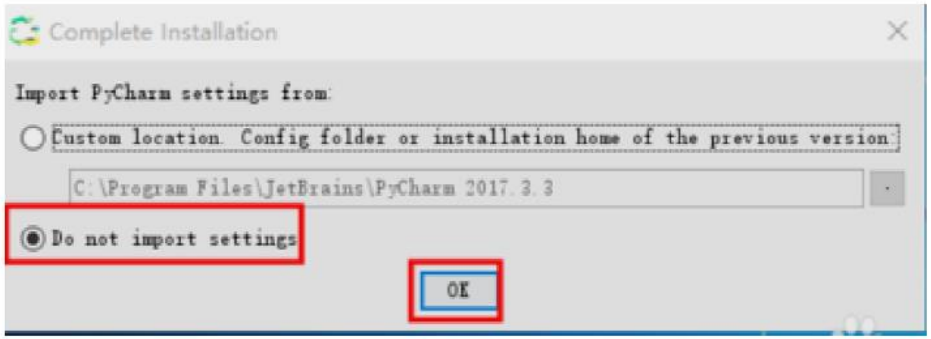
三. PyCharm的安装

PyCharm在Windows的安装

5、等待安装完成后出现如图所示的界面，点击 Finish 按钮即可，这里勾选了立即启动选项。



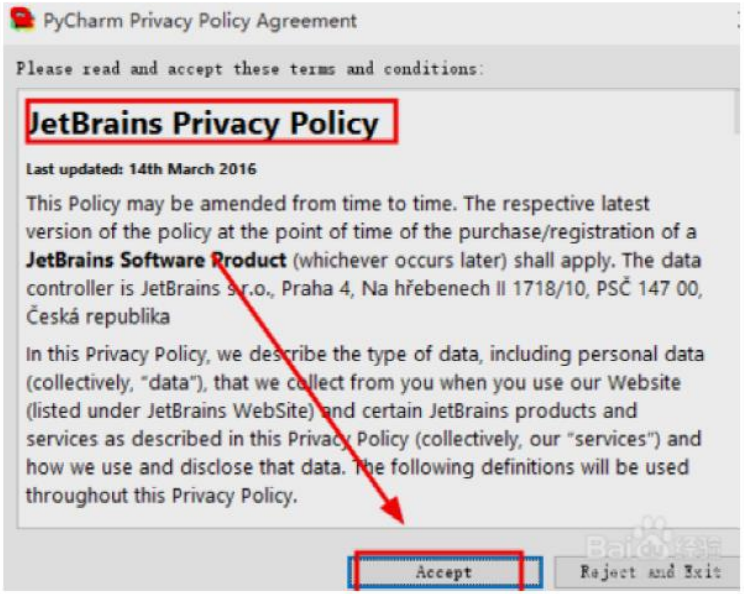
6、勾选第二个选项不加载配置文件，点击 OK 按钮



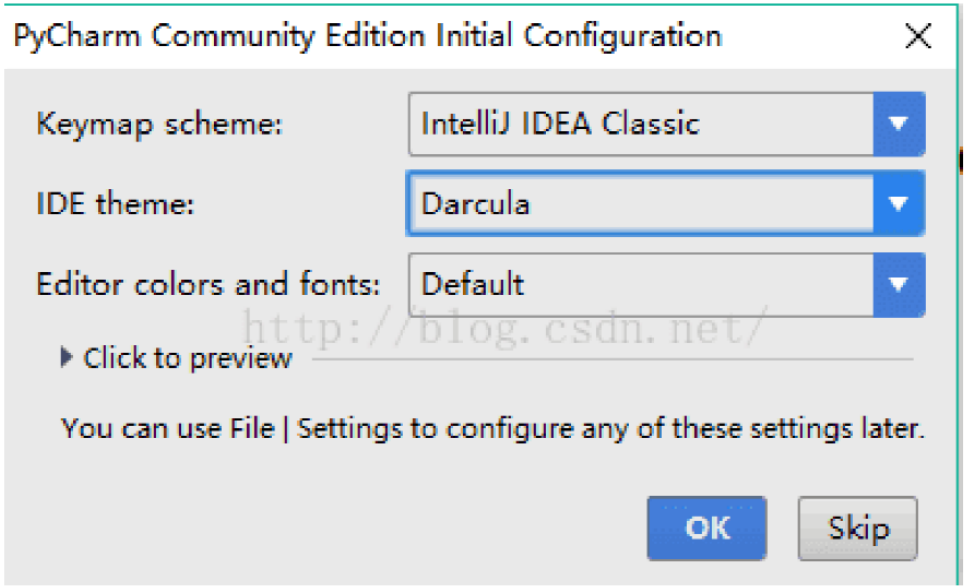
三. PyCharm的安装

PyCharm在Windows的安装

7、阅读应用程序的使用许可协议，点击下方的 I Agree 按钮即可



8、选择如下，然后点击图中的 ok 进入下一步

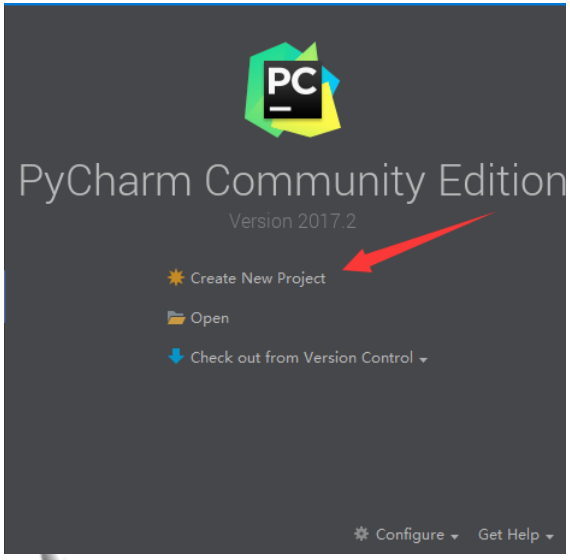


http://blog.csdn.net/

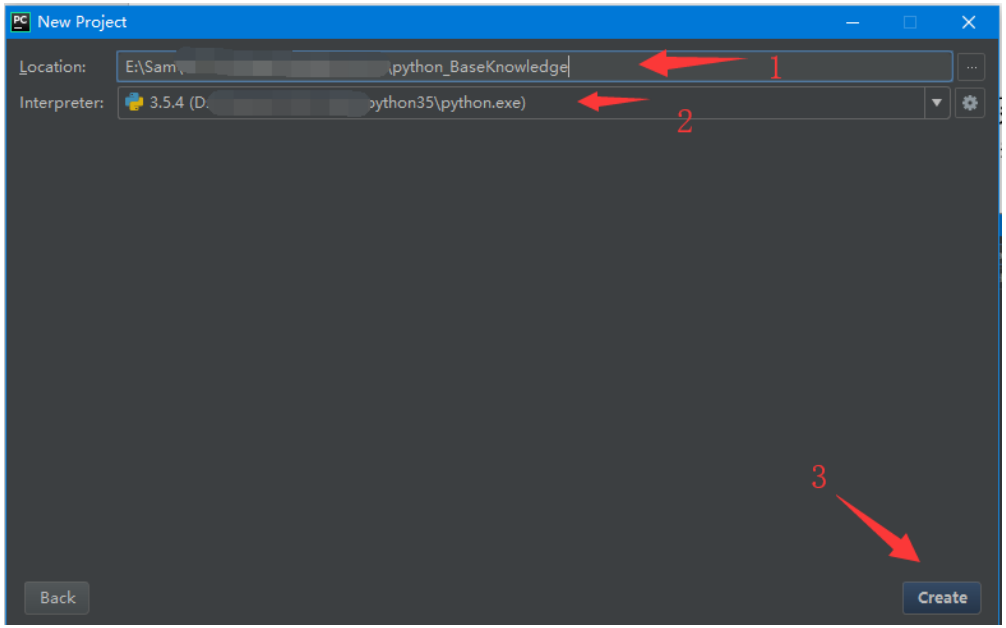
三. PyCharm的安装

PyCharm在Windows的安装

9、点击Create New Project



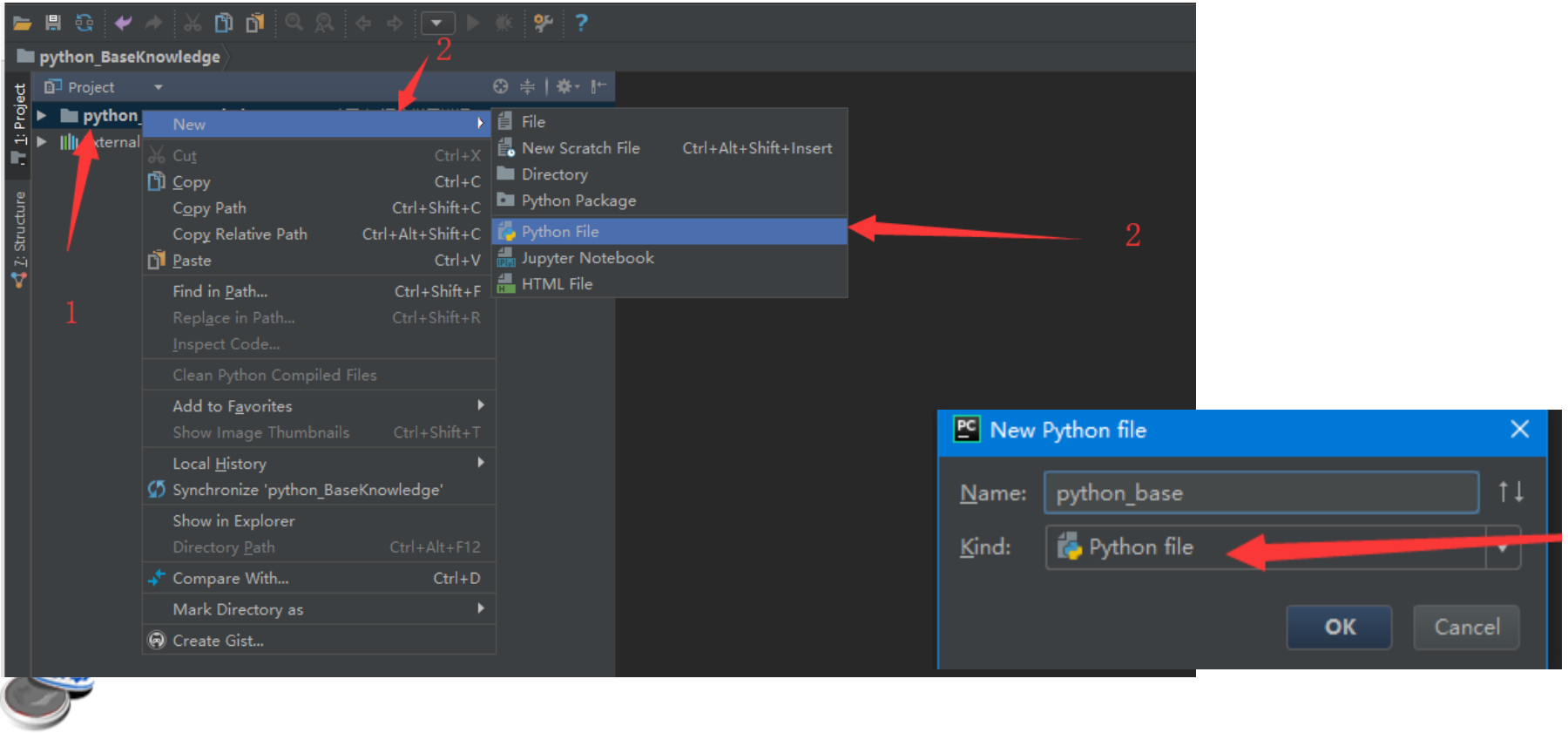
10、Location是选择创建的程序所在的文件夹，Interpreter选择的是安装在windows下的python.exe选择好后，点击create



三. PyCharm的安装

PyCharm在Windows的安装

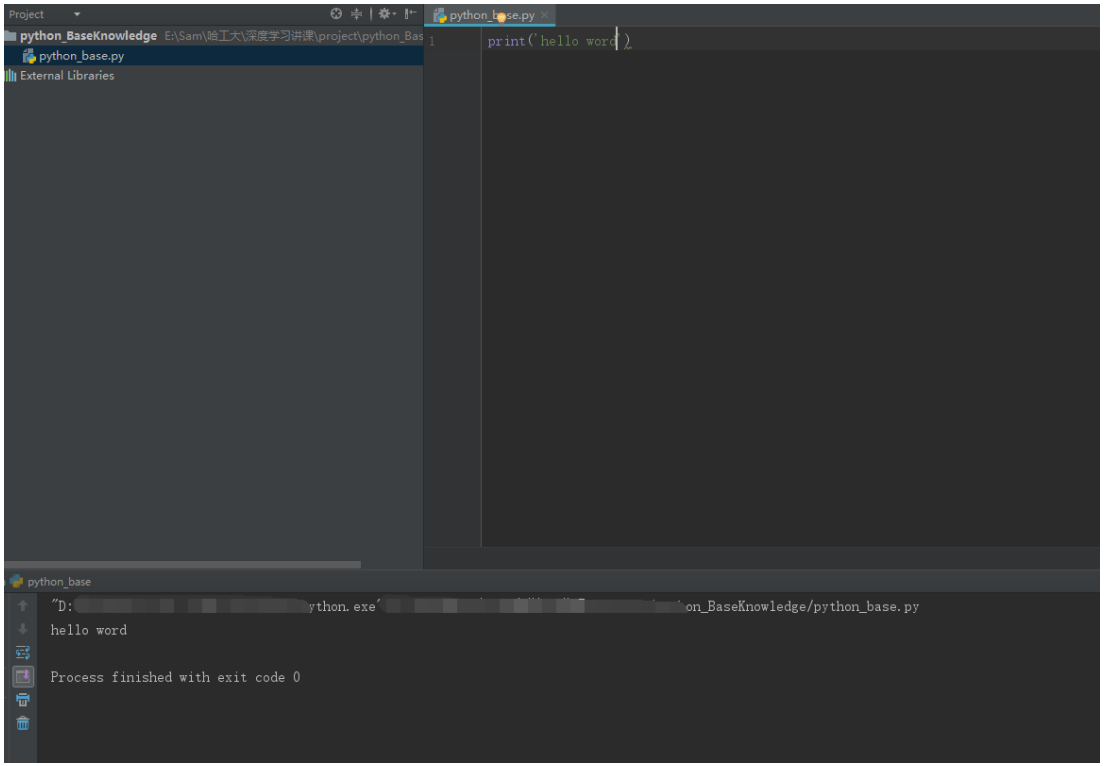
11、鼠标右击图中箭头指向的地方，然后最后选择python file，在弹出的框中填写文件名（任意填写）



三. PyCharm的安装

PyCharm在Windows的安装

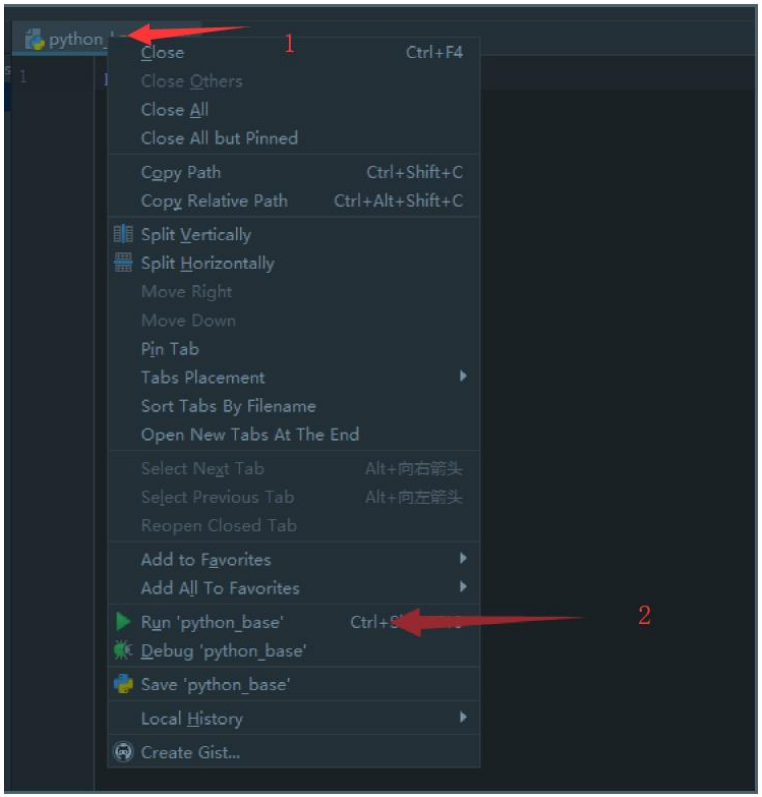
12、文件创建成功后便进入如下的界面，便可以编写自己的程序了，当然如果你对这个界面不满意的话，可以自己设置背景，这里就不详细说明了（自行百度即可）。



三. PyCharm的安装

PyCharm在Windows的安装

13、运行程序，右击文件的小框，选择RUN或者Debug。



四. Python的基础

Python的基础语法

（A）python最具特色的就是使用缩进来表示代码块，不需要使用大括号 {} 。缩进的空格数是可变的，但是同一个代码块的语句必须包含相同的缩进空格数。实例如下：

```
if True:
    print("True")
else:
    print("False")
```

（B）Python中单行注释以 **#** 开头，实例如下

```
# 第一个注释
print("Hello, Python!")  # 第二个注释
```

（C）print 默认输出是换行的，如果要实现不换行需要在变量末尾加上 **end=""**：

```
# 不换行输出
print(x, end=" ")
```

四. Python的基础

Python的数据结构

(A)**List**是Python中最基本的数据结构。
序列中的每个元素都分配一个索引，第一个索引是0，第二个索引是1…
List可以进行的操作包括: 索引，切片，加，乘，检查成员。

样例：

```
list1 = ['Google', 'Runoob', 1997, 2000]
list2 = [1, 2, 3, 4, 5 ]
list3 = ["a", "b", "c", "d"]
```

四. Python的基础

Python的数据结构

(A)**List**内建函数：

len(list)

列表元素个数

max(list)

返回列表元素最大值

min(list)

返回列表元素最小值

list(seq)

将元组转换为列表

- list.append(obj) 向列表中添加一个对象obj
- list.count(obj) 返回一个对象obj在列表中出现的次数
- list.extend(obj) 把序列obj中的内容添加到列表中
- list.index(obj,i=0,j=len(list)) 返回list[k]
- list.insert(index,obj) 在index位置插入对象obj
- list.pop(index=-1) 删除并返回指定位置的对象，默认是最后一个对象
- list.remove(obj) 从列表中删除对象obj
- list.reversed()
- list.sort()

四. Python的基础

Python的数据结构

(B) **Tuple**与List类似，不同之处在于Tuple的元素不能修改。
Tuple使用小元组括号，List使用方括号。
Tuple创建很简单，只需要在括号中添加元素，并使用逗号隔开。

样例：

```
tup1 = ('Google', 'Runoob', 1997, 2000)
tup2 = (1, 2, 3, 4, 5)
tup3 = "a", "b", "c", "d";  # 不需要括号也可以
```

注意：

1、 Tuple中的元素值是不允许修改的，但可对Tuple进行连接；
2、 Tuple中的元素值是不允许删除的，但可使用del语句删除整个Tuple

连接：tup3 = tup1 + tup2
删除：del tup

四. Python的基础

Python的数据结构

(C) **Dict**是另一种可变容器模型，且可存储任意类型对象。
Dict的每个键值(key:value)对用冒号(**:**)分割，每个对之间用逗号(**,**)分割，整个Dict包括在花括号(**{}**)中。
key必须是唯一的，但value则不必。
value可以取任何数据类型，但key必须是不可变的，如字符串，数字或元组。

样例：

```
d = {key1 : valu1, key2 : value2 }
dict = {'Alice': '2341', 'Beth': '9102', 'Cecil': '3258'}
dict1 = { 'abc': 456 }
dict2 = { 'abc': 123, 98.6: 37 }
```

四. Python的基础

Python的数据结构

(C) **Dict**

读取：相应的key放入到方括号中，如：dict['Name‘]

修改：向Dict添加新内容的方法是增加新的key/value对，修改或删除已有key/value。如:

```
dict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8              #菜鸟教程更新 Age
dict['School'] = “CMU"     # 添加信息
```

删除：能删单一的key/value对也能清空Dict，清空只需clear()操作。删除一个字典用del命令，如：

```
dict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First’}
del dict['Name’]   # 删除键 'Name'
dict.clear()       # 清空字典
del dict           # 删除字典
```

四. Python的基础

Python的条件控制

if语句的一般形式如下所示

```
if condition_1:
    statement_block_1
elif condition_2:
    statement_block_2
else:
    statement_block_3
```

实例演示了狗的年龄计算判断：

程序：

```
age = int(input("请输入你家狗狗的年龄: "))
print("")
if age < 0:
    print("你是在逗我吧!")
elif age == 1:
    print("相当于 14 岁的人。")
elif age == 2:
    print("相当于 22 岁的人。")
elif age > 2:
    human = 22 + (age -2)*5
    print("对应人类年龄: ", human)
```

输出：

```
请输入你家狗狗的年龄: 1

相当于 14 岁的人。
```

```
请输入你家狗狗的年龄: 3

对应人类年龄:  27
```

四. Python的基础

Python的循环语句

循环语句有 **for** 和 **while**

for语句的一般形式如下所示

```
for <variable> in <sequence>:
    <statements>
else:
    <statements>
```

while语句的一般形式如下所示

```
while 判断条件:
    语句
```

实例演示了狗的年龄计算判断：

程序：

```
languages = ["C", "C++", "Perl", "Python"]
for x in languages:
    print (x)
```

```
C
C++
Perl
Python
```

```
n = 100

sum = 0
counter = 1
while counter <= n:
    sum = sum + counter
    counter += 1

print("1 到 %d 之和为: %d" % (n,sum))
```

```
1 到 100 之和为: 5050
```

四. Python的基础

Python的函数

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 ()。

函数的一般形式如下所示

```
def 函数名（参数列表）：
    函数体
```

实例演示了狗的年龄计算判断：

程序：

```
# 计算面积函数
def area(width, height):
    return width * height

def print_welcome(name):
    print("Welcome", name)

print_welcome("Runoob")
w = 4
h = 5
print("width =", w, " height =", h, " area =", area(w, h))
```

输出：

```
Welcome Runoob
width = 4  height = 5  area = 20
```

四. Python的基础

Python的模块

（A）模块是一个包含所有定义的函数和变量的文件，其后缀名是.py，实现代码重用。

（B）模块可以被别的程序引入，以使用该模块中的函数等功能。这也是使用 python 标准库的方法。

（C）一个模块只会被导入一次，不管你执行了多少次import。这样可以防止导入模块被一遍又一遍地执行。

四. Python的基础

Python的模块

import 与 from...import

在 python 用 import 或者 from...import 来导入相应的模块。

将整个模块(somemodule)导入，格式为： import somemodule

从某个模块中导入某个函数,格式为： from somemodule import somefunction

从某个模块中导入多个函数,格式为： from somemodule import firstfunc, secondfunc

将某个模块中的全部函数导入，格式为： from somemodule import *

```
import os
import numpy as np

files = os.listdir('./')
arr_zero = np.zeros((3, 2), dtype=np.int8)

print(files)
print(arr_zero)
```

输出：

```
['.idea', 'python_base.py']
[[0 0]
 [0 0]
 [0 0]]
```

五. Tensorflow的安装

（A）WIM10以管理员身份启动cmd，这里安装CPU版本tensorflow 1.6.0，则运行:

pip install TensorFlow==1.6.0

（B）中间需要确认安装其他的依赖库时，输入‘y’选择安装。

（C）安装完成后，验证。在cmd里输入python，
再输入import tensorflow as tf，
最后tf.__version__，注意是左右两个_。
成功打印tf的版本则安装成功。

五. Tensorflow的安装

（A）pip install TensorFlow==1.6.0

```
C:\Users\Sam>pip install tensorflow==1.6.0
Collecting tensorflow==1.6.0
  Downloading https://files.pythonhosted.org/packages/b1/60/0367a766d2936ed207037ac8a94238c7f40a41d90de50d904ed1f8bd51cc/tensorflow-1.6.0-cp35-cp35m-win_amd64.whl (32.3MB)
    21% |████████                        | 6.8MB 2.8MB/s eta 0:00:10
```

（B）安装完成时：

```
C:\Users\Sam>pip install tensorflow==1.6.0
Collecting tensorflow==1.6.0
  Downloading https://files.pythonhosted.org/packages/b1/60/0367a766d2936ed207037ac8a94238c7f40a41d90de50d904ed1f8bd51cc/tensorflow-1.6.0-cp35-cp35m-win_amd64.whl (32.
    100% |████████████████████████████████| 32.3MB 305kB/s
Requirement already satisfied: wheel>=0.26 in d:\python35\lib\site-packages (from tensorflow==1.6.0) (0.31.1)
Requirement already satisfied: numpy>=1.13.3 in d:\python35\lib\site-packages (from tensorflow==1.6.0) (1.14.4)
Requirement already satisfied: astor>=0.6.0 in d:\python35\lib\site-packages (from tensorflow==1.6.0) (0.6.2)
Requirement already satisfied: termcolor>=1.1.0 in d:\python35\lib\site-packages (from tensorflow==1.6.0) (1.1.0)
Requirement already satisfied: protobuf>=3.4.0 in d:\python35\lib\site-packages (from tensorflow==1.6.0) (3.5.2.post1)
Requirement already satisfied: grpcio>=1.8.6 in d:\python35\lib\site-packages (from tensorflow==1.6.0) (1.12.1)
Collecting tensorboard<1.7.0,>=1.6.0 (from tensorflow==1.6.0)
  Downloading https://files.pythonhosted.org/packages/b0/67/a8c91665987d359211dcdca5c8b2a7c1e0876eb0702a4383c1e4ff76228d/tensorboard-1.6.0-py3-none-any.whl (3.0MB)
    100% |████████████████████████████████| 3.1MB 1.6MB/s
Requirement already satisfied: six>=1.10.0 in d:\python35\lib\site-packages (from tensorflow==1.6.0) (1.11.0)
Requirement already satisfied: gast>=0.2.0 in d:\python35\lib\site-packages (from tensorflow==1.6.0) (0.2.0)
Requirement already satisfied: absl-py>=0.1.6 in d:\python35\lib\site-packages (from tensorflow==1.6.0) (0.2.2)
Requirement already satisfied: setuptools in d:\python35\lib\site-packages (from protobuf>=3.4.0->tensorflow==1.6.0) (28.8.0)
Requirement already satisfied: werkzeug>=0.11.10 in d:\python35\lib\site-packages (from tensorboard<1.7.0,>=1.6.0->tensorflow==1.6.0) (0.14.1)
Requirement already satisfied: markdown>=2.6.8 in d:\python35\lib\site-packages (from tensorboard<1.7.0,>=1.6.0->tensorflow==1.6.0) (2.6.11)
Requirement already satisfied: bleach==1.5.0 in d:\python35\lib\site-packages (from tensorboard<1.7.0,>=1.6.0->tensorflow==1.6.0) (1.5.0)
Requirement already satisfied: html5lib==0.9999999 in d:\python35\lib\site-packages (from tensorboard<1.7.0,>=1.6.0->tensorflow==1.6.0) (0.9999999)
Installing collected packages: tensorboard, tensorflow
  Found existing installation: tensorboard 1.8.0
    Uninstalling tensorboard-1.8.0:
      Successfully uninstalled tensorboard-1.8.0
Successfully installed tensorboard-1.6.0 tensorflow-1.6.0
```

总结

一、介绍为什么现在Python这么火
二、首先介绍了python的安装
三、介绍PyCharm的安装，便于Python的编程
四、简要介绍Python的基本语法等
五、介绍Tensorflow的CPU版本安装

谢谢聆听

参考网页：

为什么用Python：https://zhidao.baidu.com/question/8140568224664249 32.html
Python基础：http://blog.az009.com/14106.html
Python教程：https://www.runoob.com/python3/python3-tuple.html