第 3 章

编程基础：认识程序

程序指为了完成某种特定功能，以某种程序设计语言编写的有序指令的集合。在计算机中，CPU 只能执行二进制代码，而平常用户书写的一般是人类能够理解的编程语言。因此，要想让计算机理解用户书写的程序，就需要将程序翻译为计算机能够理解的二进制代码。根据翻译形式的不同，程序设计语言可以分为编译型语言和解释型语言。

Python 是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。Python 的设计具有很强的可读性，相比其他语言经常使用英文关键字、标点符号，Python 的语法结构更有特色。本章内容是对 Python 基础知识的概览，旨在帮助初学者快速了解 Python 语言。

- Python 环境搭建
- 变量与基本数据类型
- Python 基本编程方法

3.1 环境搭建

3.1.1 安装 Python

Python 是跨平台的，要学习 Python，首先需将其安装到计算机上。下面介绍 Windows 10 平台上的 Python 安装过程。

第 1 步　根据 Windows 操作系统版本是 64 位还是 32 位，从 Python 官网（https://www.python.org/downloads/windows）下载对应的 Python 安装文件。这里强烈推荐安装 Python 3.x 版本，因为 Python 2.x 版本马上就不再支持。

注意：安装过程中应选中 Add Python 3.X to PATH 复选框，如图 3-1 所示，单击 Install Now 即可开始 Python 安装。默认会将 Python 安装到 C:\Python3X 目录下。

第 2 步　安装完成后，单击“开始”→“运行”，输入“CMD”后回车，打开命令提示符窗口，输入 python，会出现如下两种情况。

图 3-1　Python 安装

情况一：如果出现图 3-2 所示信息，说明 Python 安装成功。提示符号>>>表示已处于 Python 交互环境中，可以输入任何 Python 代码，回车后会立刻得到执行结果。输入 exit()并回车，即可退出 Python 交互环境（直接关闭命令行窗口，或者使用快捷键 Ctrl＋C 也可以）。

```
C:\Users\IEUser>python
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37)
[MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more
information.
>>> _
```

图 3-2　Python 安装成功的提示信息

情况二：如图 3-3 显示一个错误信息，提示 Python 不是内部或外部命令，也不是可运行的程序或批处理文件。这是因为 Windows 操作系统会根据 Path 的环境变量设定的路径查找 Python.exe，如果没有找到，就会报错。多数情况下，这是由于安装 Python 过程中没有选中 Add Python 3.X to PATH 复选框。

```
C:\Users\IEUser>python
'python' is not recognized as an internal or external comm
and,
operable program or batch file.

C:\Users\IEUser>_
```

图 3-3　Python 安装不成功的提示信息

此时，需要通过以下方式将 Python.exe 所在的路径添加到 Path 中。

（1）右击“计算机”，在弹出的快捷菜单中选择“属性”命令，在打开的窗口中单击“高级系统设置”超链接。

（2）选择“系统变量”窗口下面的 Path，双击，在 Path 行添加 Python 安装路径即可。

3.1.2 安装 Anaconda

Anaconda 本质上是一个软件发行版，包含了 Conda、Python 等 180 多个科学包及其依赖项，支持 Linux、macOS、Windows 操作系统，可以很方便地解决多版本 Python 并存、切换以及各种第三方包的安装问题。安装了 Annaconda，就相当于安装了 Python、Conda 和可能用到的 NumPy、SciPy、Pandas 等常见的科学计算包，而无须再单独下载配置，如图 3-4 所示。

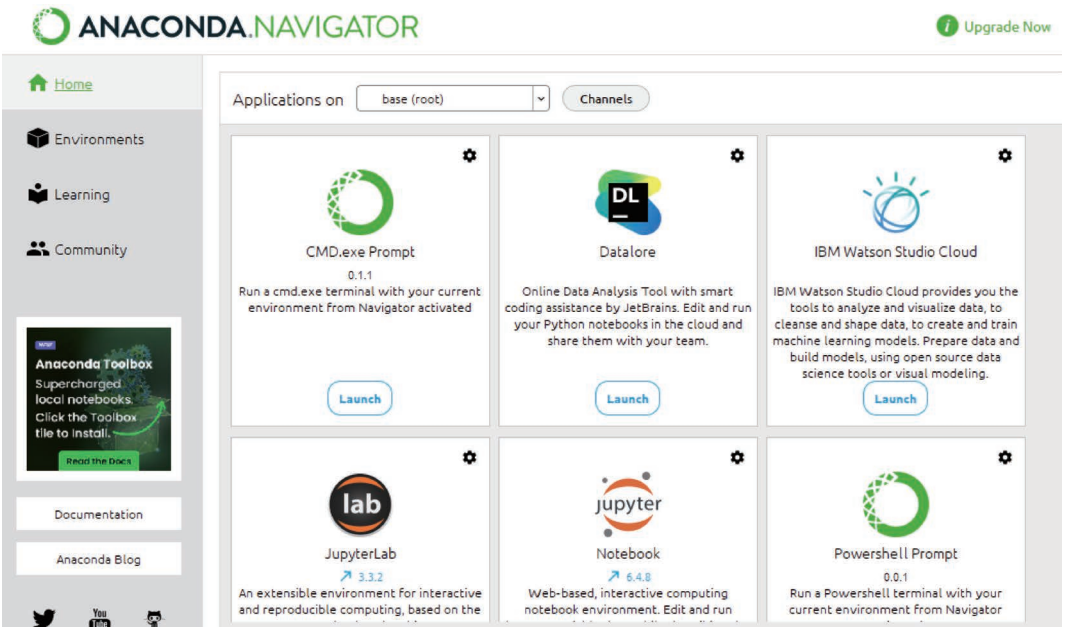

图 3-4 Anaconda 界面

Anaconda 对于 Python 初学者而言极其友好，相比单独安装 Python 主程序，选择 Anaconda 可以帮助省去很多麻烦。Anaconda 里添加了许多常用的功能包，如果单独安装 Python，则需要自行安装功能包，而在 Anaconda 中则不需要考虑这些，同时 Anaconda 还附带捆绑了两个非常好用的交互式代码编辑器（Spyder、Jupyter notebook）。

Anaconda 的安装步骤如下。

第 1 步　根据 Windows 操作系统版本是 64 位还是 32 位，从 Anaconda 官网（https://www.anaconda.com/download）下载对应的 Anaconda 安装软件，如图 3-5 所示。

第 2 步　下载 Anaconda 安装文件，双击安装文件，打开图 3-6 所示的界面，单击 Next 按钮，开始安装。

第 3 步　单击 I Agree 按钮，同意 Anaconda 的安装协议，如图 3-7 所示。

第 4 步　选择 Anaconda 的安装类型，这里默认第一个，单击 Next 按钮，如图 3-8 所示。

图 3-5　选择 Anaconda 的安装软件版本

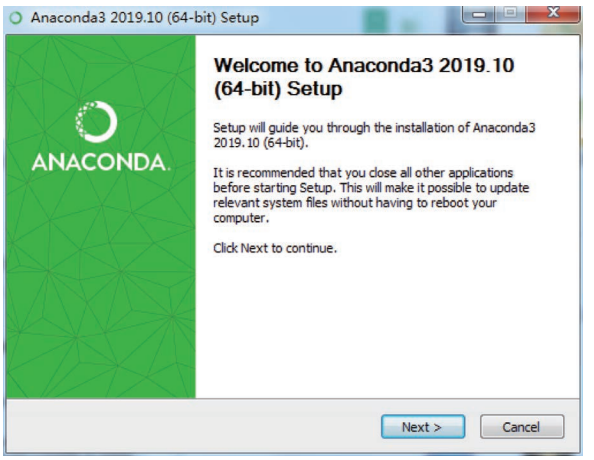

图 3-6　开始安装 Anaconda

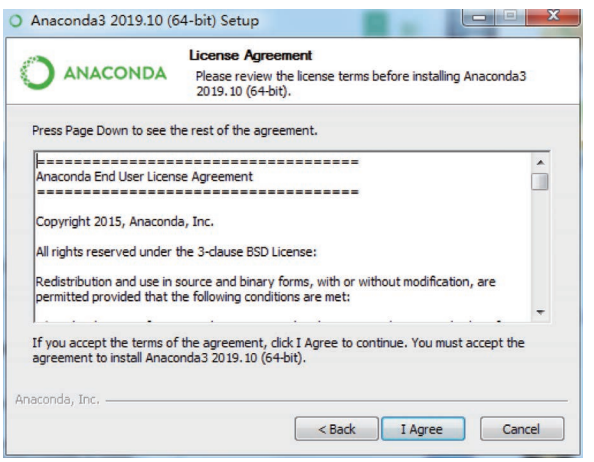

图 3-7　用户协议界面

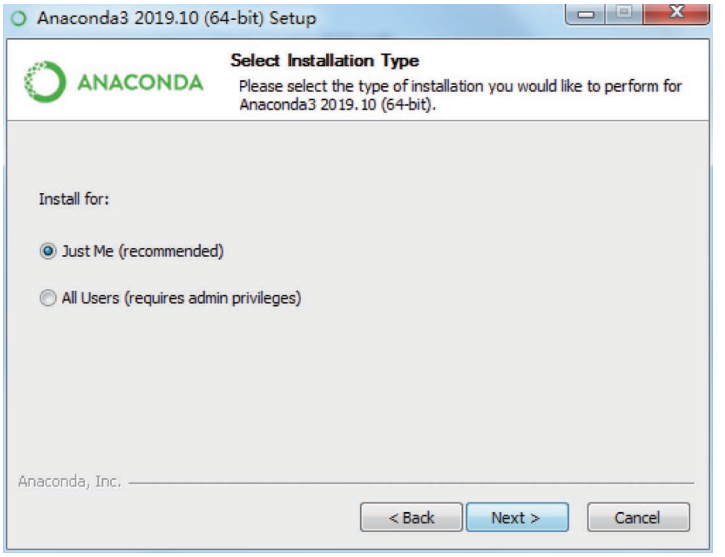

图 3-8 选择安装类型

第 5 步 选择安装位置，如图 3-9 所示。使用默认安装位置，单击 Next 按钮。

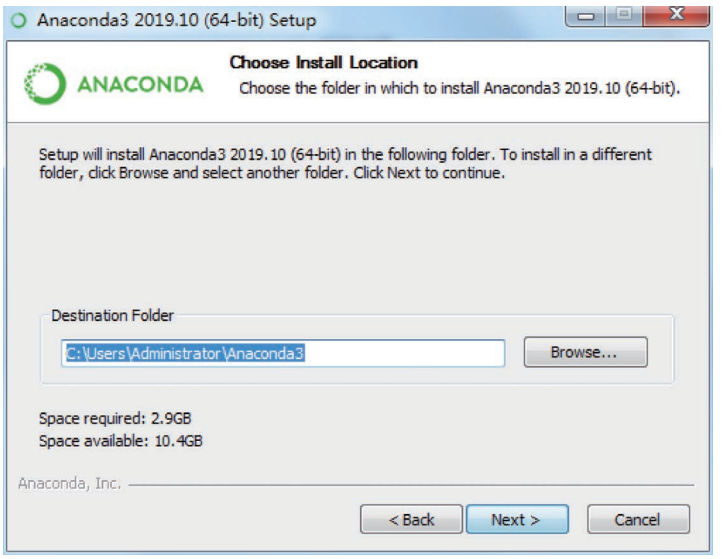

图 3-9 选择安装位置

第 6 步 安装高级选项，这里默认选择第二项，如图 3-10 所示，单击 Install 开始安装。

第 7 步 正在安装 Anaconda，界面显示安装进度，如图 3-11 所示。

第 8 步 Anaconda 安装成功，如图 3-12 所示。

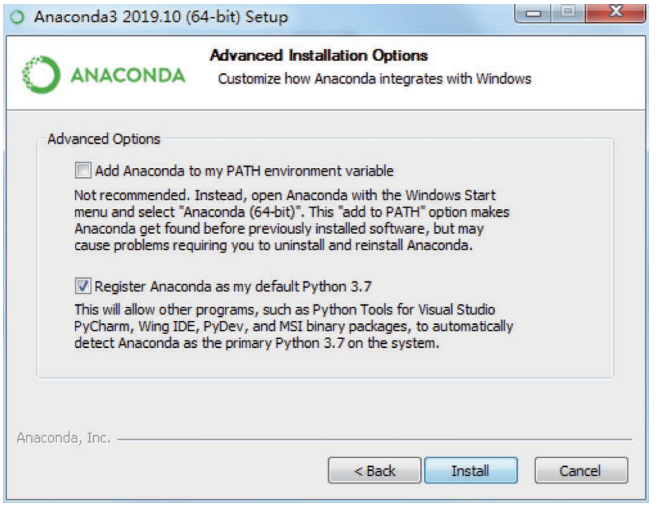

图 3-10　选择安装高级选项

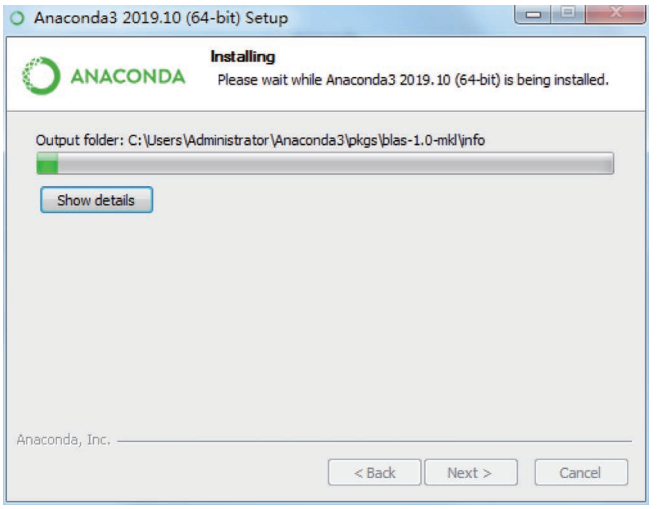

图 3-11　显示安装进度

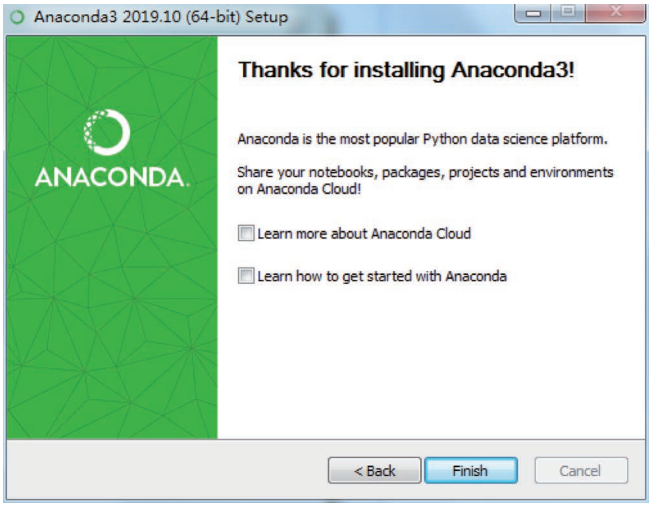

图 3-12 Anaconda 安装完成

3.2 输入和输出

1. 输出

产生输出的最简单方法是使用 print()函数，将要打印查看结果的字符串放入到括号内。例如，要输出“hello，world”，用代码实现如下：

```
>>> print('hello, world')
```

print()函数通过逗号“，”分隔零个或多个字符串，使其可以连成一串输出。print()函数会依次输出每个对象，每遇到一个逗号“，”，就输出一个空格。例如：

```
>>> print("Python is really a great language,", "isn't it?")
Python is really a great language, isn't it?
```

print()函数也可以输出整数，或者计算结果。例如：

```
>>> print(300)
300
>>> print(100 + 200)
300
```

因此，可以把计算 100 + 200 的结果输出得更美观一点：

```
>>> print('100 + 200 = ', 100 + 200)
100 + 200 = 300
```

注意，对于 100 + 200，Python 解释器自动计算出结果 300。但是，“100＋200＝”是字

符串而非数学公式，因此 Python 将其视为字符串。请读者自行解释上述输出结果。

2. 输入

Python 提供了 input()内置函数，可以从标准输入读入一行文本，默认的标准输入是键盘。input()函数可以接收一个 Python 表达式作为输入，并返回运算结果。例如，输入用户的名字：

```
>>> name = input()
CYH
```

当输入 name=input()并按 Enter 键后，Python 交互式命令行即在等待用户输入。这时，用户可以输入任意字符，按 Enter 键完成输入。例如：

```
>>> name
'CYH'
```

有了输入和输出，就可以把前面输出“hello，world”的程序改写为有意义的程序，如下：

```
>>> name = input()
>>> print('hello,', name)
```

运行上述程序，第一行代码会让用户输入任意字符作为自己的名字，并存入 name 变量中；第二行代码会根据用户的名字向用户说 hello。但是程序运行时，没有任何信息提示用户。幸好，input()可以显示一个字符串来提示用户，于是把代码改成：

```
>>> name = input('please enter your name: ')
>>> print('hello,', name)
```

再次运行该程序会发现，程序会首先输出“please enter your name:”，此时用户即可根据提示进行输入。

3.3　变量

变量是编程中最基本的存储单元，变量存储内存中的值，这就意味着在创建变量时会在内存中开辟一个空间。对于基于变量的数据类型，解释器会分配指定内存，并决定什么数据可以被存储在内存中。因此，变量可以指定不同的数据类型。在计算机程序中，这些变量可以存储整数或浮点数，还可以是字符串。

每个变量在内存中创建时，都包括变量的标识、名称和数据等信息。每个变量在使用前都必须赋值，只有赋值以后该变量才会被创建。等号“=”用来给变量赋值，等号“=”左边是一个变量名，等号“=”右边是存储在变量中的值。例如，英国作家道格拉斯·亚当斯(Douglas Adams)在著名科幻小说《银河系漫游指南》中虚构了一个故事：一台名为 Deep Thought 的计算机，通过 750 万年的计算，得到了生命、宇宙以及一切问题的答案是 42，如果用编程语言来表达，就是图 3-13 所示等式，一个称为 answer 的变量被赋值为 42。

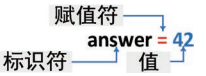

图 3-13 变量赋值

3.4 数据类型

在 Python 程序设计中，内存中存储的数据可以有多种类型，每个值对应一种数据类型。例如，一个人的年龄可以用整数来存储，名字可以用字符来存储。但是，用户并不需要声明变量的数据类型，Python 会根据每个变量的初始赋值情况分析其类型，并在内部对其进行跟踪。Python 定义了 5 种标准的数据类型，包括 Number（数值类型）、String（字符串）、List（列表）、Tuple（元组）、Dictionary（字典），用于存储各种类型的数据。

3.4.1 数值类型

Number 数值类型用于存储数值，Python 3 支持以下 4 种不同的数值类型。

（1）整型（Integer）：是不包含小数部分的数值型数据，是正或负整数，不带小数点。Python 3 的整型类型没有限制大小，可以当作 Long 类型使用。

（2）浮点型（Floating Point Number）：或称为浮点数，表示实数，由一个小数点分隔的整数部分和小数部分组成。浮点数也可以使用科学记数法表示。

（3）复数（Complex Number）：由实数部分和虚数部分构成，形式如 a＋bj，或者 complex(a,b)，其中复数的实部 a 和复数的虚部 b 都是浮点型。

（4）布尔型（Boolean）：Python 3 中把 true 和 false 定义成关键字，但它们的值还是 1 和 0。

有时需要对数值内置类型进行转换，此时只需将数值类型作为函数名即可。数据进行类型转换时有如下 4 个函数可以使用：

（1）int(x)：将 x 转换为一个整数。

（2）float(x)：将 x 转换为一个浮点数。

（3）complex(x)：将 x 转换为一个复数，实数部分为 x，虚数部分为 0。

（4）complex(x,y)：将 x 和 y 转换为一个复数，实数部分为 x，虚数部分为 y。x 和 y 是数字表达式。

下面看几个常用的数值类型运算：

```
>>> 2 / 4          # 除法，得到一个浮点数 0.5
0.5
>>> 2 // 4         # 整除，得到一个整数 0
0
>>> 17 % 3         # 取余，得到一个整数 2
```

```
2
>>> 2 ** 5          #乘方,得到一个整数 32
32
```

3.4.2　字符串

字符串是 Python 中常用的数据类型，是一个有序的字符集合。用户可以使用引号('或")创建字符串。创建字符串的方法很简单，只需为变量分配一个值即可。例如：

```
1 >>> str1 = "Hello Python!"
```

Python 的字符串列表有两种取值顺序：

(1) 从左到右索引默认从 0 开始，最大范围是字符串长度减 1。

(2) 从右到左索引默认从－1 开始，如图 3-14 所示。

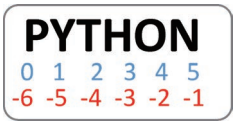

图 3-14　字符串索引

如果要实现从字符串中获取一段子串，可以使用[头下标:尾下标]截取相应的字符串。截取字符串的语法格式如下：

```
变量[头下标:尾下标]
```

或

```
变量[头下标:尾下标:步长]
```

[头下标:尾下标] 获取的子字符串包含头下标的字符，但不包含尾下标的字符。例如：

```
>>> str = 'Python'
>>> print (str)                #输出字符串
Python
>>> print (str[0:-1])          #输出第一个到倒数第二个所有字符
Pytho
>>> print (str[2:5])           #输出从第 3 个开始到第 5 个字符
tho
>>> print (str[2:])            #输出从第 3 个开始其后所有字符
thon
```

3.4.3　列表

列表是 Python 中使用较频繁的数据类型，是 Python 中的内置有序序列。列表可以实现大多数集合类的数据结构，如图 3-15 所示。

图 3-15 列表样例

列表支持字符、数字、字符串，甚至还可以包含列表(嵌套)。列表的所有元素放在一对中括号“[]”中，并使用逗号分隔。列表中值的切割也可以使用变量［头下标:尾下标］，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)，如图 3-16 所示。

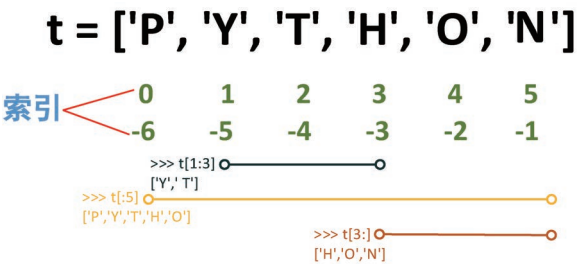

图 3-16 列表索引

列表可以进行赋值、删除、添加等修改性操作。当列表元素增加或删除时，列表对象自动扩展或释放内存，保证元素之间没有缝隙。列表可以包含不同类型的元素，但是通常所有的元素具有相同的类型。例如：

```
>>> t = [1,2,3,4]
>>> t[1] = 4
>>> t
1,4,3,4
>>> del t[0]
>>> t
4,3,4
>>> t[:1] = []
>>> t
3,4
```

3.4.4 元组

Python 的元组与列表类似，但是元组不能二次赋值，相当于只读列表，因此元组常常又被称为“不可变列表”。元组使用小括号()表示，如图 3-17 所示。

元组创建很简单，只需在括号中添加元素，并使用逗号隔开。例如：

```
>>> tup1 = ('physics', 'chemistry', 1997, 2000)
>>> tup2 = (1, 2, 3, 4, 5 )
```

图 3-17　元组样例

以下操作对于元组是无效的，因为元组中的元素值不允许修改，而列表允许更新。

```
tuple = ('hello', 1 , 2, 'python')
list = ['hello', 1 , 2, 'python']
tuple[2] = 3                    #元组中是非法操作
list[2] = 3                     #列表中是合法操作
```

但用户可以对元组进行连接。例如：

```
>>> tup1 = (12, 34)
>>> tup2 = ('abc', 'xyz')
>>> tup3 = tup1 + tup2
>>> tup3
(12, 34, 'abc', 'xyz')
```

3.4.5　字典

程序设计中，其实很多概念是来自现实生活中的原型。Python 中的字典就如现实世界中的字典一样，使用“名称：内容”进行数据的构建，在 Python 中分别对应“键（key）：值（value）”，习惯称之为键值对，如图 3-18 所示。

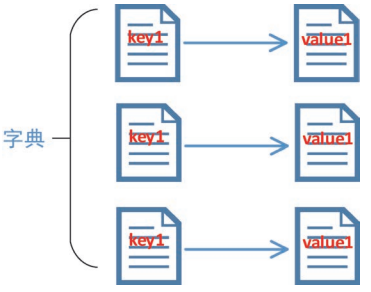

图 3-18　字典样例

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，而不是通过偏移存取。字典是一种映射类型，是一个无序的键值对“键（key）：值（value）”的集合，每个键值对之间用逗号“,”分隔，整个字典包括在大括号“{}”中。字典的格式如下：

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

键（key）一般是唯一的，如果重复，则最后一个键值对会替换前面的，值不需要唯一。

例如：

```
>>> dict = {'a': 1, 'b': 2, 'b': '3'}
>>> dict['b']
'3'
>>> dict
{'a': 1, 'b': '3'}
```

3.5 Python 编程

3.5.1 控制流

控制流元素非常重要，可以在程序中包含有意义的业务逻辑。很多商务处理和分析依赖于业务逻辑，如“如果客户的花费超过一个具体值，就……”或“如果销售额属于 A 类，则编码为 X；如果销售额属于 B 类，则编码为 Y；否则编码为 Z”，这些逻辑语句在代码中就可以用控制流元素表示。Python 提供了若干种控制流元素，包括 if-elif-else 语句、for 循环、while 循环和 break 与 continue 语句。

1. if 条件控制

条件控制就是对 if-else 的使用。正如它们的名字所示，if-else 语句提供的逻辑为“如果这样，那么就……，否则……”。else 代码块（code block）并不是必需的，但可以使代码更加清楚。if-else 语句的基本结构如图 3-19 所示。

图 3-19 if-else 语句的基本结构

if-else 语句中，条件（condition）指的是成立的条件，即返回值为 true 的布尔表达式。

一般情况下，设计程序时需要考虑逻辑的完备性，并对用户可能会产生困扰的情况进行预防性设计，这时就会形成多条件判断。多条件判断同样很简单，只需在 if 和 else 之间增加 elif 即可，用法和 if 一致。多条件判断是依次进行的，首先判断条件是否成立，如果成立就运行内部代码，如果不成立就顺次判断下面的条件是否成立，如果不成立则运行 else 对应的语句。if-else 多条件判断的基本结构如图 3-20 所示。

【例 2-1】 已知某课程的百分制分数 mark，将其转换为五级制（优、良、中、及格、不及格）的评定等级 grade，评定条件如图 3-21 所示。

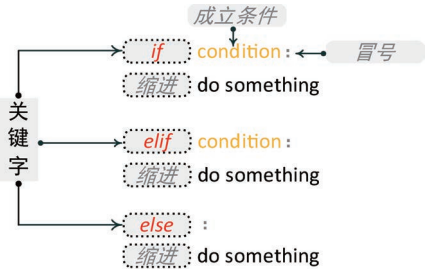

图 3-20　if-else 多条件判断的基本结构

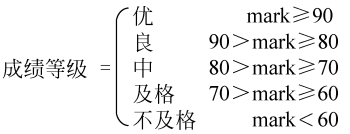

图 3-21　分数五级制

根据评定条件，其可以通过如下多条件控制流实现。

```
>>> mark = int(input("请输入分数:"))
>>> if (mark >= 90):
        grade = "优"
    elif (mark >= 80):
        grade = "良"
    elif (mark >= 70):
        grade = "中"
    elif (mark >= 60):
        grade = "及格"
    else:
        grade = "不及格"
>>> print(grade)
```

在上面的代码中可以清晰地看到代码块。代码块的产生是由于缩进，即具有相同缩进量的代码实际上是在共同完成相同层面的事情。另外，if…elif …elif…序列用于替代其他语言中的 switch 或 case 语句。

2. for 循环

可迭代对象(Iterables)一次可以返回一个元素，因此可以适用于循环。Python 包括如下几种常用的可迭代对象：列表、元组、字典或字符串。Python 3 中的内置对象 range 可以产生指定范围的数字序列，格式如下：

```
range(start, stop[, step])
```

range 返回的数字序列从 start 开始，到 stop 结束(不包含 stop)。range 可以指定可选的步长 step。例如：

```
>>> for i in range(1,11):
>>> print(i, end = ' ')          #输出:1 2 3 4 5 6 7 8 9 10
>>> for i in range(1,11,3):
>>>     print(i, end = ' ')      #输出:1 4 7 10
```

for 语句可以遍历可迭代对象集合中的元素，如遍历列表、元组或字符串等包含的元素。也可以使用 range()函数与 len()函数一起作用于列表，生成一个索引序列，用在 for 循环中。for 语句的基本结构如图 3-22 所示。

图 3-22 for 语句的基本结构

【例 2-2】 利用 for 循环求 1～100 中所有奇数的和以及所有偶数的和。

```
1  >>> sum_odd = 0; sum_even = 0
2  >>> for i in range(1,101):
           if i % 2 != 0:
               sum_odd = sum_odd + i
           else:
               sum_even = sum_even + i
3  >>> print("1 - 100 之间奇数的和是:", sum_odd)
#输出:1 - 100 之间奇数的和是:2500
4  >>> print("1 - 100 之间偶数的和是:", sum_even)
#输出:1 - 100 之间偶数的和是:2550
```

在编程中还有一种常见的循环，即嵌套循环或称为多重循环。嵌套循环是指一个循环体内又包含另一个完整的循环结构。在嵌套循环中，两种循环语句(for 循环和 while 循环)可以相互嵌套。

3. while 循环

Python 编程中，while 语句也可以用于循环执行程序，即在某条件下循环执行某段程序，以处理需要重复处理的相同任务。但是，for 循环用于集合中每个元素的一个代码块，集合中元素被穷尽遍历时停止；而 while 循环可以不断进行，直到不满足指定条件为止。因此，while 的作用概述成一句话就是“只要…条件成立，就一直做…”。while 循环的基本结构如图 3-23 所示。

图 3-23 while 循环的基本结构

【例 2-3】 while 循环示例：给定指定包含整数集合的列表，分别产生奇数列表和偶数

列表。

```
>>> numbers = [12, 17, 3, 20, 8, 7]
>>> even = []
>>> odd = []
>>> while len(numbers)> 0:
        number = numbers.pop()
        if(number % 2 == 0):
            even.append(number)
        else:
            odd.append(number)
```

4. break 与 continue 语句

break 语句用于提前终止 for 或 while 循环语句，即循环条件没有 false 条件或者序列没有被完全递归完也会停止执行循环语句，接着执行循环语句的后继语句。continue 语句类似于 break 语句，也必须在 for 或 while 循环中使用，但是 continue 语句仅用于跳出本次循环。continue 语句用来告诉 Python 跳过当前循环的剩余语句，然后继续进行下一轮循环。break 与 continue 语句如图 3-24 所示。

```
sites = ['baidu', 'qq', 'weixin', 'toutiao', 'wiki']

for site in sites:
    if len(site) != 4:
        continue

    print(f'Hello, {site}')

    if site == 'wiki':
        break
print('Done!')
```

图 3-24　break 与 continue 语句

continue 语句与 break 语句的区别：continue 语句仅结束本次循环，并返回循环的起始处，如果循环条件满足，就开始执行下一次循环；而 break 语句则是结束循环，跳转到循环的后继语句执行。

3.5.2　函数

前面已经介绍了部分函数，其中 print() 函数和 input 函数两个基本函数如图 3-25 所示。

函数是组织好的、可重复使用的、用来实现单一或相关联功能的代码段。函数能提高应用的模块性和代码的重复利用率。Python 中函数的应用非常广泛，前面我们已经接触过多个函数，如 input()、print()、range() 函数等，这些都是 Python 的内置函数，可以直接使用。

Python 自带的函数数量是有限的，除了可以直接使用的内置函数外，Python 还支持自

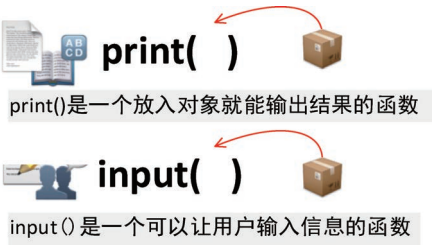

图 3-25 基本函数

定义函数，帮助我们做更多的事情。设计符合使用需求的函数，即将一段有规律的、可重复使用的代码定义成函数，从而达到一次编写、多次调用的目的，以下是简单的规则：

（1）函数代码块以 def 关键词开头，后接函数标识符名称和圆括号()。

（2）任何传入参数和自变量必须放在圆括号中。圆括号之间可以用于定义参数。

（3）函数的第一行语句可以选择性地使用文档字符串(用于存放函数说明)。

（4）函数内容以冒号起始，并且缩进。

（5）return [表达式] 结束函数，选择性地返回一个值给调用方；不带表达式的 return 相当于返回 None。

我们读一遍：Define a function named 'function' which has two agruments：arg1 and arg2，returns the result 'Something'，是不是很易读很顺畅？代码的表达比英文句子更简洁，具体的语法格式如图 3-26 所示。

图 3-26 函数定义

调用函数的方法就是直接写出自定义的函数名，如果有参数，还需要指定实际传入的参数值。函数调用的语法格式如下：

```
函数名([实参列表])
```

【例 2-4】 定义计算并返回第 n 阶调和数(1＋2＋3＋…＋n)的函数，从命令(input)获取所需输出的调和数个数 n，输出前 n 个调和数。

```
>>> def harmonic(n):                          #计算 n 阶调和数(1 + 2 + 3 + … + n)
        total = 0.0
        for i in range(n):
            total += i
        return total
>>> n = int(input("输出的调和数个数 n:"))      #三角形行数
```

```
>>> for i in range(1, n + 1):
        print(harmonic(i))
```

本章小结

Python 是一种面向对象的编译型、解释型的计算机程序设计高级语言，提供高效的高级数据结构，成为多数平台上写脚本和快速开发应用的编程语言。本章作为 Python 的基础知识部分，首先对编程环境的配置进行了介绍，学习了 Python 及其发行版 Anaconda 的安装方法；其次，介绍了 Python 语言的输入输出、变量以及 5 种标准的数据类型，包括数值类型、字符串、列表、元组、字典；在流程控制语句方面，介绍了条件语句、循环语句；在 Python 函数方面，介绍了自定义函数的创建。

习题

1. Python 包括哪些不可变序列数据类型？哪些可变序列数据类型？

2. 输入三角形的 3 条边 a、b、c，判断此 3 边是否可以构成三角形。若能，则进一步判断三角形的性质，即为等边、等腰、直角或其他三角形。

3. 随便给定一个在一定范围内的数字，让用户猜该数字是多少，并输入自己猜测的数字，系统判断是否为给定数字。如果猜测数字大于给定值，则提示“输入值过大”；否则，提示“输入值过小”；如果等于给定数字，就提示“猜对了”，并展示猜了多少次才猜中。