第00讲 课程导学 - Python环境的安装和配置

作者：欧新宇（Xinyu OU）

本文档所展示的测试结果，均运行于：Intel Core i7-7700K CPU 4.2GHz

一. 标准Python环境的安装和配置

该安装配置模式适用于仅需要完成《程序设计基础（Python）》和《计算机数学》课程的同学，对于需要学习其他Python和AI相关课程的同学，建议参看 **“2.标准Python环境安装和配置”**。

1. 安装Python环境

a. 访问Python官网并下载最新版Python，URL: https://www.python.org/

b. 双击并运行安装，勾选【Add Python 3.8 to PATH】

2. 安装课程所需要的其他库文件

2.1 更新python底层

```
>> pip install --upgrade pip
```

2.2 安装第三方库

- numpy库

```
>> pip install numpy
```

- scipy库

```
>> pip install scipy
```

3. Python环境测试

方法一：打开IDLE交互环境，并执行下列指令进行测试

方法二：打开IDLE文件编辑器，并输入下列代码并运行

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

```
Hello World!
```

```
import numpy
import scipy

# import后，只要没有报错信息，就说明安装成功
```

二. 高级Python环境安装和配置

1. Python编程环境的安装与配置

- 推荐安装Anaconda版Python，因为Anaconda包含了大量的Python库函数，安装之后基本就不需要手动安装其他库了。
- URL：https://www.anaconda.com/products/individual
- 安装过程较为简单，按照提示运行安装程序即可

2. Notebook 编程环境的安装与配置

Notebook是Anaconda内置的一种交互式Python开发环境之后，非常适合进行代码的调试。

打开步骤为：`“开始”菜单` -> `Anaconda3(64bit)` -> `Jupyter Notebook(Anaconda3)`

3. JupyterLab 编程环境的安装与配置

JupyterLab是Jupyter最新的客户端软件，也是Jupyter的发展方向。

3.1 JupyterLab的安装

- 打开【Anaconda Prompt (Anaconda3)】并执行以下语句

```
>> conda install jupyterlab
```

- 更新JupyterLab内核

```
>> conda update jupyter_core jupyter_client
```

3.2 JupyterLab的配置 (可选)

默认情况下，打开JupyterLab，需要先打开【Anaconda Prompt (Anaconda3)】，然后在命令行中输入【jupyter lab】。同时，JupyterLab和Notebook一样，工作路径为'C:\Users\Administrator'，为了方便使用，我们可以将该地址修改为指定文件夹。

方法一：修改JupyterLab的默认路径

- 生成配置文件

```
>> jupyter notebook --generate-config
```

- 编辑配置文件
  - 打开【C:\Users\计算机的用户名.jupyter\jupyter_notebook_config.py】
  - 修改字段【c.NotebookApp.notebook_dir】为指定路径
  - 其中【C:\Users\计算机的用户名】为Anaconda的默认路径，可以通过启动【Anaconda Prompt (Anaconda3)】查看默认地址。

方法二：使用快捷方式打开指定文件夹

此处提供一种基于批处理(*.bat)的快速打开JupyterLab的方法。

a. 新建一个文本文件，输入以下字段

```
C:\ProgramData\Anaconda3\Scripts\jupyter-lab.exe
D:\CloudStation\MyWebsites\Teaching\ComputerMath\
```

以上JupyterLab的路径也可能会存在于User文件夹下，请根据本机的配置和路径，参照修改。

b. 将该文件另存为JupyterLab.bat（或其他名字.bat，建议保存到桌面方便执行）

c. 使用时，只需要双击该批处理(*.bat)文件即可

4. Visual Studio Code (VSCode) 编程环境的安装与配置

4.1 VSCode的安装

- VSCode是当今最流行的集成开发环境，不仅适用于Python，也同样适用于Html+CSS、Javascript及php等Web前端的开发，同时也支持Java、C++、C等程序的开发。
- 下载最新版本的VSCode，URL：https://code.visualstudio.com/

4.2 VSCode的配置 (可选，更丰富的插件有利于提高开发效率)

VSCode的强大之处来源于各种插件，下面将推荐一些常用及好用的插件。安装时，首先打开左边的【Extensions】标签（4个小方块），或按快捷键【Ctrl+Shift+X】打开插件管理界面。输入插件名称，并点击插件旁的【Install】按钮。

- **自动同步配置**
  - 插件名：【Settings Sync】
  - 使用方法：【Alt+Shift+U/D】（上传/下载）
  - Settings Sync插件的使用，需要配合Github使用，此处不再累赘介绍该插件的安装方法，各位可以自行【百度】。
  - 配置好【Settings Sync】后，后续的所有安装和配置，基本上就只需要执行一次，将来可以直接使用【Settings Sync】进行同步。同时该插件也可以实现多台计算机配置（及所有插件）的同步。
- **中文界面**
  - 使用快捷键【Ctrl+Shift+P】打开搜索按钮
  - 搜索【Configure Display Language】，选择安装简体中文或直接搜索插件【Chinese (Simplified) Language Pack for Visual Studio Code】
  - 安装完成后按照提示重启VSCode，或手动重启VSCode以激活简体中文语言包
- **启用Flake8代码检测**
  - 点击左下角【Setting】按钮，并搜索【Flake】
  - 勾选【Python › Linting: Flake8 Enabled】
- **有用的插件**
  - Python开发包：Python (Microsoft)
  - 突出显示成对的括号：Bracket Pair Colorizer 2+++
  - 突出显示缩进：indent-rainbow、 Guides
  - 漂亮的文件夹工具包：vscode-icons
  - 安装方法：点击左侧【Extensions】
- **有用的命令**
  - 自动格式化代码：【Alt+Shift+F】
- **其他问题**
  - 在Python调用第三方库时，pylint无法完成语法检测。
  - 解决方法是：打开设置，搜索并编辑【settings.json】；在大括号里增加：【“python.linting.pylintArgs”:[“–generate-members”]】

三. 在线编程环境

1. Python123课程平台

该网站是由北京理工大学开发的Python课程平台，我们可以利用该平台的在线编程环境完成本课程的代码编写。网址:https://python123.io/

2. Notebook 在线平台

该平台为公开免费的Jupyterlab开发平台，可用于调试程序。该平台无法持久化保存程序（每天清空），但可将Notebook导出为 html 格式。

平台地址：http://115.28.150.200:8000/ 。
用户名：输入你的姓名
密码： pygis

四. 线性代数必需库的安装和测试

本课程的代码需要一些基于Python的第三方库的支持，主要包括Numpy和Scipy。使用基于Anaconda开发包安装的Python，这两个库都不需要再额外安装，基于官方版的Python需要额外进行安装，前面已经提供了安装方法。

1. Numpy 基础科学计算库

Numpy是Python中最基础的科学计算库，它的功能主要包括高位数组（Array）计算、线性代数计算、傅里叶变换以及产生伪随机数等。Numpy是机器学习库scikit-learn的重要组成部分，因为机器学习库scikit-learn主要依赖于数组形式的数据来进行处理。

更多信息请参考：RUNOOB站的Numpy栏目：https://www.runoob.com/numpy/numpy-tutorial.html

【知识点】Numpy基础科学库极简使用说明

以下代码用于测试和生成一个数组。

```
# 使用import关键字引入numpy库，为了简便使用缩写 “np”来表示numpy库。
import numpy as np

# 定义一个变量 i，用于保存数组
i = np.array([[12,34,56],[78,90,11]])

# 输出变量 i
print("i = \n{}".format(i))
```

```
i =
[[12 34 56]
 [78 90 11]]
```

2. Scipy 科学计算工具集

Scipy是Python中用于进行科学计算的工具集，它可以实现计算机统计学分布、信号处理、线性代数方程等功能。在机器学习中，稀疏矩阵的使用非常频繁，Scipy库中的sparse函数可以用来生成这种稀疏矩阵。稀疏矩阵用于存储那些大部分数值为0的np数组。以下代码用使用sparse()函数生成和测试稀疏矩阵。

```python
# 对scipy的使用需要利用from关键字来引用其内部的子库
import numpy as np
from scipy import sparse

# 使用numpy的eye()函数生成一个6行6列的对角矩阵
# 矩阵中对角线上的元素值为 1，其余元素为 0
matrix = np.eye(6)

# 将np数组转化为 CSR格式的Scipy稀疏矩阵 (sparse matrix)
sparse_matrix = sparse.csr_matrix(matrix)
```

```python
# 输出对角矩阵
print("对角矩阵：\n{}".format(matrix))
```

```
对角矩阵：
[[1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1.]]
```

```python
# 输出CSR格式的稀疏矩阵
print("CSR格式的稀疏矩阵：\n{}".format(sparse_matrix))
```

```
CSR格式的稀疏矩阵：
  (0, 0)    1.0
  (1, 1)    1.0
  (2, 2)    1.0
  (3, 3)    1.0
  (4, 4)    1.0
  (5, 5)    1.0
```