本文档给出了《基于 Python 的科学与数值计算》课程的 Python 环境设置及相关软件的简单使用说明，供上课同学参考。

1. Python 安装与配置

1.1. 不同的配置方案

Python 的应用广泛，很多操作系统自身就带有 Python 程序，但要打造一个适合自己使用的 Python 环境，需要学会如何安装 Python、配置环境、安装库。

Python环境的配置方案包括：

- 系统自带（Linux，Mac，Win10-WSL），版本一般不够新。可以用 pip 安装新的包。
- 安装包 from python.org，只包括 Python 程序和标准库。可以用 pip 安装新的包。
- 推荐：anaconda
  - 包含了常用的数学、科学库，numpy/scipy/matplotlib
  - 方便进行包/库的安装和管理（conda 和 pip）
  - 标准版（ananconda）和 mini 版（Miniconda，体积小，安装快，带的包小）
  - 方便配置不同的环境（Envs）

环境 Env：有时也称作虚拟环境，指的是 Python 程序及其相应的库/包的组合。由于 Python 的库非常多，非常灵活，不同的库/包之间可能有不兼容的问题（版本不匹配）。经常为了使用某个特定版本的库，可以为其单独安装一个 Python 环境，在其中可以指定 Python 版本、库的版本。不同的环境是相互隔离的（沙盒），为测试、试验提供了方便。

1.2. pip: 包管理

pip 是 Python 「官方」的库/包的管理命令，从 PyPI （https://pypi.org）搜索安装。

pip 随 Python 程序一起安装。使用时要注意当前的 pip 属于哪一个 Python 环境。pip 是一个命令行工具，需要打开终端（Windows 下的 cmd、Powershell，Linux/Mac 下的 terminal），运行 pip。

- 安装包：pip install xxxx
- 升级包：pip install -U xxxx
- 删除包：pip uninstall xxxx
- 查看已安装包：pip freeze，pip list
- 用 poetry search xxxx （先 pip install poetry 安装。2020年后 pip search功能无法使用）
- 升级所有已安装包（Linux/Mac 下适用）：

```
pip freeze --local | grep -v '^\-e' | cut -d = -f 1 | xargs -n1
pip install -U
```

1.3. Anaconda

Anaconda 是一个用于数据科学/数值计算/深度学习的 Python 及其相关库的集成软件。可以方便的配置/维护 Python环境，支持 Windows、Linux、Mac。

1.3.1. 安装 anaconda

从 https://www.anaconda.com 下载相应操作系统/版本的安装包安装（ https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/ 提供了更快速的镜像）。标准包包含了常用的库（如 Numpy、Scipy、Matplotlib、Sympy、jupyter-notnebook 等）；mini包（miniconda）只包含了 Python 程序和最小的库，下载/安装速度快，适用于安装、定制较小系统的情况。

1.3.2. 配置 anaconda

配置 anaconda 的命令是 conda，这是一个命令行工具，需要打开终端（Windows 下的 cmd、Powershell，Linux/Mac 下的 terminal），运行 conda 命令。

- 升级系统：conda update conda; conda update anaconda
- 搜索包：conda search xxx
- 安装包：conda install xxx
- 删除包：conda remove xxx
- 查看已安装包：conda list
- 帮助：conda -h
- 设置使用清华 tuna 镜像（速度快）：

```
conda config --add channels
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
conda config --set show_channel_urls yes # 设置搜索时显示通道地址
```

修改源的另外一种方式是修改配置文件，配置文件一般在用户目录下的 .condarc 文件，用文本编辑器打开，修改其中的 channels 部分。具体可参考：https://mirrors.tuna.tsinghua.edu.cn/help/anaconda/。

1.4. 虚拟环境

虚拟环境（Virtual environment）是一个沙盒，安全隔离的 Python 运行/开发 环境，包括特定版本的 Python 和相应的库，可用于搭建 测试或特定的 Python 环境。

环境可以用 Python 自带的 venv 工具创建和管理；也可以用 anaconda 的 conda 命令创建管理。更高级的可以用 Docker 创建 OS 级的环境。

1.4.1. 用 Python 自带的 venv 工具创建、管理

```
python -m venv test-env  # 创建 test-env 的环境，创建目录 test-env，该环境下安装的库都在其中
test-env\Scripts\activate.bat # Windows 上激活该环境
source test-env/bin/activate # Unix或MacOS上激活该环境
pip install numpy
deactivate  # 退出当前 venv 环境
```

1.4.2. 用 anaconda 创建管理

如果安装的是 anaconda ， 建议使用 conda 管理 Python 环境。

```
conda create --name test python=3.10 # 创建名为 test 的环境，使用 3.10 版本的 Python
conda activate test # 激活 test 环境
conda deactivate # 退出当前环境
conda remove --name test --all # 删除 test 环境
conda info --envs # 列出当前可用的环境，还可以用 conda env list 命令查看
```

1.5. 搜索路径

Python 的一大优势是有庞大数量的库。用户可以方便的安装使用各种库，包括官方的、网络中开源的或者自己/团队开发的。但是在使用时，所安装的库/包/模块 必须在python 的搜索路径内，才能够使用（import）。

通过pip、conda 安装的包自动加入到默认的路径中，可以直接使用，不需要添加搜索路径。

而手动安装的库（包括自己开发的库），需要明确告知 Python 其所在的位置（路径）。

1.5.1. Python 内查看/管理路径

如果是临时修改路径，可以直接用 Python 的 sys 库（与系统操作有关的操作）进行。

```
import sys  # 导入 sys 库
sys.path  # 显示当前的搜索路径
sys.path.append('/Users/xxh/Work/Carnot') # 将 /Users/xxh/Work/Carnot 添加到搜索路径中
```

1.5.2. 为系统添加路径

如果想一次添加、持续使用，则添加到系统路径中。

1.5.2.1. Linux/Mac

Linux/Mac 系统下，用环境变量PYTHONPATH 存储 Python 的搜索路径。该变量在所用的 shell 配置文件（shell profile）中定义，取决于所采用的 shell，如 bash 的profile 为用户目录下的 .bashrc ， zsh则为 .zshrc。

以 bash 为例，添加某个路径的命令如下：

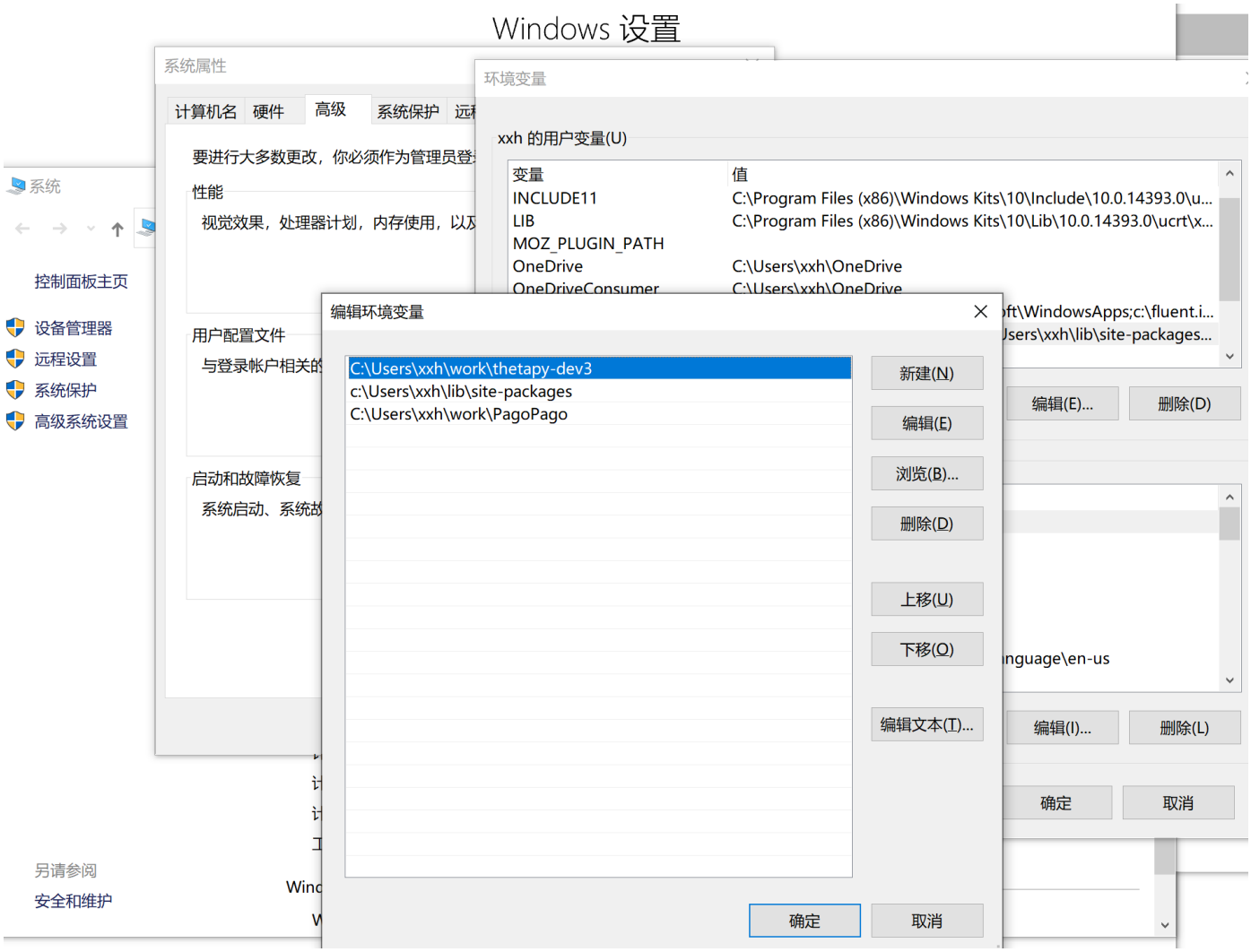
```
echo "export PYTHONPATH=/path/to/module" >> ~/.bashrc
source ~/.bashrc
```

我的 Mac 上所用的shell 为 zsh，对应的 .zshrc 中 PYTHONPATH 语句为：

```
export
PYTHONPATH="/Users/xxh/Work/thetapy":"/Users/xxh/Work/Carnot"
```

- Windows

Windows 下的 Python 路径也是用 PYTHONPATH 这一环境变量所定义。设置该变量的路径为：系统属性（或控制面板）/高级系统设置/高级/环境变量，在其中添加一个变量 PYTHONPATH，再添加所需的路径。



2. 集成开发与编辑器

2.1. 概述

Python 程序有不同的开发和运行方式：

- 传统：编辑器/IDE 编写程序代码，在终端运行
- 现代：Jupyter notebook 开发+运行一体化
- 即时使用：交互式
- 其他：web、远程

其中，

- 开发 Python库或较大的程序，推荐用编辑器编辑，便于保存和版本管理。
- 如果做数据的处理、进行某个问题的研究、需要绘制图像/曲线，推荐采用 Jupyter notebook/lab 环境。
- 简答的测试语句、函数功能等，可以打开一个 python console。比如我的一个习惯是保持打开一个 jupyter-qtconsole（可以用 conda /pip 安装，终端下执行 jupyter-qtconsole 启动）。

支持 Python 编程的编辑器/IDE 有：

- VS Code (推荐)
- PyCharm
- Sublime Text
- Spyder（与 Matlab 环境类似）
- vim/emacs

2.2. Jupyter notebook 和 Jupyter lab

Jupyter 是一个基于浏览器的集成 Python 开发/运行环境。它的基本文件类型叫做 notebook，其中集成了代码编辑、运行，Markdown 文本，公式、图片。在其中可以编写并运行代码（包括 Python 代码，还支持 R、Julia 等其他语言），并将运行结果，包括图片直接插入当前文档，形成一个可复现文件。

notebook 非常适合于做「研究性」的编程。当研究一个问题是，可以把问题的描述、数学模型等用 Markdown 格式记录在 notebook 中，然后进行编程实现对问题的求解或分析，再将程序的运行结果、曲线、图表等自动插入到 notebook 中，形成一个初步的研究报告。notebook 的一个重要优点是：问题的描述、模型、代码、结果是不分离的，可以在任何时间回到这个问题，快速的修改模型、程序，改进研究。另外的一个优点是，该 notebook 是可复现的，别人拿到该文件，可以很容易的复现其中的结果。

2.2.1. 启动

使用 notebook 的直接方式是启动一个 jupyter-notebook 服务，然后在浏览器中访问 http://localhost:8888。jupyter notebook 需要在终端中启动（如Windows 下的 cmd、powershell，或 linux 下的 terminal），启动后一般会自动打开默认浏览器访问 http://localhost:8888。

Jupyter lab 是 jupyter notebook 的升级，提供更多的功能（如独立的代码编辑器、终端等）。启动方式类似，在终端中运行 jupyter-lab。

如果想在服务器上为多人提供 notebook 使用，可以用 jupyter hub。

jupyter notebook 和 lab 都可以对外提供服务（远程使用），启动时加上 --ip=xx.xx.xx.xx，就可以在其他联网的设备（电脑、手机、pad）上访问，从而使用 notebook，此时的 notebook 存储在服务器上，Python 及硬件资源都是服务器提供的。

提供远程服务的端口、密码等设置在 jupyter 的配置文件中修改（配置文件一般在用户目录下的 .jupyter/jupyter_notebook_config.py，用 jupyter notebook -config 生成）。具体的配置修改可以在网络上搜索查询。

相关命令汇总：

```
jupyter-notebook   # 启动 jupyter notebook
jupyter-lab        # 启动 jupyter lab
jupyter-lab --ip=0.0.0.0  # 启动对外服务的 jupyter lab
jupyter notebook -config # 生成 notebook 的配置文件
jupyter_notebook_config.py
jupyter lab -config # 生成 lab 的配置文件 jupyter_lab_config.py
```

2.2.2. 使用

notebook 可以在浏览器（IE、Chrome、Safari等）中使用，也可以在 VS Code 中使用。

notebook 的基本单元是 cell，cell 可以是 Markdown 文本、Python 代码或原始文本（raw），按 Shift+Enter 执行当前 cell。

2.2.3. magic commands

magic commands: 用于 IPython/Jupyter 环境的魔法命令（不属于 Python 语言），以 `%`开头。

```
%cd chp1  # 和命令终端下的 cd 相同，改变当前的路径
%pwd  # 显示当前路径
%conda install xxx  # 运行 conda 命令
%lsmagic  # 列出当前可用的 magic 命令
%matplotlib inline # 让 matplotlib 输出图像到当前页面（嵌入）
%run xxx # 运行 xxx.py
%sx xxx # 执行 shell 程序 xxx，或 !xxx
!ls
!!ls
%time xxx # 运行 xxx 并统计时间
%timeit xxx # 统计xxx的平均运行时间
```

2.2.4. 快捷键

2.2.4.1. 命令模式（按 Esc 键进入）

1. Y：单元转入代码状态
2. M :单元转入markdown状态
3. R：单元转入raw状态
4. A：在上方插入新单元
5. B：在下方插入新单元
6. X：剪切选中的单元
7. C：复制选中的单元
8. V /Shift-V：粘贴到下/上方单元
9. D,D：删除选中的单元
0. Shift-M：合并选中的单元
1. I,I：中断Notebook内核
2. 0,0：重启Notebook内核

2.2.4.2. 编辑模式（按Enter 转入）

1. Tab：代码补全或缩进
2. Shift-Tab：提示
3. Ctrl-]：缩进
4. Ctrl-[：解除缩进
5. Ctrl-A：全选

6. Ctrl-Z：复原

2.2.4.3. 基本快捷键（任何模式都可以）

1. Shift-Enter：运行并选中下单元
2. Ctrl-Enter：运行本单元
3. Alt-Enter：运行并下插新单元
4. Ctrl-S：文件存盘

2.3. VS Code

全称 Visual Studio Code，是 Microsoft 基于 Electron 开发的一款免费、通用、功能强大的编辑器，支持几乎所有的编程语言和标记语言的编辑，有庞大、灵活的插件支持。

安装：从 https://code.visualstudio.com 下载对应操作系统的版本，安装。

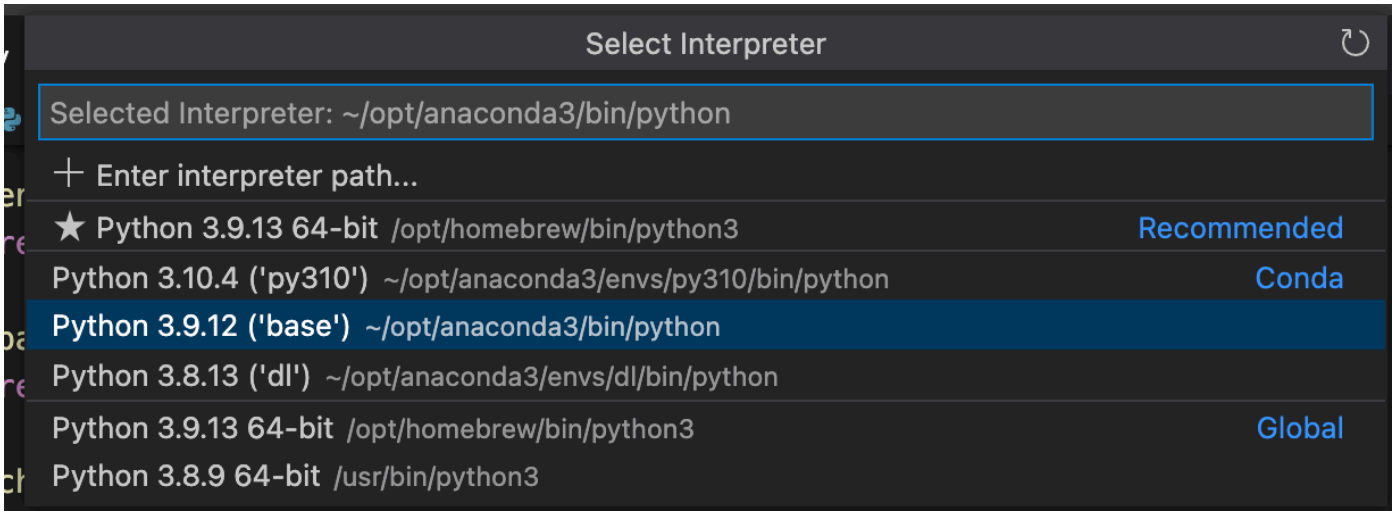
2.3.1. 设置

功能键：Ctrl+Shift+p，Mac 下是 Command+Shift+P。该键呼出命令面板 Command Palette，即位于顶部的命令框，用于执行命令。

2.3.1.1. 选择 Python 解释器

当安装有多个 Python环境时（系统自带、Anaonda 等），需要指定所用的 Python 解释器。

方法：功能键 Ctrl+Shift+p 呼出命令面板，输入命令 select interpreter，然后选择所用的 Python解释器。



2.3.1.2. 首选项

用于修改全局设置。可以通过 Ctrl+Shift+p 呼出命令面板，输入 settings 搜索出 首选项。然后进行设置。

2.3.2. 扩展（Extensions）

VS Code 的强大之处在于庞大数量的扩展。

推荐扩展：

- Python：Python extension for Visual Studio Code
- IntelliCode：智能代码补全
- Python indent：正确缩进和格式
- Jupyter：在 VS Code 中使用 Jupyter notebook
- Markdown all in one：Markdown 支持
- copilot：AI 代码助手。需要申请账号

安装 Jupyter 扩展后，在 VS Code 中打开 notebook 文件（后缀 .ipynb），就启动了 Jupyter notebook，可以直接运行其中的代码块，也可以在其中编辑代码块，运行后的结果会直接显示在 notebook 中，与浏览器中运行 Jupyter notebook 是相同的。

3. 相关的软件与提示

3.1. Markdown

Markdown是一种轻量级标记语言，排版语法简洁，让人们更多地关注内容本身而非排版。它使用易读易写的纯文本格式编写文档，可与HTML混编，可方便的导出 HTML、PDF、LaTeX、Docx 等各种格式文本。因简洁、高效、易读、易写，Markdown被大量使用，如Github、Wikipedia、简书等，Jupyter notebook 也支持 Markdown。

由于 Markdown 简洁、易于转换的特效，它也适用于学术写作，可参见另外一篇文档《Markdown 用于学术写作》。

本文以及本课程的课件都由 Markdown 格式写作。

Markdown 只是一种标记语言（或者说是一种文本格式），可以用任何一种文本编辑器进行编辑。但是现在有很多编辑器或专门的 Markdown 写作软件，可以更高效的进行 Markdown 写作。推荐的编辑器（软件）有：

- Typora：一个轻量化的、所见即所得的 Markdown 写作软件，直接支持 LaTeX 公式。现在为收费软件。
- Marktext：和 Typora 很类似，开源免费。
- Notable
- VS Code：通过扩展支持，也非常方便好用。
- Zettlr

另外，很多开源的笔记软件也是基于 Markdown 的，可以作为编辑器使用：

- Obsidian
- Joplin
- 思源笔记

3.2. LaTeX

LaTeX 是一种广泛用于学术写作的排版系统。大量的学术论文（包括期刊论文和学位论文）都是用 LaTeX 进行写作的。世界上主要的大学都提供了学位论文的 LaTeX 模版，很多有名的期刊也提供 LaTeX 模版。有了模版的协助，作者不必关心论文的格式、排版，只需专注于内容，可以大幅度提高效率。特别是 LaTeX 可以方便快捷的生成格式优美的数学公式。

平时的一般文档写作可能用不到 LaTeX。但是常见的涉及到公式输入的软件（如各种 Markdown软件）、网站（WikiPedia、知乎等）都支持 LaTex 公式的输入。另外，绘制学术图像等软件/程序，如 Matplotlib、Matlab 等，都支持以 LaTeX 格式输入公式和符号，甚至 Word 中也可以用 LaTeX 语法输入公式（提供插件）。

因此，掌握必要的 LaTeX 知识是必要的。对于简单的语法入门，可以从《The Not So Short Introduction to LaTeX 2e》（简称 lshort）开始。

LaTeX 的安装可以下载最新版本的 texlive（https://tug.org/texlive/）。具体的配置和使用可以在网上搜索相关文档。

3.2.1. overleaf

overleaf（https://www.overleaf.com）是一个在线的 LaTeX 编辑器，可以方便的进行 LaTeX 文档的编辑、编译、分享、协作等。它提供了丰富的模版，可以直接使用，也可以下载到本地使用。它还提供了丰富的文档，可以方便的学习和使用 LaTeX。

学校也提供了 overleaf 的服务，访问 https://overleaf.tsinghua.edu.cn ，用自己的学号登录，就可以使用。上面有丰富的模版，包括学位论文模版，大作业模版，开题/答辩汇报模版等。

3.3. git

git 是目前应用最广泛的开源版本控制系统。最大的开源托管网站 github（https://github.com）就是基于 git 打造的。

虽然在大学阶段，可能用不到太多分布式版本控制。但是对于个人的作业、平时积累的代码等也可以在个人电脑上用 git 进行版本管理，从而熟悉版本控制的基本概念和流程，为今后参与多人合作的协作项目打下基础。

如果想更深入的体验，可以去 github 上创建自己的项目，或者克隆别人的项目。学校也提供了托管平台，基于 gitlab 建造。访问 https://git.tsinghua.edu.cn ，用自己的学号登录，就可以上传自己的项目，进行版本管理。本课程的课件，都在 git.tsinghua.edu.cn 上进行管理，地址为 https://git.tsinghua.edu.cn/xxh/psnc ，登录后可访问，克隆命令 `git clone git@git.tsinghua.edu.cn:xxh/psnc.git`，或浏览器访问 https://git.tsinghua.edu.cn/xxh/psnc.git 下载。

关于 git 的具体使用，可以在网上搜索相关文档进行学习。