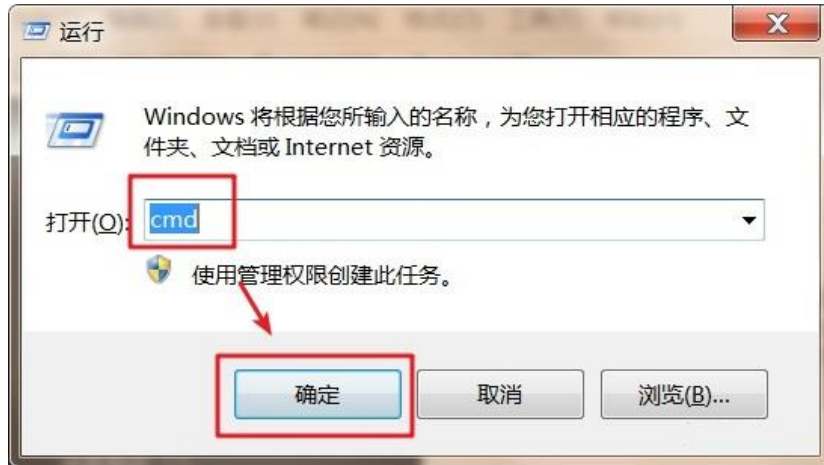


实验一 利用 scikit-learn 实现 K-均值聚类

准备工作:

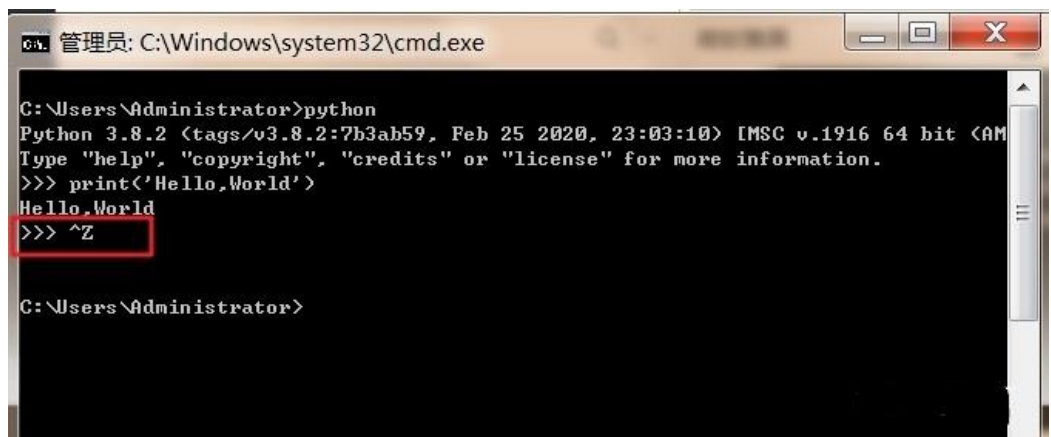
1. 在计算机上安装好 python (推荐 3.x 版本)
 - (1) 首先检查计算机上是否已经安装了 python



进入命令行, 输入 “python”, 出现如下的界面则表示安装成功



ctrl+Z, 回车即退出 python 环境



- (2) 否则, 需要新安装 python, 从官网下载安装包:

<https://www.python.org/downloads/windows/>

选择跟计算机操作系统以及 32 位/64 位相匹配的版本 (.exe 文件)



- (3) 测试一下 pip 有没有安装好, pip 是用来安装第三方库的神器, 这个我们接下来会用到:

在命令行窗口中, 退出了 python 环境后, 输入 pip 回车, 如果出现了一长串命令指南, 说明我们的 pip 也安装好了

```
C:\Users\Administrator>pip

Usage:
  pip <command> [options]

Commands:
  install          Install packages.
  download         Download packages.
  uninstall        Uninstall packages.
  freeze           Output installed packages in requirements format.
  list             List installed packages.
  show             Show information about installed packages.
  check            Verify installed packages have compatible dependencies.
  config           Manage local and global configuration.
  search           Search PyPI for packages.
  wheel            Build wheels from your requirements.
  hash             Compute hashes of package archives.
  completion       A helper command used for command completion.
  debug           Show information useful for debugging.
  help            Show help for commands.

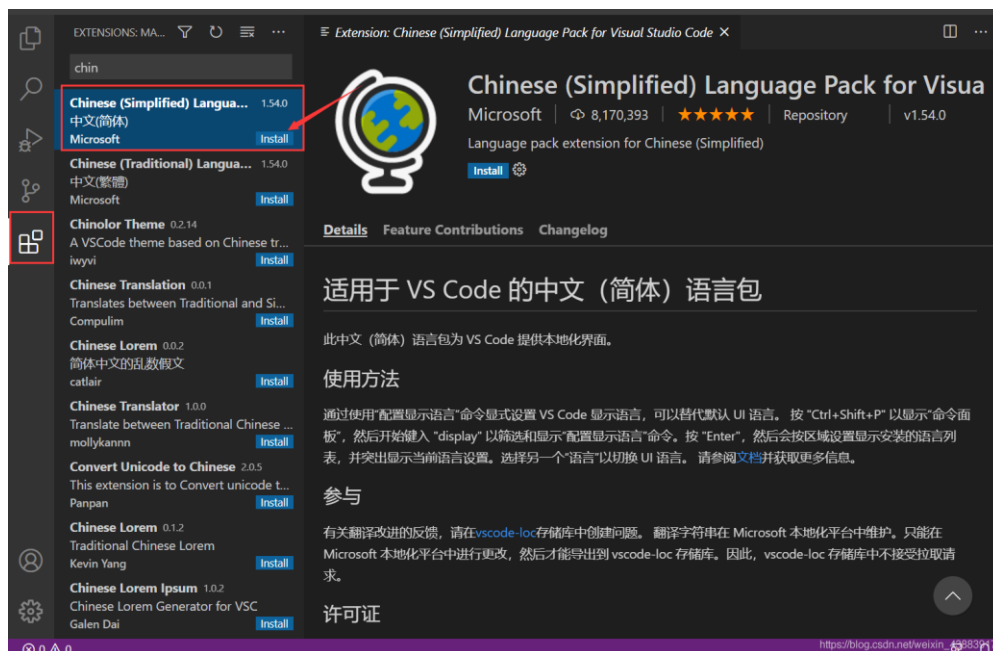
General Options:
  -h, --help       Show help.
```

2. 安装编译器和编译环境

- (1) 机房的计算机上应该已经安装好了 Visual Studio Code, 否则从官网下载安装包进行安装 (过程略):

<https://code.visualstudio.com/download>

- (2) 如果不是中文界面, 搜索 Chinese 语言包安装, 重启后即可



- (3) 在 VS Code 中安装 python 插件:

点击 VS Code 左侧的扩展图标, 在输入框中输入 “python”, 选择第一个带星号的点击安装



(4) 测试一个简单的 python 程序：

在 VS Code 中打开文件夹后，点击下图中的按钮新建文件，在弹出的框中输入文件的名字（后缀为.py）



输入以下内容并保存文件：

```
print ("Hello World!")
```

运行查看输出结果是否正常

3. 安装 scikit-learn 工具包

(1) 打开命令行窗口

(2) 用 pip 命令安装依赖库 numpy, scipy 和 pandas:

```
pip install numpy
pip install scipy
pip install pandas
```

(3) 用 pip 命令安装 scikit-learn 库：

```
pip install scikit-learn
```

(4) 用 pip list 命令查看是否正确安装

利用 scikit-learn 实现 K-均值聚类：

1. 了解 scikit-learn 工具包

英文主页：<https://scikit-learn.org/stable/>

中文社区：<https://scikit-learn.org.cn/>

scikit-learn

Machine Learning in Python

入门 0.23版本的发布要点 GitHub

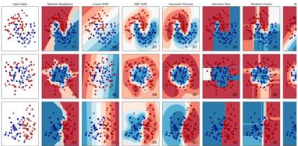
- 简单有效的工具进行预测数据分析
- 每个人都可以访问，并且可以在各种情况下重用
- 基于NumPy, SciPy和matplotlib构建
- 开源，可商业使用-BSD许可证

分类

标识对象所属的类别。

应用范围：垃圾邮件检测，图像识别。

算法：SVM 最近邻 随机森林 更多...



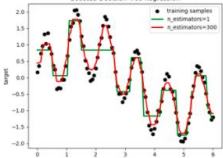
Examples

回归

预测与对象关联的连续值属性。

应用范围：药物反应，股票价格。

算法：SVR 最近邻 随机森林 更多...




Examples

聚类

自动将相似对象归为一组。

应用：客户细分，分组实验成果。

算法：K-均值 谱聚类 MeanShift 更多...



Examples

2. 学习其中的 sklearn.cluster.KMeans 函数方法（输入输出参数、如何使用）

<https://scikit-learn.org.cn/view/383.html>

Prev Up Next

CDA数据科学研究院
提供翻译支持

sklearn.cluster.KMeans

sklearn.cluster.KMeans

```
class sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='deprecated',
```

[\[源码\]](#)

K-均值聚类

在[用户指南](#)中阅读更多内容。

参数	方法
n_clusters	int, default=8 要形成的簇数以及要生成的质点数。
init	{'k-means++', 'random', ndarray, callable}, default='k-means++' 初始化方法： 'k-means++': 明智地选择初始聚类中心进行k均值聚类，加快收敛速度有关详细信息，请参阅k_init中的Notes部分。 'random': 从初始质心的数据中随机选择n_clusters观测(行)。 如果一个ndarray被传递，它应该是形状的(n_clusters, n_features)，并给出初始中心。 如果传递了一个可调函数，它应该接受参数X、n_clusters和一个随机状态，并返回一个初始化。
max_iter	int, default=300 相对容忍度与Frobenius范数，连续两次迭代之间的聚类中心的差异声明收敛。不建议将其设置为tol=0，因为由于舍入错误，可能永远不会声明收敛。用一个很小的数字代替。
	{'auto', True, False}, default='auto' 预计算距离(速度更快，但占用更多内存)。

3. 利用 KMeans 函数实现一个简单的聚类例子

```
>>> from sklearn.cluster import KMeans
>>> import numpy as np
>>> X = np.array([[1, 2], [1, 4], [1, 0],
...               [10, 2], [10, 4], [10, 0]])
>>> kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
>>> kmeans.labels_
array([1, 1, 1, 0, 0, 0], dtype=int32)
>>> kmeans.predict([[0, 0], [12, 3]])
array([1, 0], dtype=int32)
>>> kmeans.cluster_centers_
array([[10.,  2.],
       [ 1.,  2.]])
```

4. 实现第二章作业中的聚类问题

现有样本集 $X=\{(0,0), (0,1), (4,4), (4,5), (5,4), (5,5), (1,0)\}$ ，试用C-均值算法进行聚类分析（ $C=2$ ）。

5. 学习一个复杂一点的聚类示例（如何对聚类结果可视化、聚类中心、不同类别上色）

<https://scikit-learn.org.cn/view/70.html>

K-Means和MiniBatchKMeans聚类算法的比较

我们想比较一下MiniBatchKMeans和KMeans的性能：MiniBatchKMeans更快，但给出的结果略有不同(看 [Mini Batch K-Means](#))

我们将对一组数据进行聚类，首先使用KMeans，然后使用MiniBatchKMeans，然后绘制结果。我们还将绘制两个算法之间标记不同的点。

