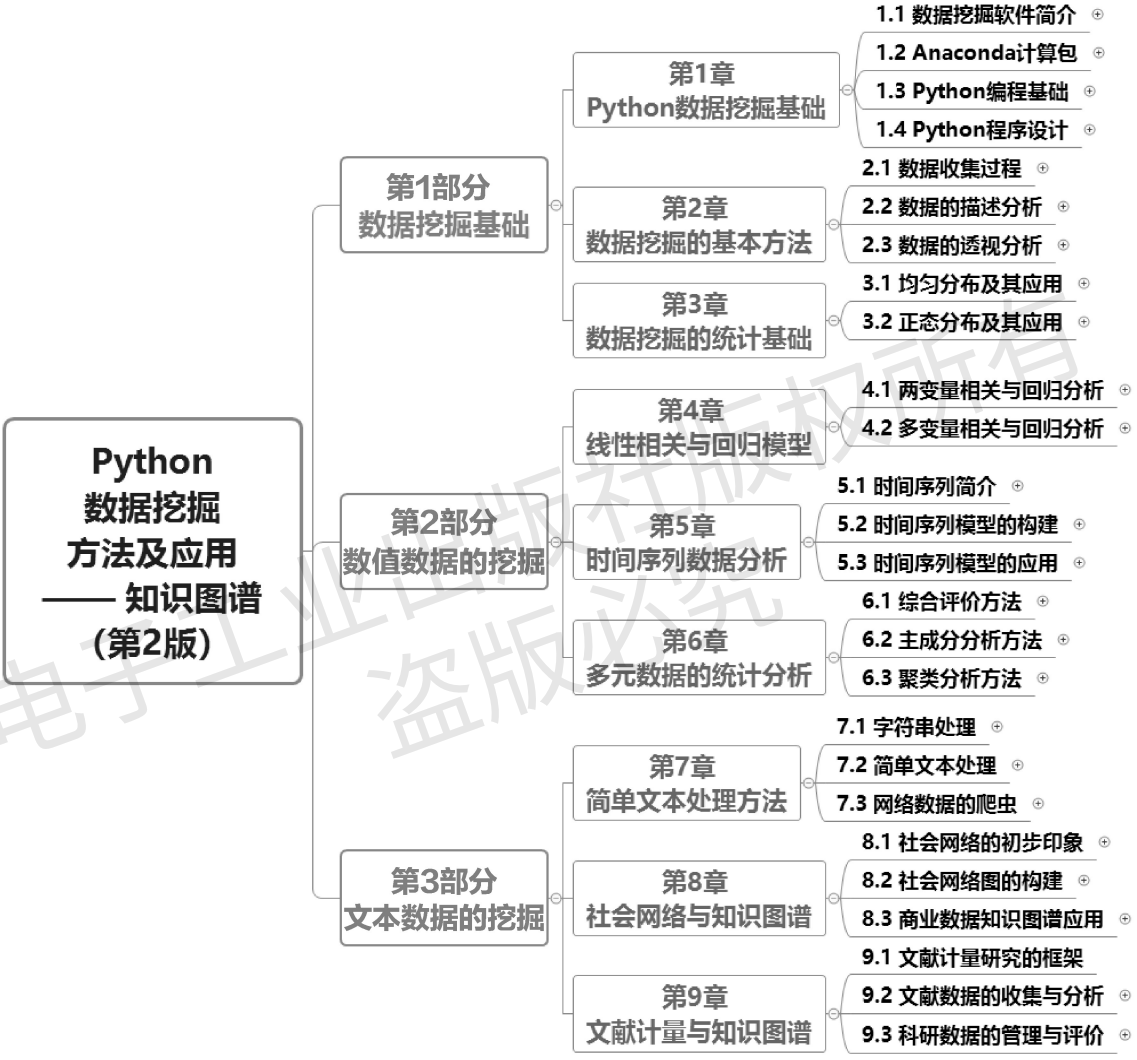
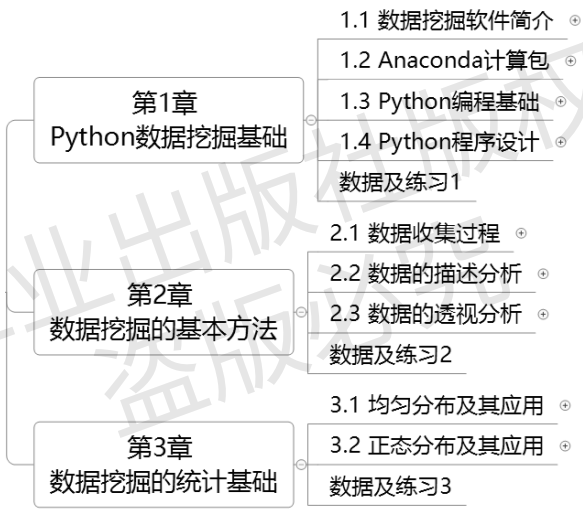


第 1 部分
数据挖掘基础



第 1 部分思维导图

第 1 章　Python 数据挖掘基础

第 1 章内容的知识图谱

1.1　数据挖掘软件简介

1.1.1　基本数据挖掘软件

能进行数据挖掘的软件有很多，如电子表格、SAS、SPSS、MATLAB、R、Python、Stata、Eviews 等，下面简单介绍这些软件。

电子表格（Excel、WPS 等）不仅是数据管理软件，还是数据挖掘的入门工具。尽管其统计分析功能并不十分强大，但是它可以快速地进行一些基本的数据分析工作，也可以创建供大多数人使用的数据图表。由于电子表格在数据存量、图形样式、统计方法和统计建模方面功能受限，所以它们很难成为专业的数据分析软件。

SAS（Statistical Analysis System）是使用最为广泛的三大著名统计分析软件（SAS、SPSS 和 Splus）之一，被誉为统计分析的标准软件。SAS 是功能最为强大的统计软件，有完善的数据管理和统计分析功能，是熟悉统计学并擅长编程的专业人士的首选。

SPSS（Statistical Package for the Social Science）也是世界上著名的统计分析软件之一。SPSS 中文名为社会科学统计软件包，这是为了强调其社会科学应用的一面，而实际上它在社会科学和自然科学的各个领域都能发挥巨大作用。与 SAS 相比，SPSS 是非统计学专业人士的首选。

MATLAB 是美国 MathWorks 公司出品的商业数学软件，是用于算法开发、数据可视化、数据分析及数值计算的高级技术计算语言和交互式环境，主要包括 MATLAB 和 Simulink 两大部分。它在数值计算和模拟分析方面首屈一指，主要应用于工程计算、控制设计、信号处理与通信、图像处理、信号检测、金融建模设计与分析等领域。

Stata 是一套完整的、集成的统计分析软件包，可以满足数据分析、数据管理和统计图形的所有需要。Stata 12 增加了许多新的特征，如结构方程模型（SEM）、ARFIMA、Contrasts、ROC 分析、自动内存管理等。Stata 适用于 Windows、Macintosh 和 Unix 平台计算机（包括 Linux）。Stata 的数据集、程序和其他的数据能够跨平台共享，且不需要转换，同样可以快速而方便地从其他统计分析软件包、电子表单和数据库中导入数据集。

Eviews 是美国 QMS 公司于 1981 年发行的第 1 版 Micro TSP 的 Windows 版本，通常称为计量经济学软件包，是当今世界最流行的计量经济学软件之一。它可应用于科学计算中的数据分析与评估、财务分析、宏观经济分析与预测、模拟、销售预测和成本分析等。由于 Eviews 提供了一个很好的工作环境，能够迅速进行编程、估计、使用新的工具和技术，所以它在计量经济建模方面有着广泛的应用。

从纯数据分析角度来说，应用最好的当属 S 语言的免费开源及跨平台系统 R 语言。R 语言是一个用于统计计算的很成熟的免费软件，也可以把它理解为一种统计计算语言，实际上很多人都直接称呼它为“R”，它比 C++、Fortran 等不知道简单了多少倍！如果你是一位数据分析的初学者，面对众多数据分析软件感到困惑且难以抉择，又想快速地掌握统计计算、数据分析，甚至目前比较流行的数据挖掘技术，那么首选的语言就是 R 语言。

不过，R 语言对于初学编程和数据分析的人来说，入门还是有一定难度的，因为它还不是真正意义上的编程语言，所以现在流行“人生苦短，我用 Python”这样的说法，说明 Python

作为一种新兴的编程语言，已深入人心。现在我国许多地区的高考试卷中都加入了 Python 编程的内容，一些中小学也开始开设 Python 编程课程。另外，由于 Python 博采众长，不断吸收其他数据分析软件的优点，并加入了大量的数据分析功能，它已成为仅次于 Java、C 及 C++ 的第四大语言，且在数据处理领域有超过 R 语言的趋势，因此本数据分析教程采用了 Python 作为分析工具。

综上所述，出于数据管理的方便，适用于一般数据分析的最好的数据管理软件应该是电子表格软件（如微软 Office 的 Excel、金山 WPS 的表格等），大量数据可以在一个工作簿中保存。所以，对于规模不是很大的数据集，建议采用该方法来管理和编辑数据，而统计软件是进行数据分析不可或缺的工具。随着知识产权保护要求的不断提高，免费和开放源代码逐渐成为一种趋势，Python 正是在这个大背景下发展起来的，并逐渐成为数据分析的标准软件。考虑到微软 Office 的 Excel 必须购买正版，而金山 WPS 的表格提供官方免费正版软件，作者认为，通常的数据处理和分析工作用 WPS+Python 足矣。

1.1.2 Python 语言介绍

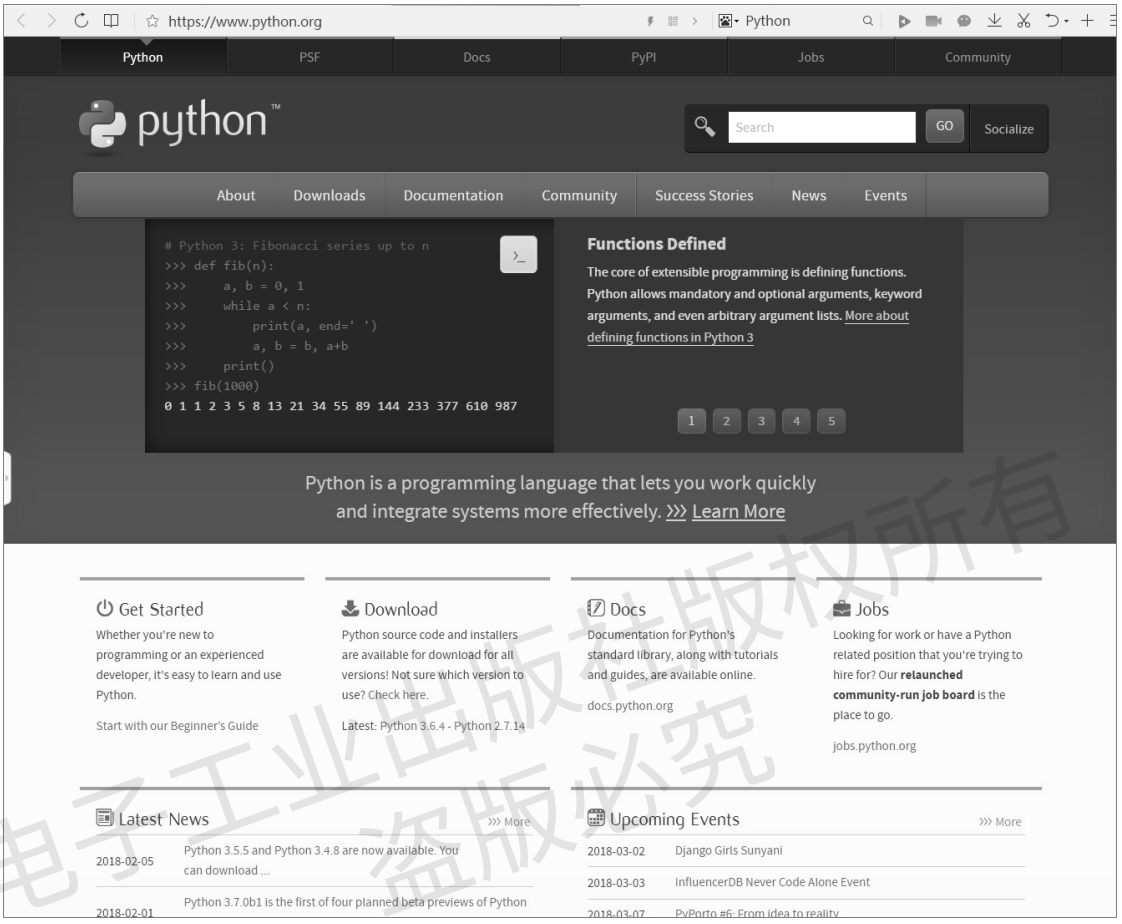
1.1.2.1 Python 简介

Python 是一种面向对象的解释型计算机程序设计语言，由荷兰人 Guido van Rossum 于 1989 年发明，第一个公开发行版发行于 1991 年。

Python 是纯粹的自由软件，源代码和解释器 CPython 遵循 GPL（General Public License）协议。Python 语法简洁清晰，特色之一是强制用空白符（White Space）作为语句缩进。

Python 具有丰富而强大的包，常被称为“胶水语言”，能够把用其他语言制作的各种模块（尤其是 C/C++）轻松地联结在一起。常见的一种应用情形是，先使用 Python 快速生成程序的原型（有时甚至是程序的最终界面），然后对其中有特别要求的部分用更合适的语言改写，如 3D 游戏中的图形渲染模块对性能要求特别高，就可以用 C/C++重写，最后封装为 Python 可以调用的扩展包。需要注意的是，在使用扩展包时可能需要考虑平台问题，某些扩展包可能不提供跨平台的实现。

由于 Python 语言的简洁性、易读性及可扩展性，在国外用 Python 进行科学计算的研究机构日益增多，一些知名大学已经采用 Python 来教授程序设计课程。例如，卡耐基梅隆大学的编程基础、麻省理工学院的计算机科学及编程导论就使用 Python 语言讲授。众多开源的科学计算软件包都提供了 Python 的调用接口，如著名的计算机视觉包 OpenCV、三维可视化包 VTK、医学图像处理包 ITK。而 Python 专用的科学计算扩展包就更多了，如以下三个十分经典的科学计算扩展包：numpy、scipy 和 matplotlib。它们分别为 Python 提供了快速数组处理、数值运算及绘图功能。因此，Python 语言及其众多的扩展包所构成的开发环境十分适合工程技术、科研人员处理实验数据、制作图表，甚至开发科学计算应用程序。Python 的官方网站为 https://www.python.org/，在该网站可以下载 Python 软件和许多程序包，以及有关 Python 的资料。



1.1.2.2　Python 的特色

Python 是一种高层次的脚本语言，其结合了解释性、编译性、互动性和面向对象，设计具有很强的可读性。

① Python 是解释型的语言：这意味着开发过程中没有编译这个环节。

② Python 是交互式的语言：这意味着可以在一个 Python 提示符下直接互动执行写程序。

③ Python 是面向对象的语言：这意味着 Python 支持面向对象的风格或代码封装在对象中的编程技术。

④ Python 是初学者的语言：Python 对初学者而言，是一种友好易学的语言，它支持广泛的应用程序开发——从简单的文字处理到网络开发再到游戏。

具体而言，Python 有以下一些特点。

① 简单、易学。

② 免费、开源。

③ 高层语言：封装内存管理等。

④ 可移植性：程序如果不使用依赖于系统的特性，那么不需要修改就可以在任何平台上运行。

⑤ 解释性：直接从源代码运行程序，不需要担心如何编译程序，使得程序更加易于移植。

⑥ 面向对象：支持面向过程的编程，也支持面向对象的编程。

⑦ 可扩展性：需要保密或高效的代码，可以先用 C/C++进行编写，然后在 Python 程序中使用。

⑧ 可嵌入性：可以把 Python 嵌入 C/C++程序，从而向程序用户提供脚本功能。

⑨ 丰富的包：包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI、FTP、电子邮件、XML、XML-RPC、HTML、WAV 文件、密码系统、GUI（图形用户界面）、Tk 和其他与系统有关的操作。

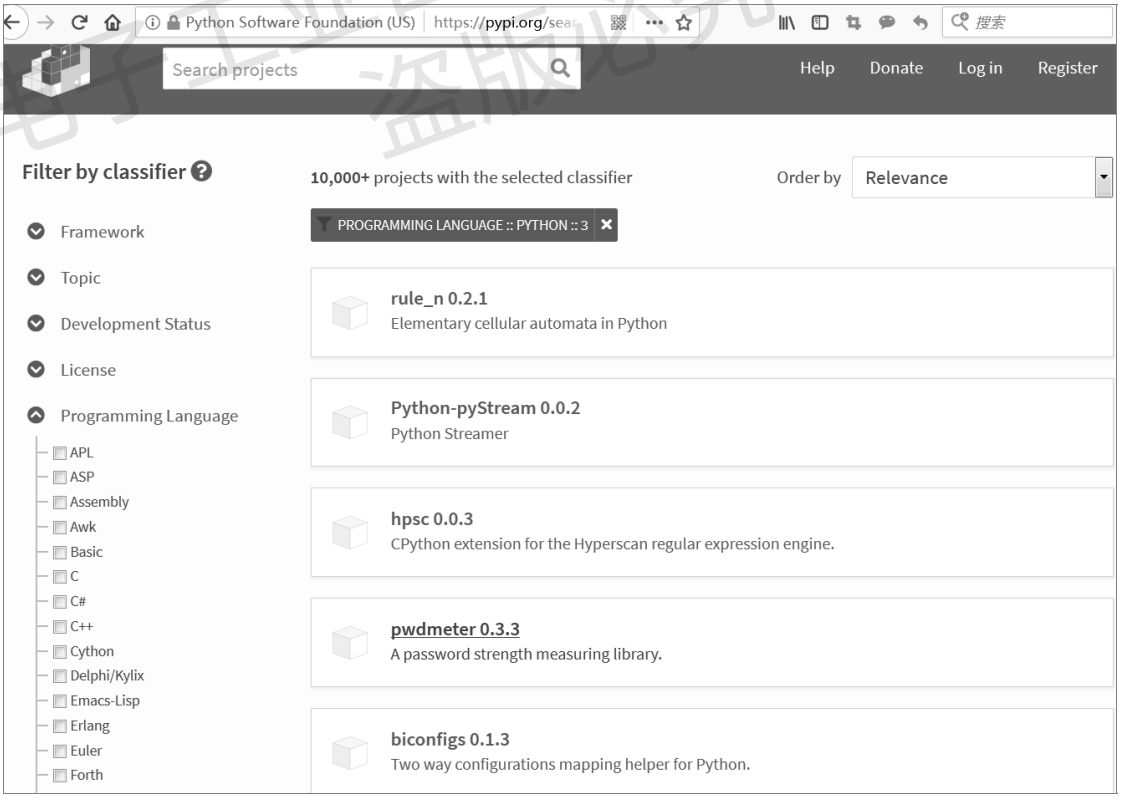
除标准包外，还有许多其他高质量的包，如 wxPython、Twisted 和 Python 图像包等。

⑩ 概括性强：Python 是一种十分精彩又强大的语言，使得编写程序简单有趣。

⑪ 规范的代码：Python 采用强制缩进的方式，使得代码具有极佳的可读性。

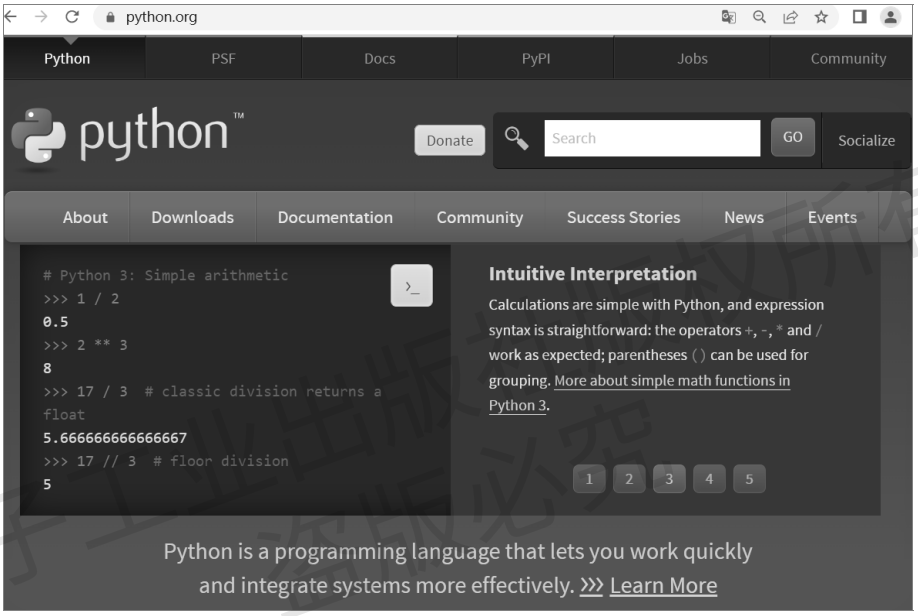
1.1.2.3 Python 的功能

Python 成为最流行的数据分析软件的特点是，它包含大量的扩展包并拥有方便的二次开发功能。Python 的扩展包包罗万象，它所能完成的数据统计模型已经超出了任何其他商业统计软件。作者做了一个统计，截至 2019 年 1 月，在 Python 的官方网站上所列的扩展包达到 165797 个（包含几十万个数据分析方法），除进行各种程序开发外，还可完全满足数据分析的要求。



1.1.2.4　Python 编程环境

Python 是一种强大的面向对象的编程语言，这样的编程环境需要使用者不仅熟悉各种命令的操作，还需要熟悉 DOS 编程环境，而且所有命令执行完即进入新的界面，这给那些不具备编程经验或对统计方法掌握不够好的使用者造成了极大的困难。从 Python 的官方网站上下载 Python 最新版，安装后只是一个包括基础包的语言环境。本书采用基于 Anaconda 的 Jupyter 平台进行数据分析。



1.2　Anaconda 计算包

如果用来讲课或演示数据分析结果，则推荐 Jupyter 平台，它有类似于 Mathematica 的界面，特点是可同时查看代码和运行结果，支持多种语言功能。如果用来进行数据挖掘和统计分析，则建议用 Spyder 平台；如果用来做大工程，则可考虑使用其他开发环境，如 Pycharm 等。你会发现，MATLAB、Rstudio、Spyder 三者“长得”很像，说明进行数据分析就应该是这样的界面。一个用熟了，其他两个就很容易上手了，可以将三者的常用功能的快捷键改成一致。

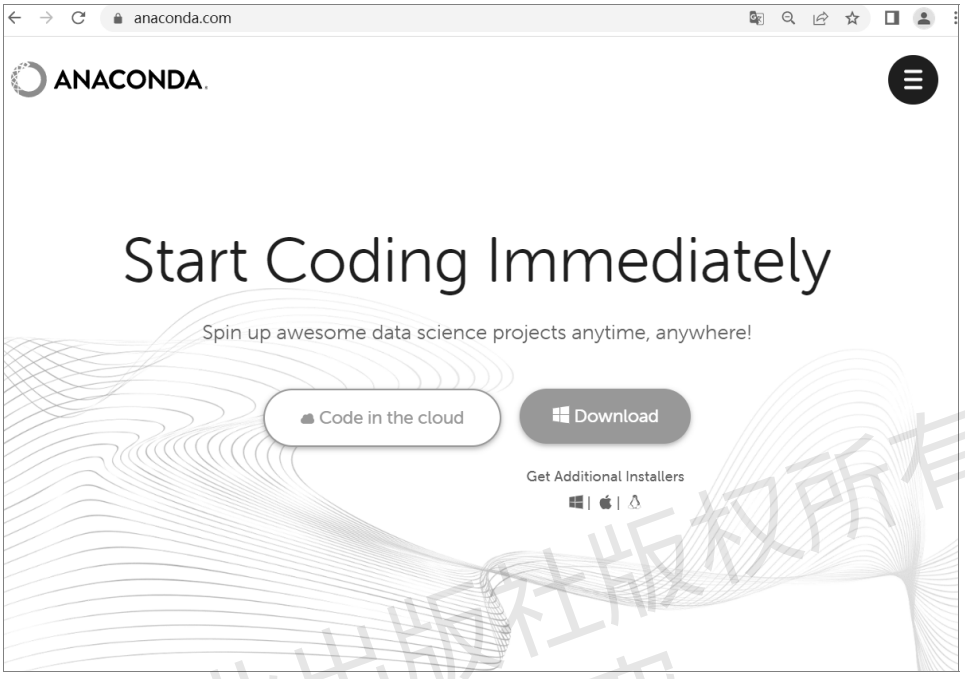
1.2.1　Anaconda 的使用

基本的 Python 编程环境只包含基本的编程模块，不包含数据分析和科学计算模块，所以数据分析工作者需要选择一个方便的 Python 编程环境。

可喜的是，现在有许多公司为了迎接大数据时代的来临，构建了许多基于 Python 的发行版，其中包含用于编程的 IDE（Integrated Development Environment，集成开发环境）、常用的编程和数据分析包。

这里给大家推荐一款用于科学计算和数据分析的 Python 的发行版 Anaconda，可登录

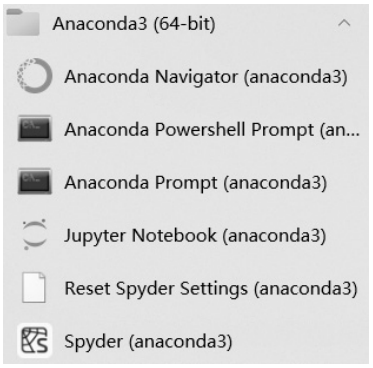
https://www.anaconda.com/网站下载其安装包。



注意：Anaconda 指的是一个开源的 Python 发行版，包含 numpy、pandas、matplotlib、scipy 等 180 个科学包及其依赖项。因为包含大量的科学计算包，所以 Anaconda 的下载文件比较大（约 500MB），但安装后可满足大多数数据分析的需求。

下载 Windows 版 Anaconda 的最新版本，按常规方法安装，安装后在 Windows 系统菜单中会出现子菜单，可选择其中一个程序来使用 Python。

在 Windows 中安装好 Anaconda 后，将会在 Windows 菜单中出现下面的界面。

	Anaconda 菜单： 系统导航（Navigator） 命令行（Prompt） 数据分析平台 （Jupyter Notebook） 数据挖掘平台 （Spyder）

从这里单击菜单中的按钮进入 Jupyter Notebook 或 Spyder。

第三方程序包通常在命令行上安装，安装命令为 pip install 包名或 conda install 包名，如要安装 nbextensions 扩展包，则在命令行执行：

```
>>> pip install jupyter_contrib_nbextensions
```

下面是一些包的命令。

列出当前安装的包：>>> pip list。

列出可升级的包：>>> pip list --outdate。

升级一个包：>>> pip install --upgrade jupyterlab。

卸载一个包：>>> pip uninstall jupyterlab。

如果要进行基本的数据分析和展示，则可执行 Jupyter Notebook。

1.2.2　Jupyter 分析平台

1.2.2.1　Jupyter Notebook

1）Jupyter Notebook 简介

Jupyter Notebook（此前称为 IPython Notebook）是一个交互式编程笔记本，支持运行 40 多种编程语言。Jupyter Notebook 的本质是一个 Web 应用程序，便于创建和共享流程化程序文档，支持实时代码、数学方程、可视化和 Markdown。用途包括数据清理、数据转换、数值模拟、统计建模、数据可视化机器学习等。其特点是用户可以通过电子邮件 Dropbox、GitHub 和 Jupyter Notebook Viewer，将 Jupyter Notebook 分享给其他人。在 Jupyter Notebook 中，代码可以实时生成图像、视频、LaTeX 和 JavaScript。

有时为了能与同行们有效沟通，需要重现整个分析过程，并将说明文字、代码、图表、公式、结论整合在一个文档中。显然，传统的文本编辑工具不能满足这一需求，而 Jupyter Notebook 不仅能在文档中执行代码，还能以网页形式分享。

2）Jupyter Notebook 的使用

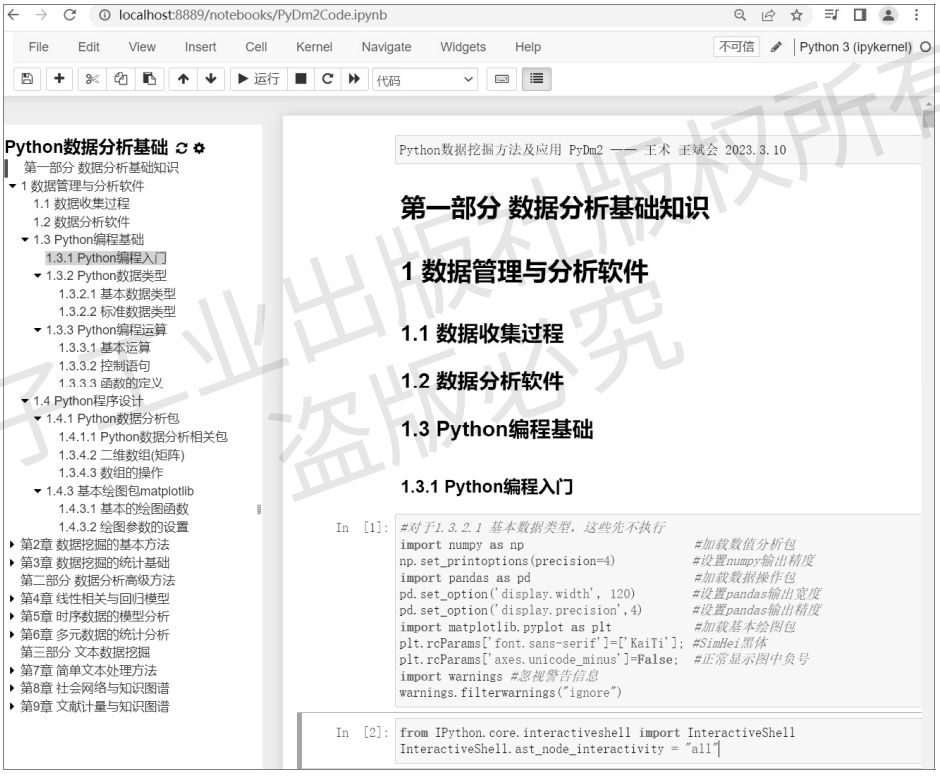
建议使用 Anaconda 发行版安装 Python 和 Jupyter，其中包括 Python、Jupyter Notebook、Jupyter Lab，以及用于科学计算和数据科学的其他常用软件包。

如果已经安装了 Jupyter Notebook，要运行笔记本，则在终端（Mac/Linux）或命令行（Windows）运行 Jupyter Notebook 命令。

如果安装的是 Anaconda，那么它已包含 Jupyter Notebook，由于 Jupyter 具有网页功能，因此直接在菜单中打开它，不容易确定当前执行目录。当你的目录不在计算机桌面上时，建议用下面的方式在当前目录（如 D:\PyDm2）中打开 Jupyter Notebook。

先在 Anaconda Prompt 命令行上将目录切换为 D:\PyDm2，然后运行 Jupyter Notebook 命令，如

```
C:\Users\Lenovo> D:
D:\>cd PyDm2
D:\PyDm2>jupyter notebook
```



3）Jupyter Notebook 的优点

Jupyter Notebook 的主要优点如下。

（1）所见即所得。

① 适合进行数据分析。想象以下混乱的场景：你在终端运行程序，可视化结果却显示在其他窗口中，而包含函数和类的脚本又存放在其他文档中，更可恶的是，你还需要另写一份说明文档来解释程序如何执行，以及结果如何。此时，Jupyter Notebook“从天而降”，将所有内容收归一处，你是不是顿觉灵台清明，思路更加清晰了呢？

② 支持多语言。Jupyter Notebook 支持 40 多种编程语言。如果你习惯使用 R 语言来进行数据分析，或者想用学术界常用的 MATLAB 和 Mathematica，那么只要安装相对应的核（Kernel）即可。

③ 分享便捷。支持以网页的形式分享，GitHub 天然支持 Jupyter Notebook 展示，也可以通过 Nbviewer 分享文档，当然也支持导出成 HTML、Markdown、PDF 等格式的文档。

④ 远程运行。在任何地点都可以通过网络连接远程服务器来实现运算。

⑤ 交互式展现。不仅可以输出图片、视频、数学公式，还可以呈现一些互动的可视化内容，如可以缩放的地图或可以旋转的三维模型。

（2）数学公式编辑。

如果你曾做过严格的学术研究，那么一定对 LaTeX 不陌生，这简直是写科研论文的必备工具，不但能实现严格的文档排版，而且能编辑复杂的数学公式。在 Jupyter Notebook 的 Markdown 单元中，也可以使用 LaTeX 的语法来插入数学公式。

在文本行插入数学公式，使用一对 $ 符号，如质能方程$E = mc^2$。如果要插入一个数学区块，则使用两对$符号。例如，用下面的公式表示 z=x/y：

```
$$ z = frac{x}{y} $$
```

关于如何在 Jupyter Notebook 中使用 LaTeX，可以上网查找相关资料。

（3）幻灯片制作。

既然 Jupyter Notebook 擅长展示数据分析的过程，那么除了通过网页形式分享，当然也可以将其制作成幻灯片的形式。

如何用 Jupyter Notebook 制作幻灯片呢？首先在 Jupyter Notebook 的菜单栏选择 View→Cell Toolbar→Slideshow，这时在文档的每个单元右上角显示了 Slide Type 的选项。通过设置不同的类型，来控制幻灯片的格式，有以下 6 种类型。

- Slide：主页面，通过按左右方向键进行切换。
- Sub-Slide：副页面，通过按上下方向键进行切换。
- Fragment：默认是隐藏的，按空格键或方向键后显示，可实现动态效果。
- Skip：在幻灯片中不显示的单元。
- Notes：作为演讲者的备忘笔记，也不在幻灯片中显示。
- Jupyter Notebook：幻灯片设置。

编写好幻灯片形式的 Jupyter Notebook 以后，如何来演示呢？这时需要使用 nbconvert：

```
jupyter nbconvert notebook.ipynb --to slides --post serve
```

在命令行输入上述代码后，浏览器会自动打开相应的幻灯片。

（4）魔术关键字。

魔术关键字（Magic Keywords），正如其名，是用于控制 Jupyter Notebook 的特殊命令。它们运行在代码单元中，以%或%%开头，前者控制一行，后者控制整个单元。

例如，要得到代码运行的时间，则可以使用%timeit；要在文档中显示 matplotlib 包生成的图形，则使用%matplotlib inline；要进行代码调试，则使用%pdb。注意：这些命令大多是在 Python Kernel 中适用的，在其他 Kernel 中大多不适用。有许多魔术关键字可以使用，更详细的清单请上网查询。

1.2.2.2　Jupyter Lab

相信 Python 开发者都对 Jupyter Notebook 这种笔记本式的开发环境非常喜欢。这种基于网

页的开发环境不仅允许用户创建和共享含有代码的文档，还可以植入公式、可视化图片和描述性文本等。

然而，所有的东西都不是十全十美的，我们在享受 Jupyter Notebook 带来便利的同时，总感觉有种或多或少的缺失感，因为感觉它不太像或压根就不算 IDE（集成开发环境），所以看着使用 PyCharm、Spyder 和 Visual Studio For Python 的用户，总有一种莫名的羡慕之感。

令所有开发者为之振奋的好消息是，Jupyter Notebook 的下一代产品 Jupyter Lab 发布了。

1）Jupyter Lab 的特点

① Jupyter Lab 是一个名副其实的 IDE，也是一个基于网页的 IDE（保留了全部的 Notebook 特性）。作者认为，仅凭这一条，Jupyter 项目就是一个飞跃。这个集成开发环境不仅有 Console，还有 IPython Terminal，所有开发所用到的资源（如图片、代码、文本等）、插件包等，都可以在其中运行 Python 和 R 等程序。

② 集成开发环境还内置了一个用起来得心应手且功能强大的 Markdown 编辑器，这对于编辑程序文档而言十分方便，再也不需要其他的编辑器来撰写 README 了。与大多数编辑器一样，该编辑器采取对照方式，一边为 Markdown 编辑页面，另一边为显示页面。

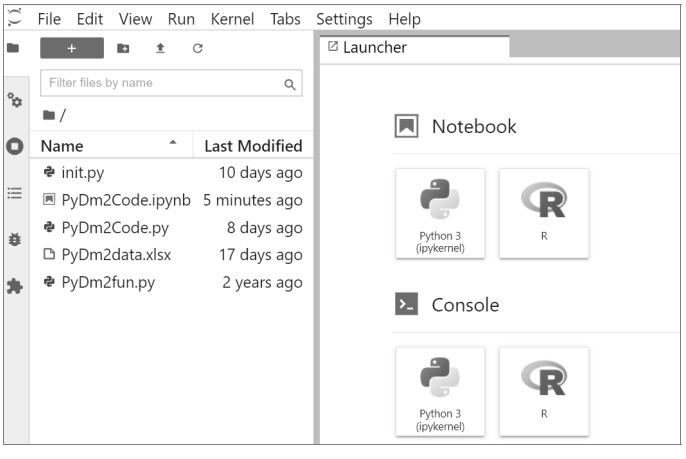
③ Jupyter Lab 有很多种打开方式，用于打开特定的数据结构和文件格式。例如，要打开一个 csv 文件，不是用 numpy/pandas 就是用 Excel，但 Jupyter Lab 提供了一种表格打开方式，可直接在页面打开这个表格数据，而不是逗号隔开的混乱数据。再例如，对于一个 Geo-JSON 文件，如何直观地实现可视化呢？用 Jupyter Lab 以地图形式打开，各个位置就直接显示在 Google Map 中了。

④ Jupyter Lab 扩展了小插件（Widget）功能。该功能采纳了其他交互性可视化项目的形式（如 Bokeh）。例如，可以通过滑块（Slider）来可视化改变变量值、图形大小、图的分布等。Jupyter Lab 还有很多令人惊喜的功能，这里不再赘述。

2）Jupyter Lab 的使用

如果你安装的是 Anaconda，那么它已包含 Jupyter Lab，由于 Jupyter 具有网页功能，因此直接打开它，不容易确定当期执行目录，可按以下步骤进行操作：进入工作目录文件夹（如 D:/PyDm2），在命令窗口中输入 Jupyter Lab，如下图所示。

```
D:\PyDm2\>Jupyter Lab
```



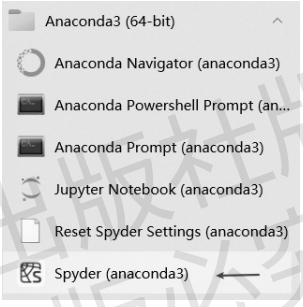
在此，就可以像在通常的编程环境中那样来编辑代码和进行数据分析了，操作类似于 Jupyter Notebook。

1.2.3　Spyder 编程平台[①]

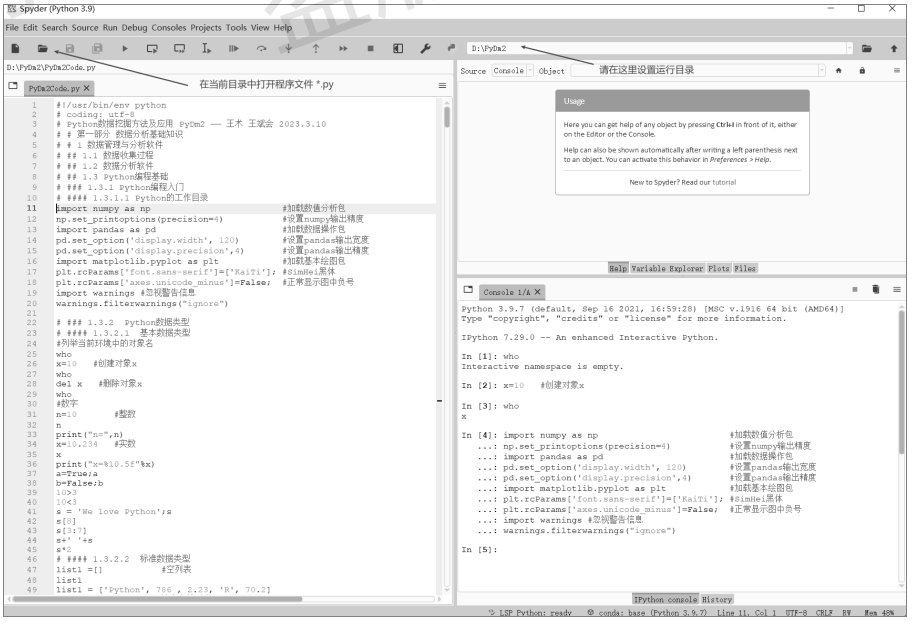
1.2.3.1　Spyder 平台简介

如果要在 Anaconda 中使用 Python 作为数据分析与开发平台，则推荐使用其 Spyder。Spyder 是 Python(x,y)（Python 的一个发行版）的作者为它开发的一个简单的集成开发环境。与其他 Python 的集成开发环境相比，它最大的优点是模仿 MATLAB 和 Rstudio 的“工作空间”功能，可以方便编辑代码和修改数组的值。

如果要进行大量的编程、数据处理和分析工作，则可使用 Spyder 编辑器实现类似 MATLAB 和 Rstudio 的开发环境。



下图所示为类似 MATLAB 和 Rstudio 的 Spyder 开发环境。



Spyder 是通过按 F9 键来运行代码的（可选择一行或多行执行）。

① 进行 Python 数据挖掘，建议使用 Spyder，编程和程序调试要比 Jupyter 方便很多。

1.2.3.2 Spyder 平台使用

关于 Spyder 的详细介绍，参见 Spyder 网站。上图就是调整后的 Spyder 界面，实际与 MATLAB 和 Rstudio 的编辑器差别不大，但更友好，熟悉 MATLAB 和 Rstudio 的用户较容易上手。

1）Spyder 的编辑

Spyder 的界面由许多窗格构成，用户可以根据自己的喜好调整它们的位置和大小。当多个窗格出现在同一个区域时，将以标签页的形式显示。在 View 菜单中可以设置是否显示 Editor、Object inspector、Variable explorer、File explorer、Console、History 和两个显示图像等窗格。

2）功能与技巧

Spyder 的功能比较多，这里仅介绍一些常用的功能和技巧。

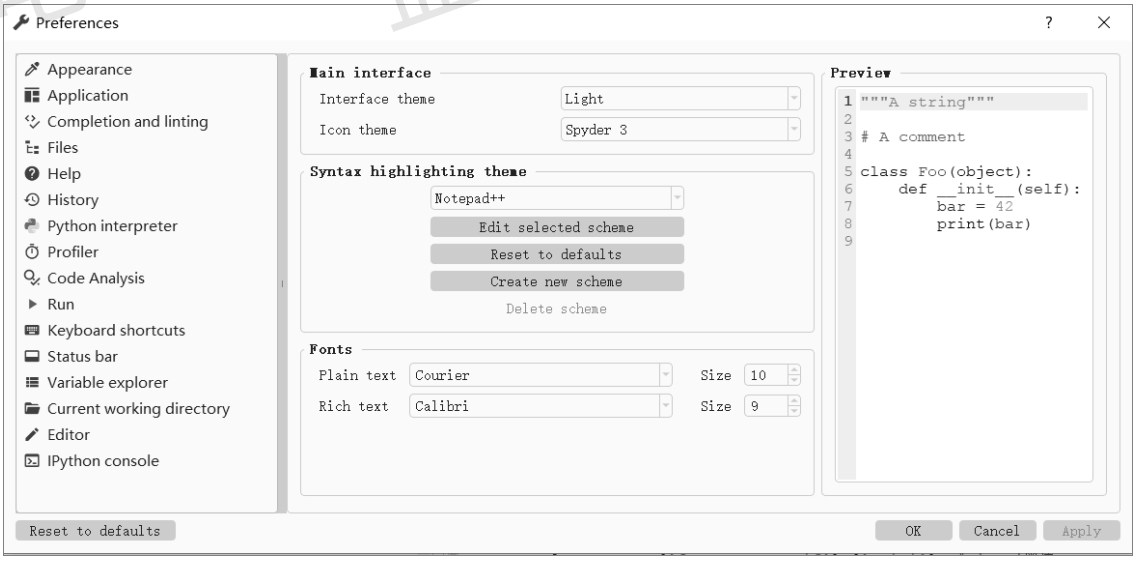
在控制台中，可以按 Tab 键进行自动补全。在变量名之后输入“?”，可以在 Object inspector 窗格查看对象的说明文档。此窗格的 Options 菜单中的 Show source 选项可以开启显示函数的源程序。

可以通过 Working directory 工具栏修改工作路径，用户程序运行时，将以此工作路径为当前路径。例如，只需要修改工作路径，就可以用同一个程序处理不同文件夹下的数据文件。

在程序编辑窗口中按住 Ctrl 键的同时单击变量名、函数名、类名或模块名，可以快速跳转到定义位置。如果是在别的程序文件中定义的，则将打开此文件。在学习一个新模块的用法时，经常需要查看模块中的某个函数或类是如何实现的，使用此功能可以快速查看和分析各个模块的源程序。

3）Spyder 的配置

Spyder 基本的配置都在 Tool→Preferences 中。



考虑到数据挖掘过程在大多数情况下需要通过编程来实现，所以本书采用 Spyder 进行操作。

1.3　Python 编程基础

网上有大量的 Python 编程基础知识介绍，如 http://www.runoob.com/Python/Python-dictionary.html，请大家自行学习。由于本书重点介绍 Python 的数据分析，所以对 Python 编程的基础知识不展开讨论。

1.3.1　Python 编程入门

Python 创建和控制的实体称为对象（Object），它们可以是变量、数组、字符串、函数或结构。由于 Python 是一种所见即所得的脚本语言，因此不需要编译。在 Python 中，对象是通过名字创建和保存的，可以用 who 命令来查看当前打开的 Python 环境中的对象，用 del 删除这些对象。

1）查看数据对象

In	#列举当前环境中的对象名 %who
Out	Interactive namespace is empty.

2）生成数据对象

In	x=10.12　　#创建对象 x %who
Out	x

3）删除数据对象

In	del x　#删除对象 x who
Out	Interactive namespace is empty.

上面列出的是新创建的数据对象 x 的名称。Python 对象的名称必须以一个英文字母打头，并由一串大小写字母、数字或下画线组成。

注意：Python 区分大小写，如 Orange 与 orange 数据对象是不同的。不要用 Python 的内置函数名作为对象的名称，如 who、del 等。

1.3.2　Python 数据类型

1.3.2.1　基本数据类型

Python 的基本数据类型包括数值型、逻辑型、字符型等，也可能是缺失值。

1）数值型

数值型数据的形式是实数，可以写成整数（如 n=3）、小数（如 x=1.46）、科学计数（y=1e9）的形式，该类型数据默认是双精度数据。

Python 支持 4 种不同的数值类型：int（有符号整型）、long（长整型，也可以代表八进制

和十六进制）、float（浮点型）、complex（复数）。

说明：Python 中显示数据或对象内容直接用其名称，相当于执行 print 函数，如下所示。

In	n=10 #整数
	n #无格式输出，相当于 print(n)
	print("n=",n) #有格式输出
	x=10.234 #实数
	print(x)
	print("x=%10.5f"%x)

Out	10 n= 10 10.234 x= 10.23400

2）逻辑型

逻辑型数据只能取值 True 或 False。

In	a=True;a b=False;b
Out	True False

可以通过比较获得逻辑型数据，如下所示。

In	10>3 10<3
Out	True False

3）字符型

字符型数据的形式是夹在双引号" "或单引号' '之间的字符串，如'MR'。**注意**：一定要用英文引号，不能用中文引号。Python 语言中的 string（字符串）是由数字、字母和下画线组成的一串字符。一般形式为

```
s = 'We love Python'
```

它在编程语言中表示文本的数据类型。

另外，Python 字符串具有切片功能，即从左到右索引默认从 0 开始，最大范围是字符串长度减 1（左闭右开）；从右到左索引默认从–1 开始。如果要实现从字符串中获取一段子字符串，则可以使用变量[头下标:尾下标]，其中下标从 0 开始算起，可以是正数或负数，也可以为空，表示取到头或尾。例如，下例中 s[8]的值是 P，s[3:7]的结果是 love。

加号（+）是字符串连接运算符，星号（*）是重复操作。

In	s = 'We love Python';s s[8] s[3:7]

	s+' '+s s*2
Out	'We love Python' 'P' 'love' 'We love Python We love Python' 'We love PythonWe love Python'

4）缺失值

有些统计资料是不完整的。当一个元素或值在统计的时候是“不可得到”或“缺失值”的时候，相关位置可能会被保留并且赋予一个特定的 nan（not available number，不是一个数）值。任何 nan 的运算结果都是 nan，如 float('nan')就是一个实数缺失值。

5）数据类型转换

有时候，需要对数据内置的类型进行转换。数据类型的转换，只需要将数据类型作为函数名即可。以下几个内置的函数可以实现数据类型之间的转换，这些函数返回一个新的对象，表示转换的值。下面列出几种常用的数据类型转换方式：

```
int(x [,base])   #将 x 转换为一个整数
float(x)         #将 x 转换为一个浮点数
str(x)           #将对象 x 转换为字符串
chr(x)           #将一个整数转换为一个字符
```

Python 的所有数据类型都是类，可以通过 type()函数查看该变量的数据类型。

1.3.2.2　标准数据类型

在内存中存储的数据可以有多种类型。例如，一个人的年龄可以用数字来存储，名字可以用字符来存储。Python 定义了一些标准类型，用于存储各种类型的数据。这些标准的数据类型是由前述基本类型构成的。

1）list

list（列表）是 Python 中使用最频繁的一种数据类型。列表可以完成大多数集合类的数据结构实现。它支持字符、数字、字符串，甚至可以包含列表，即嵌套。列表用“ [] ”标识，是一种最通用的复合数据类型。Python 的列表具有切片功能，列表中值的切割用到变量 [头下标:尾下标]，可以截取相应的列表，从左到右索引默认从 0 开始，从右到左索引默认从–1 开始，下标可以为空，表示取到头或尾。

加号（+）是列表连接运算符，星号（*）是重复操作。操作方法类似字符串。

list（列表）是进行数据分析的基本类型，所以必须掌握。

In	list1 =[]　　#空列表 list1 list1 = ['Python', 786 , 2.23, 'R', 70.2] list1　　#输出完整列表 list1[0]　　#输出列表的第一个元素 list1[1:3]　　#输出第二个至第四个元素

	list1[2:]	#输出从第三个开始至列表末尾的所有元素
	list1 * 2	#输出列表两次
	list1 + list1[2:4]	#打印组合的列表
Out	[]	
	['Python', 786, 2.23, 'R', 70.2]	
	'Python'	
	[786, 2.23]	
	[2.23, 'R', 70.2]	
	['Python', 786, 2.23, 'R', 70.2, 'Python', 786, 2.23, 'R', 70.2]	
	['Python', 786, 2.23, 'R', 70.2, 2.23, 'R']	
In	X=[1,3,6,4,9]; X	
	sex=['女','男','男','女','男']	
	sex	
	weight=[67,66,83,68,70];	
	weight	
Out	[1, 3, 6, 4, 9]	
	['女', '男', '男', '女', '男']	
	[67, 66, 83, 68, 70]	

2）tuple

tuple（元组）是另一种数据类型，类似于 list（列表）。元组用“()”标识，内部元素用逗号隔开。元组不能赋值，相当于只读列表，操作类似列表。

3）dictionary

dictionary（字典）也是一种数据类型，且可存储任意类型对象。字典的每个键值对用冒号“:”分隔，每个键值对之间用逗号“,”分隔，整个字典包括在花括号“{}”中，格式如下：

```
dict= {key1 : value1, key2 : value2 }
```

键必须是唯一的，但值则不必，值可以取任何数据类型，如字符串、数字或元组。

字典是除列表外 Python 中最灵活的内置数据结构类型。列表是有序的对象集合，字典是无序的对象集合。

两者之间的区别在于：字典中的元素是通过键来存取的，而不是通过下标来存取的。

In	{}	#空字典
	dict1={'name':'john','code':6734,'dept':'sales'};dict1	#定义字典
	dict1['code']	#输出键为'code'的值
	dict1.keys()	#输出所有键
	dict1.values()	#输出所有值
Out	{}	
	{'name': 'john', 'code': 6734, 'dept': 'sales'}	
	6734	
	dict_keys(['name', 'code', 'dept'])	
	dict_values(['john', 6734, 'sales'])	

In	dict2={'sex': sex,'weight':weight}; dict2　　#根据列表构成字典
Out	{'sex': ['女', '男', '男', '女', '男'], 'weight': [67, 66, 83, 68, 70]}

1.3.3　Python 编程运算

1.3.3.1　基本运算

与 Basic、Visual Basic、C、C++等一样，Python 具有编程功能，但 Python 是新时期的编程语言，具有面向对象的功能，同时 Python 还是面向函数的语言。既然 Python 是一种编程语言，就具有常规语言的算术运算符和逻辑运算符（见表 1-1），以及控制语句、自定义函数等功能。下面对 Python 的编程特点进行一些简单介绍。

表 1-1　Python 中常用的算术运算符和逻辑运算符

算术运算符	含　义	逻辑运算符	含　义
+	加	<（<=）	小于（小于或等于）
–	减	>（>=）	大于（大于或等于）
*	乘	==	等于
/	除	!=	不等于
**	幂	not x	非 x
%	取模	or	或
//	整除	and	与

1.3.3.2　控制语句

编程离不开对程序的控制，下面介绍几个最常用的控制语句，其他控制语句参见 Python 手册。

1）循环语句 for

Python 的 for 循环可以遍历任何序列的项目，如一个列表或一个字符串。for 循环允许循环使用向量或数列的每个值，在编程时非常有用。

for 循环的语法格式如下：

```
for iterating_var in sequence:
  statements(s)
```

Python 的 for 循环比其他语言的 for 循环更强大，例如：

In	for i in [1,2,3,4,5]: 　print(i)
Out	1 2 3 4 5
In	fruits = ['banana', 'apple', 'mango'] for fruit in fruits:

	print('当前水果 :', fruit)
Out	当前水果 : banana 当前水果 : apple 当前水果 : mango
In	[i*2 for i in [1,2,3,4,5]] #循环的快捷写法，并生成新的列表
Out	[2, 4, 6, 8, 10]

2）条件语句 if/else

if/else 语句是分支语句中的主要语句，其格式如下：

In	a = −100
	if a < 100:
	  print("数值小于 100")
	else:
	  print("数值大于 100")

Out	数值小于 100

Python 中有更简洁的形式来表达 if/else 语句。

In	−a if a<0 else a
Out	100

注意：循环和条件等语句中要输出结果，请用 print()函数，这时只用变量名是无法显示结果的。

1.3.3.3 函数的定义

在较复杂的计算问题中，有时一个任务可能需要重复多次，这时不妨自定义函数。这么做的好处是，函数内的变量名是局部的，即函数运行结束后它们不再保存到当前的工作空间，这就可以避免许多不必要的混淆和内存空间占用。Python 与其他统计软件的区别之一是，可以随时随地自定义函数，而且可以像使用 Python 的内置函数一样使用自定义的函数。

不同于 SAS、SPSS 等基于过程的统计软件，Python 进行数据分析是基于函数和面向对象的，所有 Python 的命令都是以函数形式出现的，如读取文本数据的 read_clipboard()函数和读取 csv 数据文件的 read_csv()函数，以及建立序列的 Series()函数和构建数据框的 DataFrame()函数。由于 Python 是开源的，因此所有函数使用者都可以查看其源代码。下面简单介绍 Python 的函数定义方法。定义函数的句法如下：

```
def 函数名(参数 1, 参数 2, …):
    函数体
  return
```

要学好 Python 数据分析，就必须掌握 Python 中的函数及其编程方法。表 1-2 所示为 Python 中常用的数学函数。

表 1-2　Python 中常用的数学函数

math 中的数学函数	含义（针对数值）	numpy 中的数学函数	含义（针对数组）
abs(x)	数值的绝对值	len(x)	数组中元素个数
sqrt(x)	数值的平方根	sum(x)	数组中元素求和
log(x)	数值的对数	prod(x)	数组中元素求积
exp(x)	数值的指数	min(x)	数组中元素最小值
round(x,n)	有效位数 n	max(x)	数组中元素最大值
sin(x),cos(x),…	三角函数	sort(x)	数组中元素排序
		rank(x)	数组中元素秩次

函数名可以是任意字符，但之前定义过的要小心使用，后定义的函数会覆盖先定义的函数。

注意：如果函数只用来计算，不需要返回结果，则可在函数中用 print()函数，这时只用变量名是无法显示结果的。

一旦定义了函数名，就可以像 Python 的其他函数一样使用。例如，定义一个用来求一组数据的均值的函数，可以用与 C、C++、Visual Basic 等语言相同的方式定义，但方便得多。如计算向量 $\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$ 的均值函数：

$$\overline{x}=\frac{\sum_{i=1}^{n}x_i}{n}$$

代码如下：

In	x=[1,3,6,4,9,7,5,8,2]; x def xbar(x): 　n=len(x) 　xm=sum(x)/n 　return(xm) xbar(x)
Out	[1, 3, 6, 4, 9, 7, 5, 8, 2] 5.0

当然，在 Python 中可以调用现成的均值计算函数，如下：

In	import numpy as np np.mean(x)
Out	5.0

要了解任何一个 Python 函数，使用 help()函数即可。例如，help(sum)或?sum 命令将显示 sum()函数的使用帮助。

1.4　Python 程序设计

Python 具有丰富的数据分析模块，大多数进行数据分析的人使用 Python 是因为其强大的

数据分析功能。所有的 Python 函数和数据集是保存在包里面的。只有当一个包被安装并被载入（import）时，它的内容才可以被访问。这样做，一是为了高效（完整的列表会耗费大量的内存，并且增加搜索的时间）；二是为了帮助包的开发者，防止命名和其他代码中的名称冲突。

1.4.1 Python 数据分析包

1.4.1.1 Python 数据分析相关包

由于 Anaconda 发行版已安装常用的数据分析包，所以我们只需要调用即可。下面介绍几个 Python 常用数据分析包，如表 1-3 所示。

表 1-3 Python 常用数据分析包

包　名	说　明	主 要 功 能
math	基础数学包	提供函数，完成各种数学运算
random	随机数生成包	Python 中的 random 模块用于生成各种随机数
numpy	数值计算包	numpy（numeric python）是 Python 的一种开源的数值计算扩展，一个用 Python 实现的数值计算工具包。它提供许多高级的数值编程工具，如矩阵数据类型、矢量处理，以及精密的运算包。专为进行严格的数值处理而产生
scipy	数值分析包	提供很多科学计算工具包和算法，易于使用，专为科学和工程设计的数值分析工具包。它包括统计、优化、整合、线性代数模块、傅里叶变换、信号和图像处理、常微分方程求解器等，包含常用的统计估计和检验方法
pandas	数据操作包	提供类似于 R 语言的 DataFrame 操作，非常方便。pandas 是面板数据（panel data）的简写。它是 Python 中最强大的数据分析和探索工具，因金融数据分析工具而开发，支持类似 SQL 的数据增、删、改、查，支持时间序列分析，灵活处理缺失数据
statsmodels	统计模型包	statsmodels 可以补充 scipy.stats，是一个包含统计模型、统计测试和统计数据挖掘的 Python 模块。对每个模型都会生成一个对应的统计结果，对时间序列有完美的支持
matplotlib	基本绘图包	matplotlib 主要用于绘图和绘表，是一个强大的数据可视化工具，也是一个 Python 的图形框架，类似于 MATLAB 和 R 语言。它是 Python 最著名的绘图库，提供了一整套与 MATLAB 相似的命令 API，十分适合交互式制图。也可以方便地将它作为绘图控件，嵌入 GUI 应用程序中
sklearn	机器学习包	sklearn 是基于 Python 的机器学习工具模块，里面主要包含 6 大模块：分类、回归、聚类、降维、模型选择、预处理，如使用 sklearn.decomposition 可进行主成分分解
beautifulSoup	网络爬虫包	beautifulSoup 是 Python 的一个包，最主要的功能是从网页抓取数据。beautifulSoup 提供一些简单的、Python 式的函数，用来处理导航、搜索、修改分析树等功能。通过解析文档为用户提供需要抓取的数据，通过它可以很方便地提取出 HTML 或 XML 标签中的内容
networkx	复杂网络包	networkx 是一款 Python 的软件包，用于创造、操作复杂网络，以及学习复杂网络的结构、动力学及其功能。通过它可以用标准或不标准的数据格式加载或存储网络，可以产生许多种类的随机网络或经典网络，也可以分析网络结构、建立网络模型、设计新的网络算法、绘制网络等

1.4.1.2　Python 包的安装与使用

注意：安装程序包和载入程序包是两个概念，安装程序包是指将需要的程序包安装到计算机中，载入程序包是指将程序包调入 Python 环境中。程序包的安装通常在下面的命令行状态下：

```
>>> pip install pandas
```

Python 调用包的命令是 import，如果要调用上述包，则可用：

```
import math
import random
import numpy
import scipy
import pandas
import matplotlib
```

这些包中的函数，可直接使用包名加“.”。如果要用 matplotlib 绘制 plot 图，则可用 matplotlib.plot(…)。

如果要简化这些包的写法，则可用 as 命令赋予别名：

```
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib as plt
```

这样 matplotlib.plot(…)可简化为 plt.plot(…)。

如果要调用 Python 包中某个具体函数或方法，则可使用 from…import。例如，要调用 math 包中的开方、对数和 pi 函数，则用

```
from math import sqrt, log, pi
```

这样，可在程序中直接使用，如 sqrt(2)等价于 math.sqrt(2)。

例如，下面是一些常用包的加载及设置。

In	import numpy as np	#加载数值分析包
	np.set_printoptions(precision=4)	#设置 numpy 输出精度
	import pandas as pd	#加载数据操作包
	pd.set_option('display.width', 120)	#设置 pandas 输出宽度
	pd.set_option('display.precision',4)	#设置 pandas 输出精度
	import matplotlib.pyplot as plt	#加载基本绘图包
	plt.rcParams['font.sans-serif']=['SimHei'];	#SimHei 黑体
	plt.rcParams['axes.unicode_minus']=False;	#正常显示图中负号

例如，要调用本书自定义函数文档 PyDm2fun.py 中的函数（见相关章节及附录），需要按以下方式进行操作。

（1）安装自定义模块：将 PyDm2fun.py 文档复制到当前工作目录 D:\PyDm2 下。

（2）加载自定义模块：%run PyDm2fun.py。

（3）自定义函数调用：mcor_test (X)。

In	%run PyDm2fun.py #调用自定义函数

1.4.2 数值分析包 numpy

在使用 numpy 包前，需要将其加载到内存中，语句为 import numpy，通常将其简化为 import numpy as np。

1.4.2.1 一维数组（向量）

下面是使用 Python 的 numpy 包对一维数组或向量进行的基本操作。

In	import numpy as np np.array([1,2,3,4,5])	#加载数组包 #一维数组
Out	array([1, 2, 3, 4, 5])	
In	np.array([1,2,3,np.nan,5])	#包含缺失值的数组
Out	array([1., 2., 3., nan, 5.])	
In	np.arange(9) np.arange(1,9,0.5) np.linspace(1,9,5)	#数组序列 #等差数列 #等距数列
Out	array([0, 1, 2, 3, 4, 5, 6, 7, 8]) array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. , 5.5, 6. , 6.5, 7. , 7.5, 8. , 8.5]) array([1., 3., 5., 7., 9.])	

1.4.2.2 二维数组（矩阵）

下面是使用 Python 的 numpy 包构建二维数组或矩阵的基本函数。

In	np.array([[1,2],[3,4],[5,6]])	#二维数组
Out	array([[1, 2], [3, 4], [5, 6]])	
In	A=np.arange(9).reshape(3,3);A	#形成 3×3 矩阵
Out	array([[0, 1, 2], [3, 4, 5], [6, 7, 8]])	

1.4.2.3 数组的操作

下面是对数组进行操作的一些常用函数。

1）数组的维度

In	A.shape
Out	(3, 3)

2）空数组

```
In   np.empty([3,3])                #空数组
Out  array([[ 4.6730e-307, 1.6912e-306, 1.8692e-306],
            [1.0236e-306, 1.4242e-306, 7.5660e-307],
            [8.4560e-307, 4.4505e-307, 2.3767e-312]])
```

3）零数组

```
In   np.zeros((3,3))                #零数组
Out  array([[0., 0., 0.],
            [0., 0., 0.],
            [0., 0., 0.]])
```

4）1 数组

```
In   np.ones((3,3))                 #1 数组
Out  array([[1., 1., 1.],
            [1., 1., 1.],
            [1., 1., 1.]])
```

5）单位数组

```
In   np.eye(3)                      #单位数组
Out  array([[1., 0., 0.],
            [0., 1., 0.],
            [0., 0., 1.]])
```


1.4.3　基本绘图包 matplotlib

1.4.3.1　基本的绘图函数

matplotlib 是 Python 的基本绘图包，也是 Python 的图形框架，类似于 MATLAB 和 R 语言。它是 Python 中最著名的绘图包，提供了一整套与 MATLAB 相似的命令 API，十分适合交互式制图。常用的绘图函数如表 1-4 所示。在绘制中文图形时，需要进行一些基本设置。

```
In   import matplotlib.pyplot as plt                #基本绘图包
     plt.rcParams['font.sans-serif']=['KaiTi'];     #KaiTi 楷体
     plt.rcParams['axes.unicode_minus']=False;      #正常显示图中负号
     plt.figure(figsize=(5,4));                     #图形大小
```

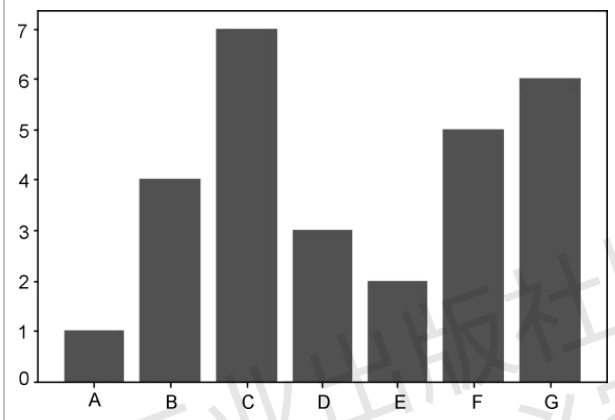
表 1-4　常用的绘图函数

计 数 数 据	用　途	计 量 数 据	用　途
bar( )	条图	plot( )	线图
pie( )	饼图	hist( )	直方图

1）计数数据的基本统计图

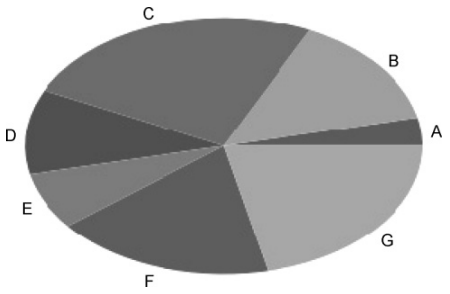
① 条图。

计数数据可以用条图描述。条图的高度可以是频数或频率，图的形状看起来一样，但是刻度不一样。matplotlib 画条图的函数是 bar()。在对计数数据绘制条图时，必须先对原始数据分组，否则绘制出的不是计数数据的条图。

In	X=['A','B','C','D','E','F','G'] Y=[1,4,7,3,2,5,6] plt.bar(X,Y);
Out	

② 饼图。

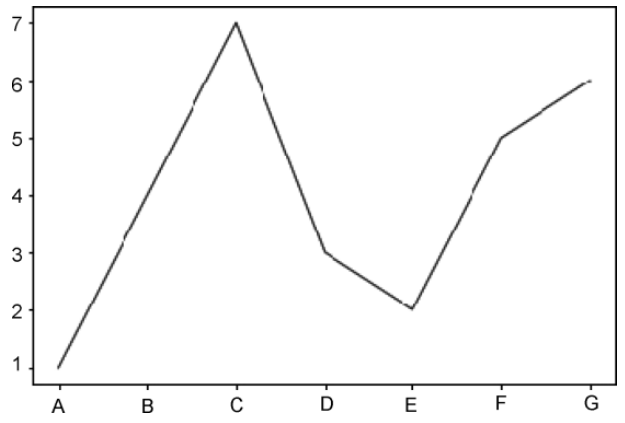
计数数据还可以用饼图描述。饼图用于表示各类别的构成比情况，它以图形的总面积为 100%，扇形面积的大小表示事物内部各组成部分所占的百分比。在 matplotlib 中绘制饼图也很简单，只要使用 pie()函数就可以了。值得注意的是，与条图一样，对原始数据绘制饼图前要先分组。

In	plt.pie(Y,labels=X);
Out	

2）计量数据的基本统计图

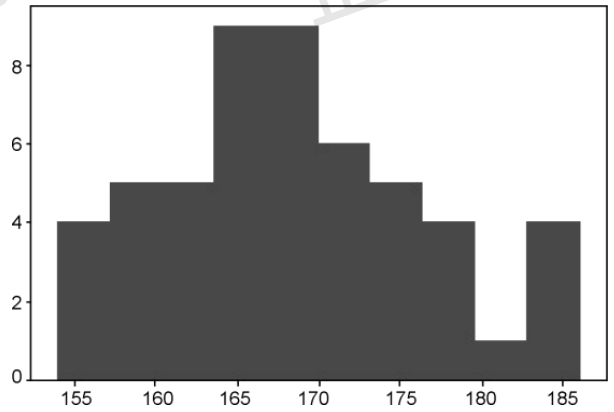
① 线图。

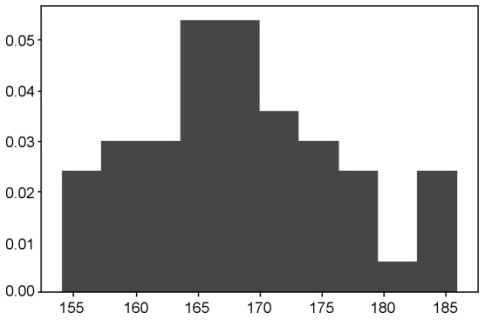
线图可以显示随时间变化的连续数据，主要用于显示在相等时间间隔下数据的趋势。

In	plt.plot(X,Y);
Out	

② 直方图。

直方图用于表示连续型变量的频数分布，常用于考察变量的分布是否服从某种分布类型，如正态分布。图形以矩形的面积表示各组段的频数（或频率），各矩形的面积总和为总频数（或等于 1）。matplotlib 中用来绘制直方图的函数是 hist()，也可以用频率绘制直方图，只要把 density 参数设置为 True 就可以了，默认为 False。

In	Z=np.random.normal(170,6,100)　#100 个身高均值为 170，标准差为 6 的随机数 plt.hist(Z)　　#频数直方图
Out	(array([4., 5., 5., 9., 9., 6., 5., 4., 1., 4.]), array([154.,157.2,160.4, 163.6, 166.8,170.,173.2,176.4,179.6, 182.8,186.]), 
In	plt.hist(Z,density=True)　　#频率直方图
Out	(array([　0.02403846　0.03004808　0.03004808　0.05408654　0.05408654 0.03605769　0.03004808　0.02403846　0.00600962　0.02403846]) array([154,157.2,160.4,163.6,166.8,170,173.2,176.4,179.6,182.8,186])



这些图是 Python 默认的形式，比较原始，可以通过设置不同的图形参数对图形进行调整和优化。

1.4.3.2 绘图参数的设置

1）图形参数设置

Python 中的每个绘图函数，都有许多参数设置选项，大多数函数的部分选项是一样的，下面列出一些主要的共同选项及其缺失值。

① 标题、标签、标尺及颜色。

在使用 matplotlib 模块绘制坐标图时，往往需要对坐标轴设置很多参数，这些参数包括横/纵坐标轴范围、坐标轴刻度、坐标轴名称等。

在 matplotlib 中有很多函数，用来对这些参数进行设置。

plt.xlim()，plt.ylim()：设置横/纵坐标轴范围。

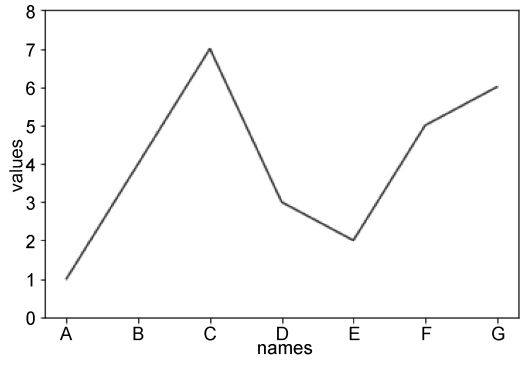
plt.xlabel()，plt.ylabel()：设置坐标轴名称。

plt.xticks()，plt.yticks()：设置坐标轴刻度。

colors 参数用来控制图形的颜色，可简写为 c，c='red'表示设置为红色。

In
```
plt.plot(X,Y,c='red');plt.ylim(0,8);
plt.xlabel('names');plt.ylabel('values');
plt.xticks(range(len(X)), X);
```

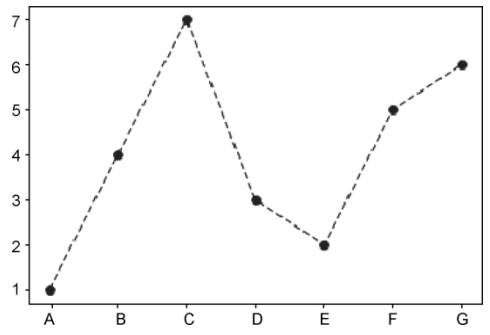
Out



② 线型和符号。

linestyle 参数用来控制连线的线型(-: 实线，--: 虚线，-: 点线)。

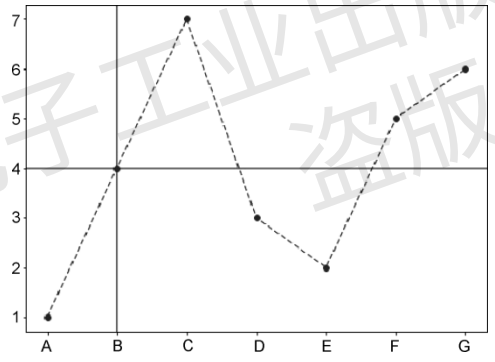
marker 参数用来控制符号的类型，如'o'为绘制实心圆点图。

In	plt.plot(X,Y,linestyle='--',marker='o');
Out	

③ 绘图函数附加图形。

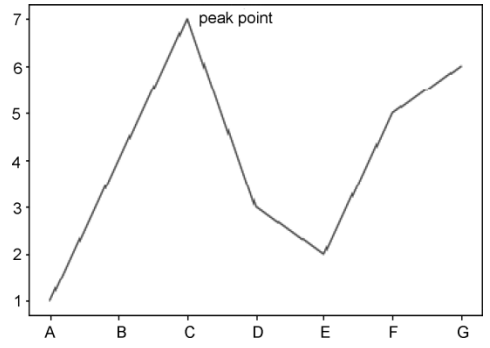
使用高级绘图函数可以绘制出一幅新图，而低级绘图函数只能作用于已有的图形之上。

- 垂线：在纵坐标 y 处画垂直线（plt.axvline()）。
- 水平线：在横坐标 x 处画水平线（plt.axhline()）。

In	plt.plot(X,Y,'o--'); plt.axvline(x=1);plt.axhline(y=4);
Out	

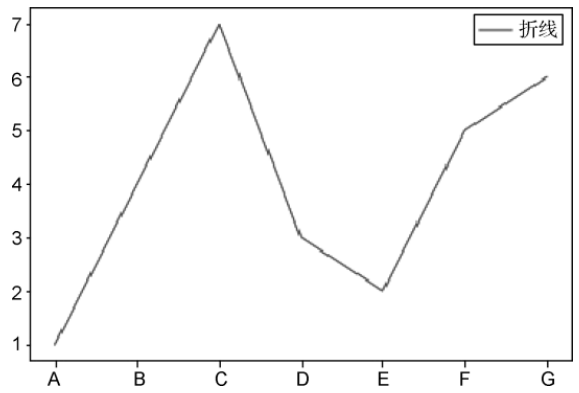
④ 文字函数。

text(x, y, labels,…)：在(x,y)处添加用 labels 指定的文字。

In	plt.plot(X,Y);plt.text(2, 7, ' peak point');
Out	

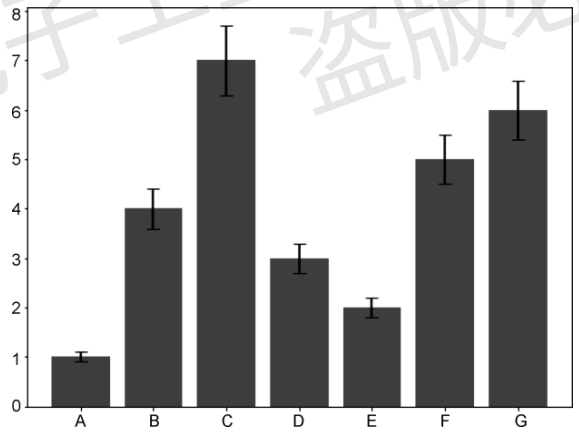
⑤ 图例。

绘制图形后，可使用 legend()函数给图形加图例。

In	plt.plot(X,Y,label=u'折线');plt.legend();
Out	

2）误差条图

误差条图由带标记的线条组成，通常这些线条用于显示有关图中所显示的数据的标准差信息。

In	s=[0.1,0.4,0.7,0.3,0.2,0.5,0.6] plt.bar(X,Y,yerr=s,error_kw={'capsize':5});
Out	

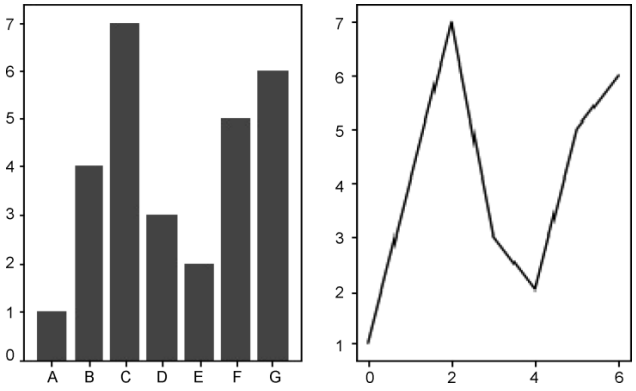
3）多图

在 matplotlib 下，一个 Figure 对象可以包含多个子图（Axes），可以使用 subplot()函数快速绘制，其调用形式如下：

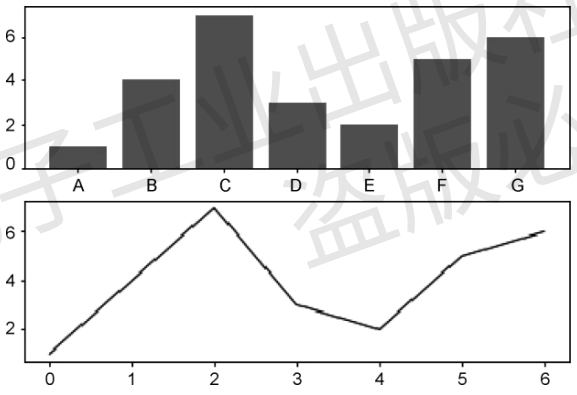
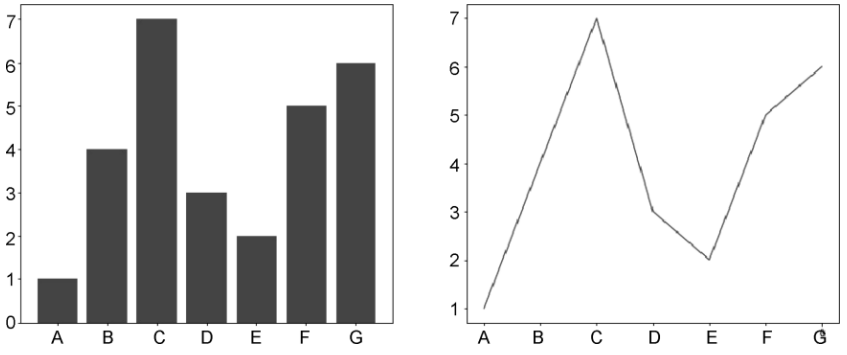
```
subplot(numRows, numCols, plotNum)
```

图表的整个绘图区域先被分成 numRows 行和 numCols 列，然后按照从左到右、从上到下的顺序对每个子区域进行编号，左上子区域的编号为 1，plotNum 参数指定创建的 Axes 对象所在的区域。

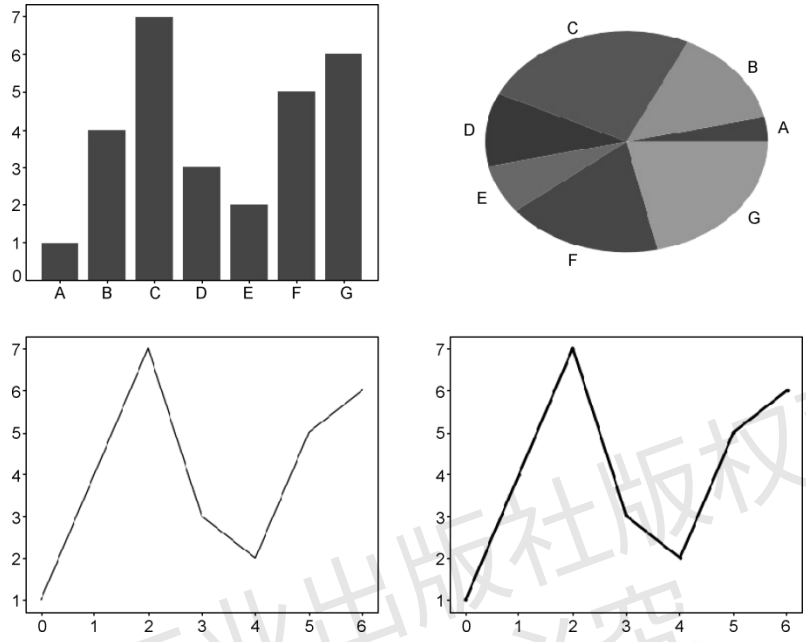
In	'''一行绘制两个图形'''
	plt.subplot(121); plt.bar(X,Y);
	plt.subplot(122); plt.plot(Y);

Out	

In	'''一列绘制两个图形 '''
	plt.subplot(211); plt.bar(X,Y);
	plt.subplot(212); plt.plot(Y);

Out	
In	fig,ax = plt.subplots(1,2,figsize=(15,6)) #一页绘制两个图形 ax[0].bar(X,Y);ax[1].plot(X,Y);
Out	
In	fig,ax=plt.subplots(2,2,figsize=(15,12)) #一页绘制四个图形 ax[0,0].bar(X,Y)

	ax[0,1].pie(Y,labels=X)
	ax[1,0].plot(Y);
	ax[1,1].plot(Y,'.-',linewidth=3);

Out	

数据及练习 1

1.1　下面有三组数据：

```
1, 2, 3, 4, 5
a, b, c, d
physics, chemistry, 1997, 2000
```

（1）将其写入列表。

（2）将其写入字典。

1.2　请创建下列 Python 数组，并计算。

（1）创建一个 2×2 的数组，计算对角线上元素的和。

（2）创建一个长度为 9 的一维数据，数组元素为 0～8，并将它重新变为 3×3 的二维数组。

（3）创建两个 3×3 的数组，分别将它们合并为 3×6、6×3 的数组后，拆分为 3 个数组。

1.3　文本数据。下面有一些文本数据：

```
name,physics,Python,math,english
Google,100,100,25,12
Facebook,45,54,44,88
Twitter,54,76,13,91
Yahoo,54,452,26,100
```

（1）请将其写入列表。

（2）请将其写入字典。

（3）请将其写入数据框。

（4）请将其保存到 csv 格式的文档中，并从 read_csv()函数读入 Python。

1.4　调查数据。某公司对财务部门人员的抽烟状况进行调查，结果：否，否，否，是，是，否，否，是，否，是，否，否，是，是，否，是，否，否，是，是。

（1）请用列表录入该数据。

（2）请将这组数据输入电子表格，并将其读入 Python。

1.5　学生成绩。从某大学统计系的学生中随机抽取 24 人，对数学和统计学的考试成绩进行调查，数据如表 1-5 所示。

表 1-5　部分学生的数学和统计学考试成绩

编号	性别	数学	统计学	编号	性别	数学	统计学
1	男	81	72	13	女	83	78
2	女	90	90	14	女	81	94
3	女	91	96	15	男	77	73
4	男	74	68	16	男	60	66
5	女	70	82	17	女	66	58
6	女	73	78	18	男	84	87
7	男	88	89	19	女	80	86
8	男	78	82	20	女	85	84
9	男	95	96	21	男	70	82
10	女	63	75	22	男	54	56
11	女	85	86	23	女	93	98
12	男	60	71	24	男	68	76

（1）试将这组数据输入电子表格。

（2）分别用 Python 的 read_csv()函数和 read_excel()函数读取。

（3）用 Python 方法获取性别、数学和统计学成绩变量，并筛选不同性别学生的成绩。

（4）请在电子表格和 Python 中分别对性别、数学和统计学成绩排序。

1.6　电子表格。将 1.1 题～1.5 题中的数据统一放入一个 Excel 或 WPS 电子表格，每个表格（Sheet）放一组，并给文档起名为 mydata1.xlsx，以备后用。