第1章

Python数据分析与挖掘基础

引导案例

聪明能干的宰相达依尔

相传古代印度国王舍罕要褒奖他的聪明能干的宰相达依尔（国际象棋发明者），问他要什么。达依尔回答："陛下只要在国际象棋棋盘的第 1 个格子上放 1 粒麦子，第 2 个格子上放 2 粒麦子，以后每个格子的麦子数都按前一格的两倍计算。如果陛下按此法给我 64 格的麦子，就感激不尽，其他什么也不要了。"国王想："这还不容易！"于是，国王让人扛了一袋麦子，但很快用光了，再扛出一袋还不够。请你为国王算一下共要给达依尔多少粒麦子？如果 1 m^3 能装约 1.4×10^8 粒麦子，需要多大体积才能装下国王给达依尔的麦子？

1.1 数据分析与挖掘简介

数据分析可以分为广义的数据分析和狭义的数据分析，广义的数据分析就包括狭义的数据分析和数据挖掘。以下是两者的含义、主要联系和区别。

视频 1.1 数据分析与挖掘简介微课视频

1.1.1 狭义的数据分析与数据挖掘的含义

数据分析（data analysis）是指用适当的统计分析方法对收集来的数据进行分析，将它们加以汇总、理解并消化，以求最大化地开发数据的功能，发挥数据的作用。数据分析可划分为描述性数据分析、探索性数据分析、验证性数据分析。

数据挖掘（data mining）是指从大量的数据中，通过统计学、人工智能、机器学习等方法，挖掘出未知的且有价值的信息和知识的过程。

1.1.2 数据分析与数据挖掘的主要联系与区别

两者的联系：

两者都是以数据为支撑，利用统计学、计算科学、可视化图表工具等知识，开发数据的价值，发挥数据为经营决策服务的作用。

两者的区别：

（1）"数据分析"的重点是从数据中发现"内在规律"，而"数据挖掘"的重点是从数

据中发现“知识规则”。

（2）“数据分析”过程需要基于业务理解进行人工建模，而“数据挖掘”过程可以基于机器学习完成数据建模。

（3）“数据分析”得出的结论一般是一个指标统计量结果，而“数据挖掘”得出的结论是机器从训练数据集发现的模型或知识规则。

1.1.3 数据分析与数据挖掘的一般流程

数据分析的一般流程包括明确数据分析需求、数据采集、数据处理、数据分析、数据展现，以及撰写数据分析报告，如图 1-1 所示。

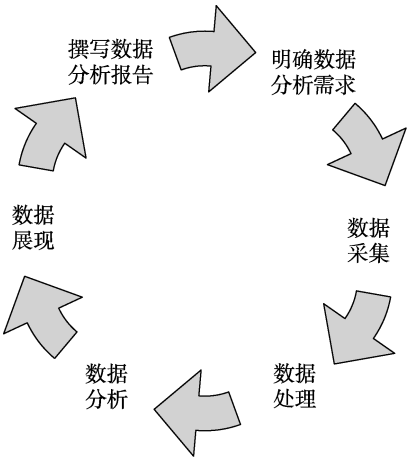


图 1-1 数据分析一般流程

（1）明确数据分析需求：数据分析的首要任务是明确数据分析的目标，确定数据分析的目的、过程以及方法。

（2）数据采集：通过一定的渠道和工具收集数据，是数据分析工作的基础。

（3）数据处理：通过一定的工具对采集到的数据中的噪声数据进行清洗、合并、转换等处理。

（4）数据分析：通过一定的分析手段、方法和技巧对处理后的数据进行探索、分析，从中发现因果关系、内部联系和业务规律。

（5）数据展现：借助图、表等可视化的方式直观的展示数据之间的关联信息，使得抽象的数据变得更加清晰、具体，易于观察，便于决策。

（6）撰写数据分析报告：以特定的形式将数据分析的过程、结果完整呈现出来，图文并茂，层次明晰，直观地看清楚问题和结论，便于需求者了解情况。

数据挖掘的一般流程包括数据收集、数据清洗、特征选择、模型构建、模型评估、模型应用，如图 1-2 所示。

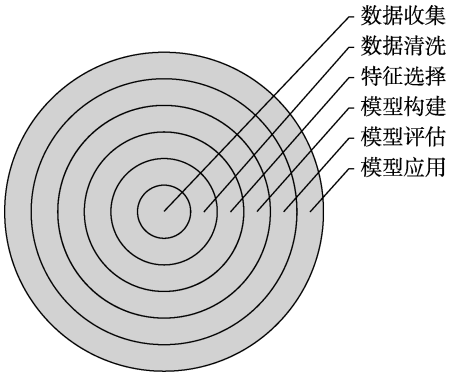


图 1-2 数据挖掘一般流程

（1）数据收集：通过一定的方式和工具从各种不同的数据源收集数据，收集的数据可能包括结构化数据、文本数据和图像数据等。

（2）数据清洗：通过一定的方式和工具对收集到的数据进行缺失值、异常值处理，格式统一处理等。

（3）特征选择：从给定的特征集中，按照一定准则选择出具有良好类别区分特性的子集,以提高学习效率或改善学习性能。

（4）模型构建：根据已选择的特征，利用机器学习技术来构建统计模型的过程。

（5）模型评估：通过一定的方法对机器学习算法训练得到的具体模型进行评估。当模型评估结果并不满足业务需求时，需要进行超级参数的调优以及尝试其他算法。

（6）模型应用：当模型的信度和效度已符合业务需求时，可以将该模型实现并部署在应用系统中，解决实际问题。

1.1.4 Python 数据分析与挖掘的优势

数据分析与挖掘的过程需要借助一定的工具，如 Excel、SQL、Power BI、SPSS、Python 等。利用 Python 进行数据分析与挖掘的优势在于以下 3 个方面。

（1）Python 是一种面向对象的脚本语言，其简单易学、开源免费，已成为当今最流行的编程语言。

（2）拥有强大的数据分析与挖掘第三方库功能，以及完备的社区生态。

（3）良好的可扩展性、可嵌入性和跨平台性。

1.2 Python 开发环境

视频 1.2 Python 开发环境微课视频

Python 程序开发一般包含两部分，即编写 Python 程序和运行 Python 程序，所以一个 Python 开发环境主要包含两个部分：编辑 Python 代码的编辑器和运行 Python 代码的解释器。Python 的集成开发环境（integrated development environment，IDE）有很多，不同的开发环境其配置难度与复杂度也不尽相同，最常用的有 PyCharm、Spyder、Jupyter Notebook 等。其中，Spyder 是一个免费、开源、科学的 Python 开发环境，由 Python 语言编写，非常适合科学家、工程师和数据分析师使用。Anaconda 是开源的 Python 发行版本，目的是将 Python 引入商业数据分析，拥有 Conda、Python、NumPy、Pandas、Matplotlib、Scikit-learn 等数据分析与挖掘所需要的库和包。而 Spyder 已经作为 Anaconda 一个组件而存在。

1.2.1 安装 Anaconda

登录 Anaconda 官方网站：www.anaconda.com，或者清华大学开源软件镜像站：https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/?C=N&O=D，下载操作系统（Windows/ Mac/Linux）匹配的安装包，本书以官方网站下载 Windows 版为例，如图 1-3 所示。

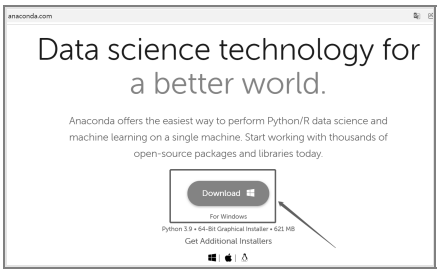


图 1-3 Anaconda 下载

双击下载的安装文件，依次单击 Next >→ I Agree→Just Me→Next >，进入更改安装路径窗

口，如图 1-4 所示。然后，勾选 Register Anaconda as the system Python 3.6，单击 Install 即可安装。

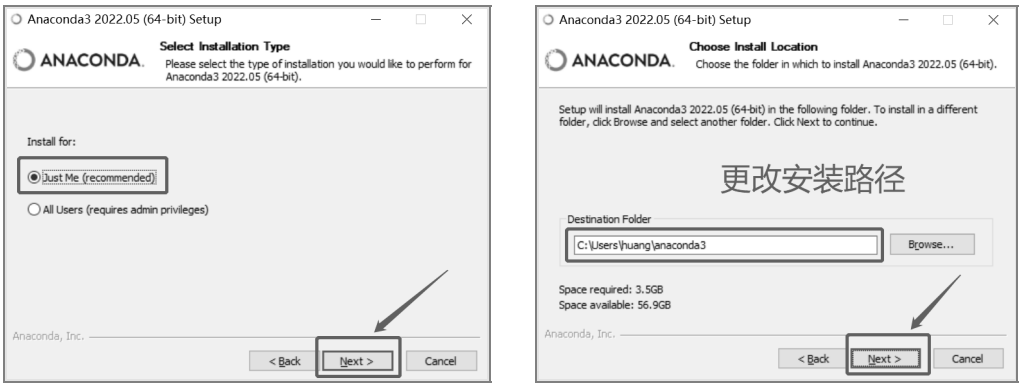


图 1-4　Anaconda 安装、选择“Just Me”及更改安装路径

Anaconda 安装成功后，单击开始菜单，查看 Anaconda3 目录，可以看到 Anaconda Navigator、Anaconda Prompt、Jupyter Notebook、Spyder 等组件。Navigator 是 Anaconda 发行版的可视化管理界面，允许用户在不使用命令行命令的情况下启动应用程序并轻松管理 conda 包、环境和通道。Prompt 是一个 Anaconda 的终端，可以使用命令行命令便捷地操作 conda 环境。Jupyter Notebook 是基于网页的用于交互计算的应用程序，常用于 Python 开发。Spyder 是一个强大的 Python 科学集成开发环境（IDE）。

1.2.2　认识 Spyder

从开始菜单中 Anaconda 目录下单击 Spyder(Anaconda3)进入 Spyder，单击菜单栏中 Tools 选项，依次选择 Preferences→Application→Advanced Settings，将 Language 由默认的 English 设置为“简体中文”，如图 1-5 所示，单击 Apply，然后同意重启 Spyder，使设置生效。

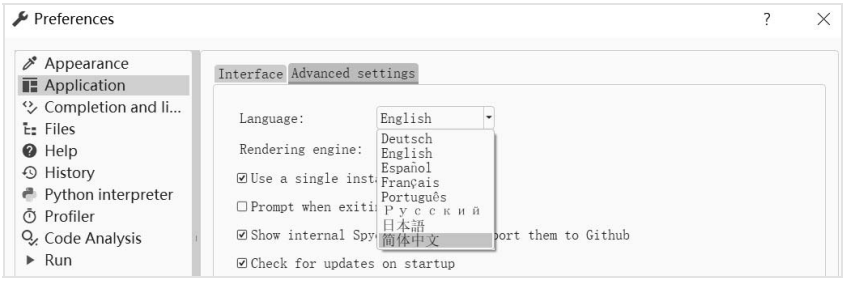


图 1-5　设置 Spyder 语言为简体中文

单击菜单栏中“查看”选项，依次选择“窗口布局”→“Matlab 布局”（图 1-6），将 Spyder 窗口设置为类 Matlab 风格或其他风格，如图 1-7 所示。

图 1-6　设置 Spyder 窗口布局为 Matlab 布局

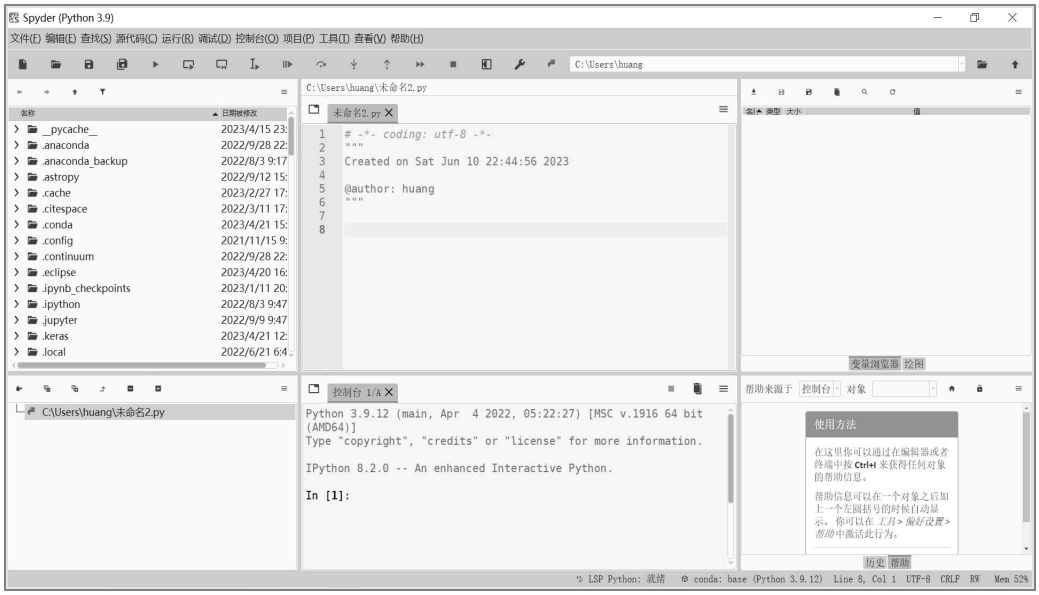


图 1-7　Matlab 布局窗口

（1）Spyder 窗口左上角工具栏允许快速访问 Spyder 中一些最常见的命令，如运行、保存和调试文件。

（2）Spyder 窗口右下角状态栏显示了当前的 Python 环境、git 分支、内存使用情况以及当前活动文件的各种属性。

（3）Spyder 窗口共有 6 个窗格，可以通过单击每个窗格右上角的汉堡包图标来显示每个窗格的选项菜单。它包含与窗格相关的有用设置和操作。

（4）在窗格上的任何地方单击鼠标右键，可以显示窗格的右键菜单。菜单显示与光标下的元素相关的操作。

（5）“编辑器”窗格可以创建、打开和编辑文件。它包含一些有用的功能，比如自动完成、实时分析和语法高亮显示，如图 1-8 所示。

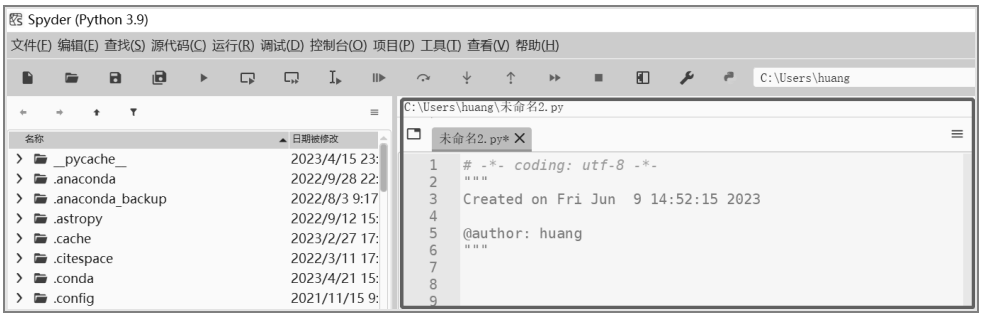


图 1-8 编辑器窗格

（6）“IPython 控制台”窗格允许从编辑器或交互地运行代码。还可以使用它来控制 Spyder 的调试器。

（7）“帮助”窗格显示了在编辑器或 IPython 控制台中使用的对象的文档。在这里你可以通过在编辑器或者终端中按 Ctrl+I 来获得任何对象的帮助信息。

（8）“历史”窗格可以查看在“编辑器”和 IPython 控制台输入的历史记录。

（9）“变量资源管理器”允许浏览在运行代码时生成的对象并与之交互。双击一个变量将打开一个专门的查看器，允许检查它的内容。

（10）“绘图”窗格显示在代码执行期间创建的图形和图像。它允许浏览、缩放、复制和保存生成的绘图。

（11）“文件”窗格可以浏览计算机上的目录，在“编辑器”中打开文件，并执行各种其他操作。

1.3 Python 数据类型及常用操作

视频 1.3 Python 数据类型及常用操作微课视频

Python 是面向对象的编程语言。Python 程序中的所有数据都是由对象或对象间关系来表示的。每个对象都有各自的标识号、类型和值，其中标识号可以理解为对象在内存中的地址，对象的类型决定该对象所支持的操作并且定义了该类型的对象可能的取值。

Python 对象的类型有很多，包括内置类型和扩展模块定义的类型。本节主要介绍内置类型中常用的 6 种基本类型及常用操作，分别是数字类型（整型、浮点型）、序列类型（字符串、列表、元组）、集合类型（集合）以及字典类型（映射）。列表、元组、字典和集合称为数据容器或数据结构，通过数据容器或数据结构可以把数据元素按照一定的规则存储起来。这些数据元素可以是数字、字符串、列表等数据类型中的一种或多种。

Python 程序的编写或应用就是通过操作数据容器中的数据，比如利用数据容器本身的方法，或者利用顺序、条件、循环语句，或者程序块、函数等形式，实现数据的处理、计算，最终达到应用目的。

1.3.1 数字类型及常用操作

1. 数字类型

数字类型对象是不可变的，一旦创建其值就不再改变。Python 中的数字非常类似数学中的数字。数字类型主要区分整型数、浮点型数。整型数又可细分为整型（int）和布尔型（bool），整型用来表示整数，布尔型表示逻辑值 True 和 False，分别对应整型的 1 和 0，浮点型（float）用来表示实数。数字是由数字字面值或内置函数与运算符的结果来创建的。创建数字类型的示例代码如下。

```
n1=200                    #整型
n2=100.3                  #浮点型
n3=float(60)              #整型转换为浮点型
t=True                    #布尔型（真）
f=False                   #布尔型（假）
```

执行结果如图 1-9 所示。

名称	类型	大小	值
f	bool	1	False
n1	int	1	200
n2	float	1	100.3
n3	float	1	60.0
t	bool	1	True

图 1-9　创建数字类型对象

2. 数字类型常用操作

整型数与浮点型数均支持常见的运算符与数值函数运算。如表 1-1 所示。

表 1-1　数字类型常用操作

运　算	结　果
x + y	x 与 y 的和
x - y	x 与 y 的差
x * y	x 与 y 的乘积
x / y	x 与 y 的商
x % y	x / y 的余数
abs(x)	x 的绝对值或大小
int(x)	将 x 转换为整数
float(x)	将 x 转换为浮点数
x ** y	x 的 y 次幂

注：数字类型更多运算及运算符优先级详见 Python3.9 官方中文文档：https://docs.python.org/zh-cn/3.9/library/stdtypes.html#numeric-types-int-float-complex。

1.3.2 序列类型及常用操作

字符串（str）、列表（list）和元组（tuple）均属于序列类型。

1. 字符串类型

字符串（str）是 Python 中处理文本数据的类型，是不可变序列。字符串字面值有多种不同的写法。

- 单引号: '允许包含有 "双" 引号'。
- 双引号: "允许嵌入 '单' 引号"。
- 三重引号: '''三重单引号''', """三重双引号"""。

使用三重引号的字符串可以跨越多行——其中所有的空白字符都将包含在该字符串字面值中。创建字符串类型的示例代码如下。

```
str1='hello world!'          #使用一对单引号创建
str2="I like Python!"        #使用一对双引号创建
str3=str('Hi,Jack')          #使用 str()函数创建
```

执行结果如图 1-10 所示。

名称	类型	大小	值
str1	str	12	hello world!
str2	str	14	I like Python!
str3	str	7	Hi,Jack

图 1-10　创建字符串类型对象

2. 列表类型

列表（list）是可变序列，用方括号括起来进行定义，元素与元素之间用逗号隔开。列表中的元素可以是数值、字符串、列表、元组等任何类型，并且在同一个列表中，元素的类型可以不同，但通常是同种类型。创建列表的示例代码如下。

```
list1=[]                     #创建空列表
list2=['how','are','you']    #使用方括号，元素以逗号分隔创建
list3=list((1, 2, 3))        #使用 list()函数创建
```

执行结果如图 1-11 所示。

名称	类型	大小	值
list1	list	0	[]
list2	list	3	['how', 'are', 'you']
list3	list	3	[1, 2, 3]

图 1-11　创建列表类型对象

3. 元组类型

元组（tuple）是不可变序列，用圆括号括起来进行定义，元素与元素之间用逗号隔开。元组中的元素可以是数值、字符串、列表、元组等任何类型的数据，并且在同一个元组中，元素的类型可以不同。元组与列表的不同之处在于元组中的元素不能修改。创建元组的示例代码如下。

```
tup1=()                          #创建空元组
tup2=(1,2,3,'Jack,GO!')          #使用圆括号，元素以逗号分隔创建
tup3=tuple('abc')                #使用 tuple()函数创建
```

执行结果如图 1-12 所示。

名称	类型	大小	值
tup1	tuple	0	()
tup2	tuple	4	(1, 2, 3, 'Jack,GO!')
tup3	tuple	3	('a', 'b', 'c')

图 1-12　创建元组类型对象

4. 序列类型常用操作

字符串、列表、元组都支持的操作，如表 1-2 所示。表格中，s 和 t 是具有相同类型的序列，n，i，j 和 k 是整数而 x 是任何满足 s 所规定的类型和值限制的任意对象。

表 1-2　序列类型常用操作

运　算	结　果
x in s	如果 s 中的某项等于 x 则结果为 True，否则为 False
x not in s	如果 s 中的某项等于 x 则结果为 False，否则为 True
s + t	s 与 t 相拼接
s * n 或 n * s	相当于 s 与自身进行 n 次拼接
s[i]	s 的第 i 项，起始为 0
s[i:j]	s 从 i 到 j 的切片
s[i:j:k]	s 从 i 到 j 步长为 k 的切片
len(s)	s 的长度
min(s)	s 的最小项
max(s)	s 的最大项
s.index(x[, i[, j]])	x 在 s 中首次出现项的索引号（索引号在 i 或其后且在 j 之前）
s.count(x)	x 在 s 中出现的总次数

注：除上表序列类型通用操作外，字符串、列表和元组均有各自的一些附加方法。详见 Python3.9 官方中文文档：https://docs.python.org/zh-cn/3.9/library/stdtypes.html#string-methods。

1.3.3 集合类型及常用操作

1. set 集合类型

set 是一种可变的集合类型，用花括号括起来定义，元素与元素之间用逗号隔开。set 集合类型是由若干不重复且无序的数据元素构成的。创建 set 集合类型的示例代码如下。

```
set1={'Jack','John','Anna','Jack'}   #使用花括号，元素以逗号分隔创建
set2=set(['Jack','John','Anna'])     #使用 set()函数创建
```

执行结果如图 1-13 所示。

名称	类型	大小	值
set1	set	3	{'Jack', 'John', 'Anna'}
set2	set	3	{'Jack', 'John', 'Anna'}

图 1-13 创建 set 集合类型对象

2. set 集合类型常用操作

set 集合类型所支持的常用操作如表 1-3 所示。

表 1-3 set 集合类型常用操作

运　算	结　果
len(s)	返回集合 s 中的元素数量（即 s 的基数）
x in s	检测 x 是否为 s 中的成员
x not in s	检测 x 是否非 s 中的成员
isdisjoint(other)	如果集合中没有与 other 共有的元素则返回 True。当且仅当两个集合的交集为空集合时，两者为不相交集合
add(elem)	将元素 elem 添加到集合中
remove(elem)	从集合中移除元素 elem。如果 elem 不存在于集合中则会引发 KeyError
discard(elem)	如果元素 elem 存在于集合中则将其移除
clear()	从集合中移除所有元素

注：set 集合类型所支持的更多操作详见 Python3.9 官方中文文档：https://docs.python.org/zh-cn/3.9/library/stdtypes.html#set-types-set-frozenset。

1.3.4 字典类型及常用操作

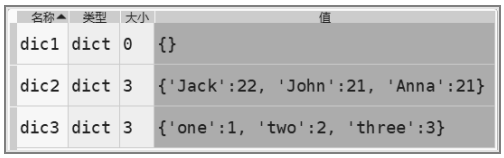
1. 字典类型

字典（dict）是一种标准的映射类型，用花括号括起来定义，元素与元素之间用逗号隔开。字典中的元素由键和值两部分组成，键在前值在后，键和值之间用冒号来区分。字典中的键必须唯一，但值不必，键可以是数字、字符串，值可以是数字、字符串或者其他 Python

数据结构（比如列表、元组等）。创建字典的示例代码如下。

```
dic1={}                                                    #创建空字典
dic2={'Jack':22,'John':21,'Anna':21}                       #使用花括号，元素以逗号
                                                            分隔创建
dic3=dict([('one', 1),('two', 2),('three', 3)]) #使用 dict()函数创建
```

执行结果如图 1-14 所示。



名称	类型	大小	值
dic1	dict	0	{}
dic2	dict	3	{'Jack':22, 'John':21, 'Anna':21}
dic3	dict	3	{'one':1, 'two':2, 'three':3}

图 1-14　创建字典类型对象

2. 字典类型常用操作

字典类型所支持的常用操作如表 1-4 所示。

表 1-4　字典类型常用操作

运　　算	结　　果
list(d)	返回字典 d 中使用的所有键的列表
len(d)	返回字典 d 中的元素个数
d[key]	返回 d 中以 key 为键的值
d[key] = value	将 d[key]的值设为 value
key in d	如果 d 中存在键 key 则返回 True，否则返回 False
get(key[, default])	如果 key 存在于字典中则返回 key 的值，否则返回 default
clear()	移除字典中的所有元素

注：字典类型所支持的更多操作详见 Python3.9 官方中文文档：https://docs.python.org/zh-cn/3.9/library/stdtypes.html#mapping-types-dict。

1.4　流程控制语句

1.4.1　条件判断语句

视频 1.4　流程控制语句微课视频

条件判断语句，是指条件成立时要执行的语句和条件不成立时要执行的语句。条件判断语句在 Python 编程语言中是基本语句，最常用的是 if 语句。if 语句包含零个或多个 elif 子句，及可选的 else 子句。关键字 elif 是 else if 的缩写。

1. if...语句

if 语句的使用方式如下。

```
if (条件判断语句):
    程序模块                #条件判断为"真"时执行的语句
```

示例代码如下。

```
x=5
import math             #导入 math 模块
if x>0:
    s=math.exp(x)       #当 x>0 时，求 e 的 x 次方
```

执行结果如图 1-15 所示。

名称	类型	大小	值
s	float	1	148.4131591025766
x	int	1	5

图 1-15　if 语句

2. if...else...语句

if...else...语句的使用方式如下。

```
if (条件判断语句):
    程序模块 1              #条件判断为"真"时执行的语句
else:
    程序模块 2              #条件判断为"假"时执行的语句
```

示例代码如下。

```
x=-9
import math             #导入 math 模块
if x>0:
    s=math.sqrt(x)      #当 x>0 时，求 x 的平方根
else:
    s=math.exp(x)       #否则（x<=0），求 e 的 x 次方
```

执行结果如图 1-16 所示。

名称	类型	大小	值
s	float	1	0.00012340980408667956
x	int	1	-9

图 1-16　if...else...语句

3. if...elif...else...语句

if...elif...else...语句的使用方式如下。

```
if (条件判断语句 1):
    程序模块 1                    #条件判断语句 1 为"真"时执行的语句
elif (条件判断语句 2):
    程序模块 2                    #条件判断语句 2 为"真"时执行的语句
else:
    程序模块 3                    #以上所有条件判断语句为"假"时执行的语句
```

示例代码如下。

```
x=0
import math                       #导入 math 模块
if x>0:
    s=math.sqrt(x)                #如果 x>0，求 x 的平方根
elif x<0:
    s=math.exp(x)                 #否则如果 x<0，求 e 的 x 次方
else:
    s=x+1                         #否则，求 x+1
```

执行结果如图 1-17 所示。

名称	类型	大小	值
s	int	1	1
x	int	1	0

图 1-17　if...elif...else...语句

1.4.2　循环语句

循环语句是用于重复执行某条语句（循环体）的语句，它包含一个控制表达式，每循环执行一次都要对控制表达式进行判断，如果表达式为真，则继续执行循环。Python 语言中常用的循环语句有 for 语句和 while 语句。

1. for 语句

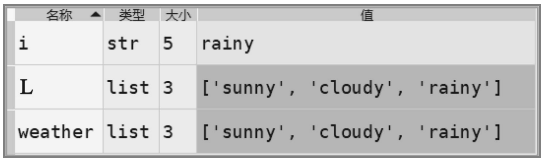
Python 的 for 语句不迭代算术递增数值，而是迭代列表或字符串等任意序列，元素的迭代顺序与在序列中出现的顺序一致。使用方式如下。

```
for 变量 in 序列:                  #for 循环遍历序列中的元素
    程序模块                       #模块中全部程序语句需要缩进并对齐
```

示例代码如下。

```
l=list()                          #定义一个空列表
weather = ['sunny','cloudy','rainy']
for i in weather:                 #for 循环遍历 weather 中的元素
    l.append(i)                   #将遍历得到的元素依次添加到列表 l 中
```

执行结果如图 1-18 所示。



名称	类型	大小	值
i	str	5	rainy
L	list	3	['sunny', 'cloudy', 'rainy']
weather	list	3	['sunny', 'cloudy', 'rainy']

图 1-18　for 语句

2. while 语句

while 循环一般需要预定义条件变量，当满足条件的时候，则循环执行循环体程序模块的内容。使用方式如下。

```
while 条件表达式:
循环体                              #循环体中的程序全部缩进并对齐
```

示例代码如下。

```
#求 1+2+3+...+100 的和
a=1                                 #定义求和运算对象变量 a，初值为 1
s=0                                 #定义求和运算结果变量 s，初值为 0
while a<=100:                       #当 a<=100 时，循环执行循环体中的语句
    s=s+a
    a=a+1
```

执行结果如图 1-19 所示。

名称	类型	大小	值
a	int	1	101
s	int	1	5050

图 1-19　while 语句

1.5　函数定义与调用

在程序设计过程中，如果若干段程序代码实现逻辑相同，那么可以考虑将这些代码定义为函数的形式。函数对象是通过函数定义创建的，对函数对象的唯一操作是调用 func(argument-list)。Python 语言中存在两种不同的函数对象：内置函数和用户自定义函数。本小节主要对用户自定义函数进行简要说明。

1. def 关键字定义函数

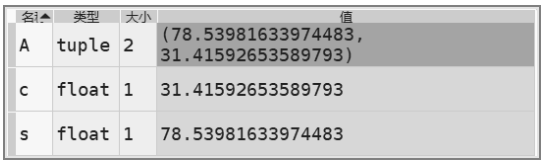
定义函数使用关键字 def，后跟函数名与括号内的形参列表。函数语句从下一行开始，并且必须缩进。使用方式如下。

```
def 函数名（形参列表）:
   函数体
   return(返回变量列表)
```

示例代码如下。

```
#定义函数
def test(r):
    import math                 #导入 math 模块
    s=math.pi*r**2              #计算半径为 r 的圆面积
    c=2*math.pi*r               #计算半径为 r 的圆周长
    return(s,c)
#调用函数
A=test(5)
s=A[0]
c=A[1]
```

执行结果如图 1-20 所示。



名称	类型	大小	值
A	tuple	2	(78.53981633974483, 31.41592653589793)
c	float	1	31.41592653589793
s	float	1	78.53981633974483

图 1-20　def 定义与调用函数

2. lambda 关键字定义函数

lambda 关键字用于创建小巧的匿名函数。lambda 函数可用于任何需要函数对象的地方。使用方式如下：lambda a, b: a+b，函数返回两个参数的和。

示例代码如下。

```
#定义函数
def make_incrementor(n):
    return lambda x: x + n
#调用函数
f=make_incrementor(10)
print(f(1))
```

执行结果如图 1-21 所示。

名称	类型	大小	值
r	int	1	11

图 1-21　lambda 定义与调用函数

本 章 小 结

本章作为 Python 大数据分析与挖掘的基础章节：首先介绍了数据分析与挖掘的内涵、联系与区别及一般流程，阐述了利用 Python 进行数据分析与挖掘的优势；其次介绍了 Python 开发环境 Anaconda 及 Spyder 的安装与使用；重点介绍了 Python 内置的 6 种基本数据类型

及常用操作；最后介绍了 Python 程序设计所需的流程控制语句及自定义函数的使用。

习题

1. 编写 Python 程序，使用 for 循环求 1+2+…+120 的和。

2. 定义一个元组 tup1=(1,2,3,'like','Python')和一个空列表 list1，以 while 循环方式，使用列表方法依次向 list1 中添加 tup1 中的元素。

即测即练

自学自测 扫描此码