

引言

Table of Contents

第一章 内容是关于什么的?	3
1.1 经济数据分析	3
1.2 数据分析方法	3
1.3 主要参考书	4
第二章 分析工具: Python	5
2.1 功能强大, 应用广泛	5
2.2 Python 的特点	6
第三章 安装和设置软件环境	7
3.1 下载安装 Python	7
3.2 安装 Jupyter Lab	8
3.3 安装 Visual Studio Code	9
3.4 安装第三方程序包和创建虚拟环境	10
第四章 脚本模式和交互模式	11
4.1 交互式	11
4.2 脚本模式	11
4.3 在 Visual Studio Code 中应用 Jupyter notebook	12
第五章 文档与帮助	15
5.1 help() 和?	15

《基于 Python 的经济分析与应用》旨在将经济学理论与现代数据分析技术相结合，内容涵盖 Python 编程基础、数据采集与处理、经济数据分析、可视化展示及实际案例应用。

通过学习，使学生掌握应用 Python 进行经济数据分析的方法，提高数据处理与决策支持能力，为未来从事数据驱动的经济分析、科学研究或制定经济决策打下坚实基础。

第一章 内容是关于什么的？

1.1 经济数据分析

对经济数据进行分析长期以来都是政策制定、投资者、企业和消费者关注的焦点：

- 宏观经济形势分析。如毕马威的[中国经济观察](#)季度报告等、[中国宏观经济论坛](#)发布的[CMF 中国宏观经济专题报告](#)等。
- 美国经济分析局（Bureau of Economic Analysis, BEA）：负责公布美国宏观经济以及行业的统计数据，以及有关美国国内生产总值（GDP）和各个市/镇/乡/村/县和大都市区的数据；
- [数据科学](#)在人工智能时代的广泛应用；
- 数据服务商的重要作用。如[彭博社](#)、[Wind 资讯](#)等。

1.2 数据分析方法

将数据分析方法应用至经济学、金融学和国际贸易等学科的有关主题。主要包括：

- 经济数据分析：如增长、不平等等、通货膨胀等宏观数据；
- 统计分析方法：t 检验、方差分析等；
- 线性回归方法
- 蒙特卡洛模拟分析
- 机器学习基础方法

- 投入产出模型；
- 网络分析方法；

1.3 主要参考书

会用到部分 Python 有关的内容，如：

- McKinney [1], [在线阅读](#)
- VanderPlas [2], [在线阅读](#)
- [Python Programming for Economics and Finance](#)

第二章 分析工具：Python

我们使用 Python 作为主要的分析工具。根据 [TIOBE Index for August 2025](#), Python 是目前最流行的编程语言。

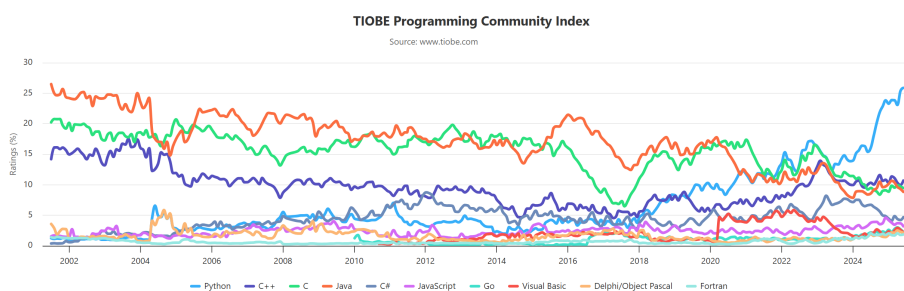


Figure 2.1: TIOBE Programming Community Index

2.1 功能强大，应用广泛

Python 广泛应用于机器学习、科学计算等各个领域：

1. 机器学习
2. 数据科学
3. 通讯
4. 网页开发
5. CGI and GUI
6. 自然语言处理
7. 游戏开发

8. 等等

2.2 Python 的特点

Python 具有许多优点：

1. 易读、易写和易调试；
2. 核心内容易学；
3. 众多库的支持；
4. 初学者友好
5. 支持多平台
6. 网络资源众多

第三章 安装和设置软件环境

3.1 下载安装 Python

- 自[官方网站](#)下载 Python，当前版本 3.13.x。
- 双击打开下载的安装程序，如果是 Windows 操作系统，在点击 “Install Now” 安装程序之前，注意勾选：Add Python to PATH，将 Python 的安装路径添加到操作系统的环境变量 Path 中，如图 Figure 3.1 所示：

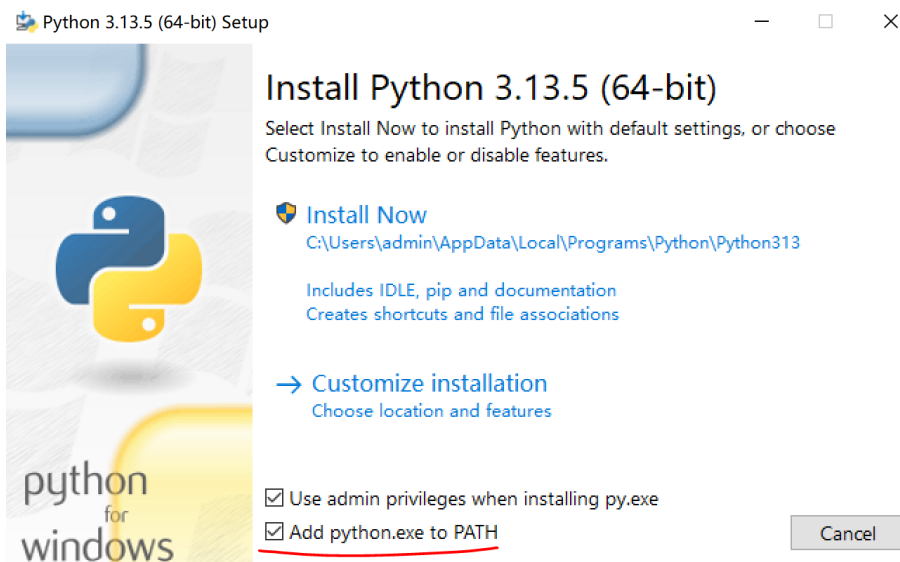
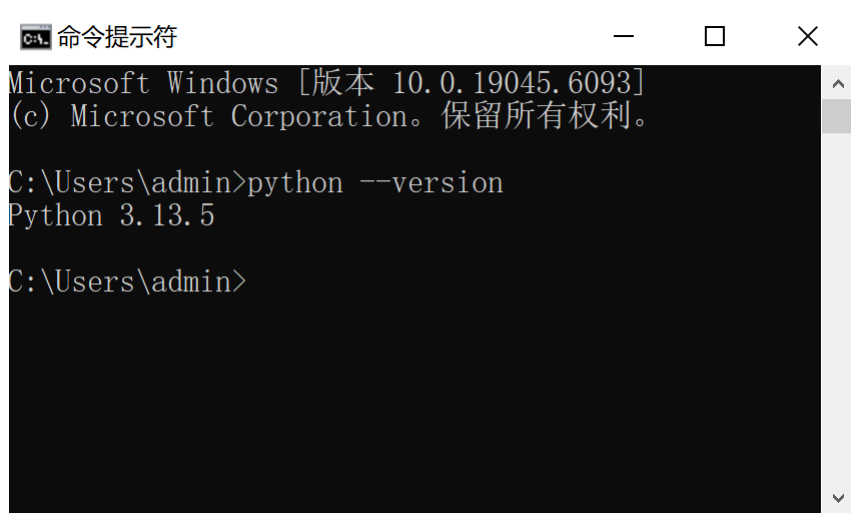


Figure 3.1: 将 Python 添加至环境

- 在“命令提示符”输入 `python --version` 查看安装版本，如图 Fig-

ure 3.2 所示。



```
命令提示符
Microsoft Windows [版本 10.0.19045.6093]
(c) Microsoft Corporation。保留所有权利。

C:\Users\admin>python --version
Python 3.13.5

C:\Users\admin>
```

Figure 3.2: Python 版本

3.2 安装 Jupyter Lab

- 安装 [JupyterLab](#)。通过命令提示符（或 Mac OS 的终端）安装：`pip install jupyterlab`。应用过程中经常需要使用 pip 安装程序，建议将镜像源配置为 [清华大学开源软件镜像站](#)。
- 在命令提示符输入：`jupyter lab`，就可以在浏览器启动 Jupyter Lab，新建一个 Notebook 就可以使用了，如 Figure 3.3 所示。选中单元格（cell），设置为“code”格式（其他两种是 markdown 和 raw），输入：

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

Hello World!

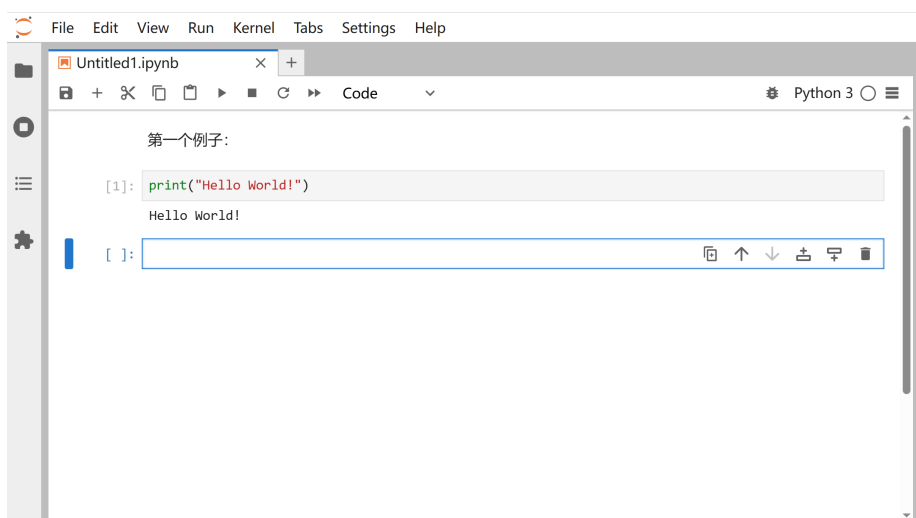


Figure 3.3: Jupyter Lab

3.3 安装 Visual Studio Code

Visual Studio Code 是由微软推出的免费、开源、跨平台的代码编辑器，支持几十种主流编程语言（如 JavaScript、Python、C++、Java、Go 等），并且与微软推出的 Copilot 人工智能工具高度融合，拥有强大的功能和灵活的扩展性。

- 下载安装 [Visual Studio Code](#)。要将软件设置为中文，可以使用快捷键 **Ctrl + Shift + P** 打开命令面板，输入 **Configure Display Language**，在出现的列表中，选择“中文（简体）”，根据提示重启 VS Code，界面语言就会变为中文。
- 在扩展（Extensions Marketplace）搜索安装插件：
 - Python
 - Jupyter,
 - Excel Viewer
 - Rainbow CSV

应用时，Visual Studio Code 可以“打开文件”或者“打开文件夹”将项目所在文件夹处打开。建议以打开文件夹方式，可以比较清楚的概览代码、数

据、图形等子文件夹。

3.4 安装第三方程序包和创建虚拟环境

要安装第三方程序包，基本的方式是通过 `pip` 命令：

```
python -m pip install SomePackage
```

例如，在命令提示符，或者在 Visual Studio Code 使用快捷键 `Ctrl + Shift + \` 新建终端，输入命令使用 `pip` 安装。：

- Numpy: `python -m pip install numpy`
- Pandas: `python -m pip install pandas`
- Matplotlib: `python -m pip install matplotlib`

建议为单独的项目设置一个 Python 虚拟环境：

- 创建虚拟环境：在 VSC 中新建终端，运行：`python -m venv <env_name>`，其中 `<env_name>` 是你想给虚拟环境起的名称；
- 激活虚拟环境：Windows 系统下使用：`<env_name>Scripts/activate`，macOS 下使用：`source <env_name>/bin/activate`
- 停用虚拟环境：`deactivate`；

更快捷的方式是通过下载记录有程序包名字的 `requirements.txt` 文件进行安装：

```
pip install -r requirements.txt
```

可以将需要的第三方程序一次安装。

第四章 脚本模式和交互模式

Python 可以交互式或脚本模式运行。

4.1 交互式

用户输入代码，回车运行。在如 IDLE，Ipython 都可以方便地进行交互式操作。例如：

- 在命令提示符（或 VSC 终端）输入 `ipython`，将打开 Ipython 的界面，输入 `3 + 3`，回车，将在屏幕上立刻显示计算结果；
- 在应用程序中打开 IDLE Shell，也可以方便的进行交互式操作；
- 在 Jupyter Notebook 的代码单元格内，输入代码，点击运行显示结果；

4.2 脚本模式

脚本模式是将代码保存在.py 格式的文件中，然后使用命令提示符调用脚本。

例如，在文件夹 `pyfiles` 中保存有一个文件 `lunch.py`，定义了一个随机选择午餐的函数 `lunch()`，当运行该函数时，随机从列表中选择一个作为推荐的午餐。

```
import random

def lunch():
```

```
"""Randomly choose a lunch option and return the result."""
lunch_list = ["Rice Bowl", "Ramen", "Salad", "Burger", "Dumplings", "Pizza"]
return random.choice(lunch_list)

if __name__ == "__main__":
    result = lunch()
    print("Recomm:", result)
```

Recomm: Pizza

在命令提示符或终端中运行：

```
python pyfiles/lunch.py
```

当然，也可以先改变当前文件夹至 pyfiles 文件夹，就可以省略路径。

4.3 在 Visual Studio Code 中应用 Jupyter notebook

打开 VSC，点击“文件- 新建文件”，从弹出的菜单选择“Jupyter Notebook”。

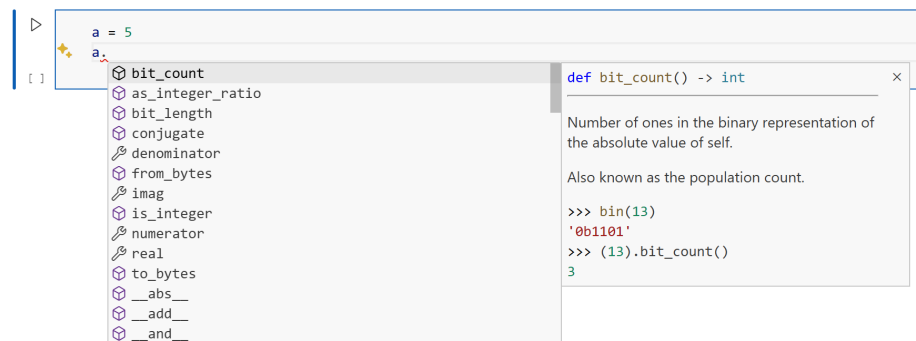


Figure 4.1: 新建 Jupyter Notebook

下面的例子来自[Matplotlib 官方网站](#)，将代码复制到 Notebook 的一个单元格中，点击左侧的运行三角箭头（VSC 也许会让你选择一个核），绘制

Figure 4.2 所示的一个累计概率分布图：

```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(19680801)

mu = 200
sigma = 25
n_bins = 25
data = np.random.normal(mu, sigma, size=100)

fig = plt.figure(figsize=(9, 4), layout="constrained")
axs = fig.subplots(1, 2, sharex=True, sharey=True)

# Cumulative distributions.
axs[0].ecdf(data, label="CDF")
n, bins, patches = axs[0].hist(data, n_bins, density=True, histtype="step",
                                cumulative=True, label="Cumulative histogram")
x = np.linspace(data.min(), data.max())
y = ((1 / (np.sqrt(2 * np.pi) * sigma)) *
      np.exp(-0.5 * (1 / sigma * (x - mu))**2))
y = y.cumsum()
y /= y[-1]
axs[0].plot(x, y, "k--", linewidth=1.5, label="Theory")

# Complementary cumulative distributions.
axs[1].ecdf(data, complementary=True, label="CCDF")
axs[1].hist(data, bins=bins, density=True, histtype="step", cumulative=-1,
             label="Reversed cumulative histogram")
axs[1].plot(x, 1 - y, "k--", linewidth=1.5, label="Theory")

# Label the figure.
fig.suptitle("Cumulative distributions")
```

```
for ax in axs:
    ax.grid(True)
    ax.legend()
    ax.set_xlabel("Annual rainfall (mm)")
    ax.set_ylabel("Probability of occurrence")
    ax.label_outer()

plt.show()
```

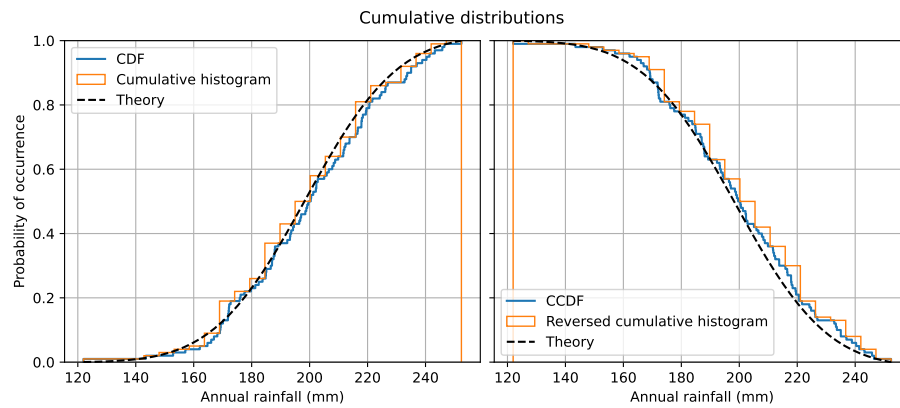


Figure 4.2: 累计概率分布

Jupyter Notebook 延续了 ipython 中的 `%run` 命令，可以脚本模式运行：

```
%run pyfiles/lunch.py
```

推荐的午餐：沙拉

第五章 文档与帮助

5.1 `help()` 和?

Python 有非常详细的官方[帮助文档](#)，帮助新用户快速的熟悉其用法。

例如，Python 有一个内置函数 `help()`，可以查看定义的文档，例如对函数 `len()`：

```
help(len)
```

```
Help on built-in function len in module builtins:
```

```
len(obj, /)
```

```
    Return the number of items in a container.
```

由于其重要性，Ipython 和 Jupyter 中可以使用? 作为缩写：

```
len?
```

当然，对自定义的对象也是适用的。下面定义的函数，有一段函数的说明文字（docstring）：

```
def square(x):  
    """  
    Calculates the square of a given number.  
  
    Args:  
        x (int or float): The number to be squared.
```

```
Returns:
    int or float: The square of the input number.
    """
    return x**2
```

如果输入 `help()` 函数:

```
help(square)
```

Help on function square in module __main__:

`square(x)`

Calculates the square of a given number.

Args:

x (int or float): The number to be squared.

Returns:

int or float: The square of the input number.

Bibliography

- [1] Wes McKinney. *Python for data analysis: Data wrangling with pandas, numpy, and jupyter*. " O'Reilly Media, Inc.", 2022.
- [2] Jake VanderPlas. *Python data science handbook: Essential tools for working with data*. " O'Reilly Media, Inc.", 2016.