2.4 Python的开发环境搭建

1. Python3.9.3软件安装,在以下网页找到相应系统和位数的版本下载

https://www.python.org/downloads/

2. 安装和环境变量配置，安装时选 customize installation, 并且在add python 3.9.3 to path 前面打√，后一步的各个选项前全都打√

3. 测试Python

按【win】+【r】键，进入cmd，输入python，按回车键，进入Python的交互式终端，如图2-7所示。在Python命令提示符>>>后可输入Python命令代码，按回车键后直接运行。

2.5.1 PyCharm安装与使用

1. PyCharm简介

PyCharm是由JetBrains公司开发的一款Python的集成开发环境软件。

2. PyCharm安装

①在jetbrains官网（https://www.jetbrains.com/pycharm/download/#section=windows）上下载社区版本的PyCharm软件，软件名称为pycharm-community-2017.3.exe 。

② 双击pycharm-community-2017.3.exe，打开PyCharm软件安装界面，参考教材安装说明进行安装和启动 PyCharm。

2.5.1 PyCharm安装与使用

3. PyCharm简单设置，如更换主题、修改源代码字体大小、编码设置等。

【操作演示】

4 PyCharm使用

① 新建项目

操作步骤：打开PyCharm，单击“File->New Project”，操作演示。

② 创建Python文件

操作步骤：右击项目名称，选择“New->Python File”，操作演示。

③ 编写和运行Python程序。

【操作演示】

4.1.2 NumPy安装与测试

1. 测试Python环境中是否安装了NumPy

在Windows操作系统下，按【win】+【r】键，进入cmd命令窗口，输入“python”命令，按回车键，进入Python命令窗口。在Python命令窗口中输入“from numpy import *”命令，如果在命令窗口中出现“ModuleNotFoundError:No module named ‘numpy’”的错误提示，则需要安装NumPy。

4.1.2 NumPy安装与测试

- 在Windows操作系统下安装NumPy方法:

pip3 install numpy scipy matplotlib pandas -i https://pypi.tuna.tsinghua.edu.cn/simple

pip3 install C:\Program Files\Python310\Scripts\numpy-1.10.0-cp26-cp26m-manylinux1_x86_64.whl (此方法必须先从https://pypi.tuna.tsinghua.edu.cn/simple 下载软件包放在C:\Program Files\Python310\Scripts\)

PyCharm 安装 NumPy方法:

File →Settings→Project→project interpreter→package 上方的“+”，在左侧窗口选中NumPy安装

4.1.3 SciPy 简介及其安装与测试

SciPy是世界著名的Python开源科学计算库，它是建立在NumPy基础之上，增加了众多的数学、科学以及工程计算中常用的库函数。它增加的功能包括插值、积分、最优化、统计、线性代数、傅里叶变换、图像处理和常微分方程求解器等一些专用函数。

与NumPy库相同，SciPy也是需要单独安装，安装步骤如下。

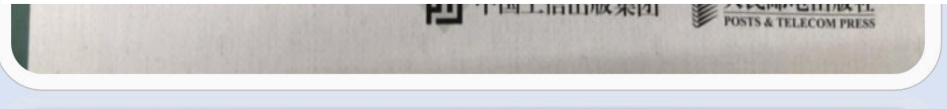
1. 测试Python环境中是否安装了SciPy
2. 在Windows操作系统下安装SciPy
3. PyCharm 安装SciPy

4.1.4 NumPy 的简单应用：一维数组相加

在科学计算中，常常会遇到数组和矩阵的计算，如有两个一维数组x和y，其中，x的值为0-10的整数，y的值为0-10的整数的平方，需要计算x与y的一维数组相加。

1. 利用Python的循环语句来实现两个一维数组相加示例
2. 利用NumPy实现两个一维数组相加的示例
3. 比较两种方法的计算速度

```
# -*- coding: UTF-8 -*-
def python_sum(n):
    x = list(range(n))        #range函数产生是包含0-n的整数元组，用list函数转换成列表
    y = list(range(n))
    z = []
    for i in range(len(x)):          #for语句
        y[i] = i**2
        z.append(x[i]+y[i])
    print(x)
    print(y)
    return z

a = python_sum(4)
print (a)
```

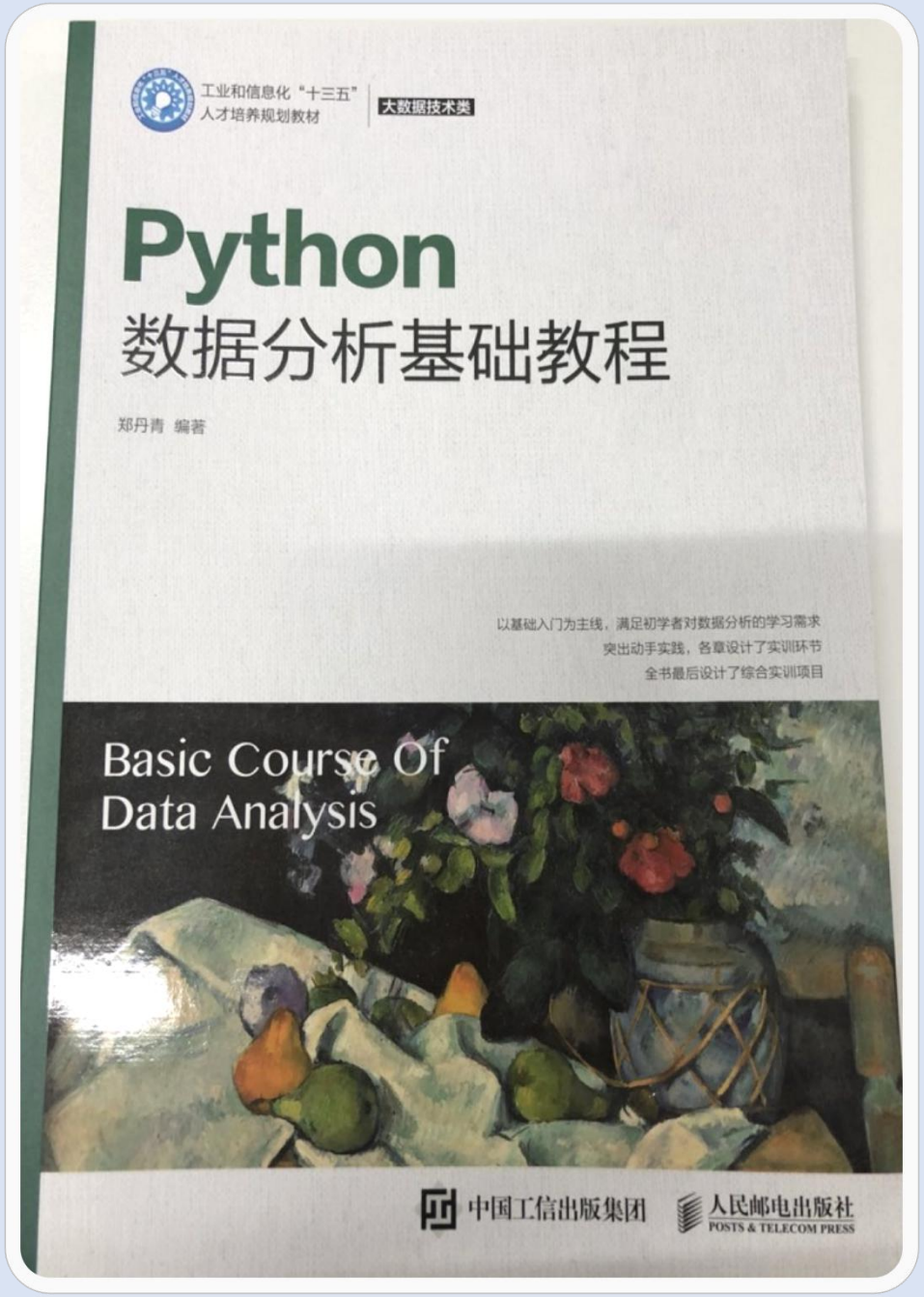



```
# -*- coding: UTF-8 -*-
import numpy as np  #numpy
def numpy_sum(n):
    x =np.arange(n)
#arange函数创建包含0-n的整数数组
    y = np.arange(n)**2
    z = x + y
    print(x)
    print(y)
    return z

a = numpy_sum(4)
print (a)
```

```
# 测试python函数执行时间
from datetime import datetime

n=100000
start_time = datetime.now()
python_sum(n)
end_time = datetime.now()
time_interval = end_time-start_time

print (time_interval)
```

中国工信出版集团 人民邮电出版社 POSTS & TELECOM PRESS

```
# 测试python函数执行时间
from datetime import datetime

n=100000
start_time = datetime.now()
b=numpy_sum(n)
end_time = datetime.now()
time_interval = end_time-
start_time

print (time_interval)
```



第3章 Python语言基础

pip install --user --upgrade

学习目标：

- 掌握Python的基础语法。
- 掌握Python 的数据类型。
- 掌握Python流程控制语句的使用方法。
- 掌握Python函数的使用方法。
- 掌握数据结构、流程控制语句和函数的综合运用。

3.1.1 Python的语法规则

1. Python编码设置

Python3安装后，系统默认其源码文件为UTF-8编码，但是如果编辑器不支持UTF-8编码文件，或者要使用其编码方式，就要为源文件指定特定的字符编码。此时，需要在文件首行或者第二行插入一行特殊的注释行作为字符编码声明，其格式如下。

```
# -*- coding: UTF-8 -*-
```

或者：

```
#coding: UTF-8 和 #coding = UTF-8
```

3.1.1 Python的语法规则

2. 代码注释

在程序设计中，常常要对程序代码添加注释，Python的代码注释有下：

① 单行注释

单行注释通常是以#号开头，在#号后面紧接的是注释说明的文字。

示例

```
# 这是输出语句（单行注释）
print ("你好！欢迎到Python社区来！"); # 这是输出语句（单行注释）
```

注意：注释行是不会被机器解释的。编码声明也是以#号开头，但它不是代码注释行，而且编码声明要放在首行或第二行才能被机器解释。

3.1.1 Python的语法规则

② 多行注释

在实际的程序文档中常常需要进行多行注释，多行注释有以下两种方法。

- 在每一行前加#号即可。
- 使用3个单引号或3个双引号将注释的内容括起来。

示例

```
'''
这是使用3个单引号的多行注释
这是使用3个单引号的多行注释
'''
print ("你好！欢迎到Python社区来！");
```

3.1.1 Python的语法规则

3. Python标识符

（1）Python标识符是由英文字母、数字及下划线（_）组成。

（2）Python中的标识符不能以数字开头。

（3）Python标识符是区分大小写的。

（4）以下划线开头的标识符是有特殊意义的，参见教材说明。

4. Python**关键字**

Python中的关键字见教材表3-1。

5. 缩进代码

Python最具特色的就是用缩进方式来标识代码块。在同一个代码块的语句必须保证相同的缩进空格数，否则，程序会报错。

3. 1. 1 Python的语法规则

6. 多行语句

在Python编程中，如果一条语句太长，可以使用反斜杠（\）将一行的语句换行分为多行，而不会被机器识别为多条语句。

但是在Python中，如果[], {} 或 () 括号里面有多行语句，在换行时不需要使用反斜杠（\），而是使用逗号换行即可。

如果一行包含多条语句，则使用分号(;)对多条语句分隔。

3.1.1 Python的语法规则

7. Python空行

在Python中，一般在函数之间或类的方法之间用空行分隔，以表示一段新的代码的开始。类和函数入口之间也可以用一行空行分隔，以突出函数入口的开始。

空行的作用是用于分隔两段不同功能或含义的代码，以便于代码的阅读、维护或重构。

8. 用户输入函数

在Python中用户输入信息的语句是input()函数。

9. 代码组

缩进相同的一组语句构成一个代码块，也称之代码组。

3.1.1 Python的语法规则

10. print()输出函数

print()函数默认输出是换行的，如果要实现不换行输出就需要在变量末尾加上end=“”，参见教材示例。

11. import与from...import语句

在Python中可以使用import或者from...import语句来导入相应的模块。import与from...import的作用如表3-2。

表 3-2 import 与 from…import 的作用

语句格式	作用
import modulename	将整个模块导入（modulename 模块名）
from modulename import functionname	从某个模块中导入某个函数（functionname 函数名）
from modulename import firstfunc, secondfunc	从某个模块中导入多个函数（firstfunc、secondfunc 均为函数名）
from modulename import *	将某个模块中的全部函数导入

3.1.2 常量、变量与标准数据类型

1. 常量

Python中的常量，就如同数字3，2.34,1.3e-2和字符串“this is a string”一样，常量的值是不能改变的。

2. 变量

（1）变量的特性

Python中的变量是指存储在计算机内存中的值。通过id()函数可以查看创建变量和变量重新赋值时内存空间的变化过程。

在Python中，变量是不需要提前声明，创建时直接对其赋值即可，变量的数据类型是由赋值给变量的值决定。创建一个变量，首先要定义变量名和变量值，然后再通过赋值语句将变量值赋值给变量名。

3. 1. 2 常量、变量与标准数据类型

2. 变量

（2）变量的命名规则

（3）变量值就是赋值给变量的数据。

（4）变量赋值

通过单变量赋值创建变量和输出变量,除了单变量赋值外，Python还允许为多个变量赋值相同的值，其格式：a = b = c = 1

另外，Python还可以为多个对象指定多个变量，

其格式：a, b, c = 1, 2, "john”

3. 1. 2 常量、变量与标准数据类型

3. 标准数据类型

Python 定义了一些标准的数据类型，以用于存储各种类型的数据。其中，Python的六个标准的数据类型分别为numbers（数字）、string（字符串）、list（列表）、tuple（元组）、dictionary（字典）和集合（set）等。

3. 1. 3 第一个Python程序

利用PyCharm开发工具编写第一个Python的脚本程序的方法。

【例3-1】设计一个Python的脚本程序，该程序的功能是：当用户输入的成绩大于85分，则输出“优秀学生”，否则输出“合格学生”，并设置程序编码方式为UTF-8。

3.2.1 数字

1. Python3的数值型数据类型

Python 中的数字是用于存储数值的，Python3支持的数值型数据类型有int（整数）、float（浮点数）、bool（布尔型）和complex（复数）。而数值型数据类型的表示方法见表3-3。

表3-3数值型数据类型

数据类型	表示方法	示例
int	整数表示是正整数或负整数	123，-450
float	浮点数表示有小数位的数值，它由整数部分与小数部分组成	124.36，-46.12
bool	布尔型表示逻辑真或逻辑假	True，False
complex	复数由实数部分和虚数部分构成，可以用 a + bj 或者 complex(a,b)表示，其中 a 是实数，b 是虚数	3+4j，3.0+4.0j，4j

3.2.1 数字

2、数字创建与删除

当给变量赋值一个数字时，数字（Number） 对象就会被创建。如果要删除这些对象引用可使用del语句。del语句的语法：

del var1[,var2[,var3[....,varN]]]

3、数字类型转换

在Python 中，如果需要对数据内置的类型进行转换，只需要将数据类型作为函数名即可。

如在Python交互式终端的命令提示符后输入var = 10.2 ;int(var)，则会输出10整数。

3.2.1 数字

4、数字运算

数字运算有+（加）、-（减）、*（乘）、/（除）或//（除）、%（取余）、**（乘方）。如6+7、6-4、3*2、6/4（6除4，返回是浮点数）、6//4（6除4，返回是整数）、6%4（6除4取余）、6**2（6的2次方）。

【例3-2】说明Python中数字的运用。具体示例参见教材。

3.2.2 字符串

Python中的字符串是一个有序的字符集合，是用于存储表示文本的信息。

1. 字符串定义

在Python中，可以使用单引号(‘)，双引号(“)或三引号(’‘’ ”“”)来标识字符串，引号的开始与结束必须是相同类型的。具体示例参见教材。

2．Python转义字符

在Python的字符串中如需要使用特殊字符时，可使用反斜杠(\)转义字符来定义，如换行符用”\n”表示。Python中常用的转义字符见表3-4。

3.2.2 字符串

3. 字符串索引

字符串的索引分为正索引和负索引，通常说的索引就是指正索引。在Python中，正索引是从左到右去标记字母，索引是从0开始，也就是说从左边开始第一个字母索引是0，第二个字母索引是1，依此类推。负索引是从右到左去标记字母，然后加上一个负号（-），负索引的第一个值是-1，也就是说从右边开始第一个字母索引是-1，第二个字母索引是-2，依此类推。

3.2.2 字符串

4. 字符串的操作

（1） 字符串提取

字符串的提取就是通过索引获取字符串中指定位置的字符。字符串提取的方法是在变量名后使用方括号([ ])将需要提取的字符索引放置在方括号中即可。

例如变量var='Hello',如果要提取出字符串中字母“H”和“e”，可用var[0]输出“H”字母，用var[1]输出“e”字母。

3.2.2 字符串

4. 字符串的操作

（2）字符串切片

字符串切片方法是变量名后使用方括号([i:j])，其中i表示截取字符串的开始索引，j表示结束索引。注意：在截取字符串时将包含起始字符，但不包含结束字符，这是一个半开闭区间。同时， Python还提供了一个len()内置函数用于返回字符串（或其他有长度对象）的长度。

例如变量var='Hello',如果要截取出子字符串“ell”，则使用var[1:4]即可。而使用var[2:5]则输出 “llo” 子字符串。

注意：在Python中，字符串是不可以更改的。

3.2.2 字符串

4. 字符串的操作

（3） 字符串拼接

字符串拼接也就是将字符串连接，字符串连接的方法就是使用加号(+)将字符串变量连接起来。如果需要重复输出字符串，可使用星号(*)，重复输出字符串格式：重复次数*字符串或者字符串*重复次数。另外，相邻的两个字符串本文会自动拼接在一起。

（4）判断字符串是否包含指定字符串

判断字符串是否包含指定字符串的方法：使用（in）操作符。

（5）判断字符串中不包含指定字符串

判断字符串中不包含指定字符串的方法：使用（not in）操作符。

3.2.2 字符串

4. 字符串的操作

（6）字符串格式化

Python格式化字符串方法，就是在格式化操作符（%）的左侧放一个格式化字符串（包含%的字符串）,在右侧放一个或多个对象，这些对象是想在左侧插入的转换目标。Python字符串格式化符号见表3-5。

【例3-3】说明Python中字符串的运用。参见教材。

3.2.3 Python数据结构组成

1. 数据结构类型

除了数字和字符串外，Python还有4个内建的数据结构，也称为容器（container），其中主要包含序列（如列表和元组）、映射（如字典）以及集合3种基本的数据结构类型。

（1）序列类型

序列是数据结构对象的有序排列，Python中有6种内建的序列，包括字符串（string）、列表（list）、元组(tuple)、Unicode字符串、buffer对象和xrange对象，其中列表和元组是最常见的类型。

3.2.3 Python数据结构组成

1. 数据结构类型

（2）映射类型

映射类型就是存储了对象与对象之间的映射关系的数据结构类型，Python中唯一的映射类型的数据类型是字典。字典中的每个元素都存在相应的名称（称为键）与之一一对应。字典相当于带有各自名称的元素组成的集合，与序列不同的是字典中的元素没有排列顺序。

（3）集合类型

Python还提供了一种称为集合的数据结构。集合当中的元素是不能重复出现，集合中的元素是相对唯一的，并且元素不存在排列顺序。集合类型包括可变集合（set）和不可变集合（frozenset）。

3.2.3 Python数据结构组成

2．可变数据类型与不可变数据类型区别

Python中可变数据类型有列表（list）和字典（dict）,不可变数据类型有整型（int）、浮点型（float）、字符串型（string）和元组（tuple）。

可变数据类型可以直接对数据结构对象进行元素的赋值修改、删除或增加操作，而不需要重新创建一个对象。

不可变数据类型是不允许变量的值发生变化，如果改变了变量的值，相当于是新建了一个对象。

3.2.4 列表

列表（list）是Python对象作为其元素并按顺序排列构成的有序集合。列表中的每个元素都有各自的位置编号，称为索引。列表中的元素不需要具有相同的数据类型。

1. 创建列表

在Python中创建列表的方法有两种，一种是使用方括号[ ]创建，另一种是使用list()函数创建。

（1） 使用方括号[ ]创建列表，只需要把所需的列表元素用逗号分隔开，并将这些列表元素用方括号([ ])括起来即可。

3.2.4 列表

```
# -*- coding: UTF-8 -*-
student_list = ['王红', '张小红', '李明', '杨丽']  # 创建学生列表
score_list = [['20170001','王红'], 70, 80, 65, 90]  # 创建成绩列表
produce_list = ['电视机', '长虹', 3600, '四川']  # 创建产品列表
emptyList = []                                  # 创建空列表

print(student_list)                             #输出完整列表
```

3.2.4 列表

1. 创建列表

（2）使用list()函数创建列表

Python中list()函数的作用实质上是将传入的数据结构换成列表类型。例如向函数传入一个元组对象，就会将对象从元组类型转变为列表类型。如果不传入任何对象到list()函数中，则会创建一个空列表，而通过list()函数对字符串创建列表非常有效。

3.2.4 列表

```
# -*- coding: UTF-8 -*-
str_list = list('Python')                    #创建字符串列表
emptylist = list()                           #创建空列表
score_list1 =list((['2017001','王红'],70, 80,65, 90)) #创建成绩列表
print(str_list)                              #输出完整列表
print(score_list1)
```

3.2.4 列表

2 . 列表索引

列表是属于序列类型的数据结构，列表中的每个元素都是有各自的位置编号（称为索引）。列表索引与字符串的索引类似，也分为正索引（也称为索引）和负索引两种。

3. 列表元素的修改

（1） 列表元素的提取

列表元素的提取就是通过索引获取列表元素中指定位置的字符。列表元素提取的方法是在列表元素后使用方括号[i]将需要提取的字符索引放置在方括号中即可。

```
# -*- coding: UTF-8 -*-
str_list = list('Python')                    #创建字符串列表
print(str_list[1])                           #输出列表中y
print(str_list[4])
```

4. 列表元素的操作

（2）列表元素切片

列表元素切片方法是序列对象后使用方括号

序列对象[i:j:k]

其中i表示截取列表元素的开始索引，j表示结束索引,k表示步长。

```
# -*- coding: UTF-8 -*-
student_list = ['王红', '张小红', '李明', '杨丽'] # 创建学生列表
print(student_list[2:5])
print(student_list[0:5:2])
```

4．列表中常用函数和方法

（1）列表元素的添加

①append()函数：使用append()函数向列表对象中添加元素的方法就是向append传入需要添加到列表对象的一个元素，则该元素会被追加到列表的尾部。

```
# -*- coding: UTF-8 -*-
student_list = ['王红','张艳','李明','杨丽']     #创建学生列表
student_list.append('王小峰')     #使用append函数向列表尾部添加元素
print(student_list)
```

3.2.4 列表

4．列表中常用函数和方法

（1）列表元素的添加

②extend()函数：使用extend()函数能够将另一个列表添加到列表的末尾，相当于将两个列表进行拼接。

```
# -*- coding: UTF-8 -*-
student_list = ['王红','张艳','李明','杨丽']     #创建列表student_list
other_list = ['王小峰','宋江','李涛']          #创建列表other_list
student_list.extend(other_list)              #使用extend函数将列表合并
print(student_list)
```

3.2.4 列表

4．列表中常用函数和方法

（1）列表元素的添加

③insert()函数：使用insert()函数也可以向列表添加一个元素，但与append()函数不同是，insert()函数可以在指定位置添加。

```
# -*- coding: UTF-8 -*-
student_list = ['王红','张艳','李明','杨丽']    #创建列表student_list
student_list.insert(1,'李涛')                    #使用insert ‘王红’ 后面插入 ‘李涛’
print(student_list)
```

3.2.4 列表

（2） 删除列表元素

① 使用del语句删除列表元素

格式：del 列表[索引号]

3.2.4 列表

（2） 删除列表元素

② 使用pop语句删除列表元素

使用pop语句删除列表元素的方法是将元素的索引传入pop语句中，将会获取该元素，并将其在列表中删除，相当于把列表中的元素抽离出来。若不指定元素位置，pop语句将指定是列表中最后一个元素。

使用pop语句格式：列表对象.pop(索引号)。

3.2.4 列表

（2） 删除列表元素

③使用remove语句删除列表元素

除了指定元素的位置进行元素的删除外，还可以指定元素进行删除。将指定的元素放入remove语句，则列表中第一次出现的该元素会被删除。

使用remove语句格式如下：列表对象.remove(元素值)。

3.2.4 列表

（3） 修改列表元素

由于列表是可变的，修改列表元素最简单的方法是提取该元素并进行赋值操作。在对列表进行添加元素、删除元素和修改元素操作时，如果需要保留原来的列表不变，只有采用copy语句创建列表的副本。

copy语句的格式：列表对象.copy()

3.2.4 列表

（4）查询列表元素位置

利用列表方法index来查询指定元素在列表中第一次出现的位置索引。判断列表中是否包含某个元素，还可以使用in函数。in函数的格式如下。

元素 in 列表对象

（5） Python中还有其他常用的操作见表3-6。

3.2.5 元组

元组与列表相似，都是有序元素的集合，并可以包含任意类型的元素。但元组与列表不同的是元组是不可变的，也就是说元组一旦创建后就不能修改。

1. 创建元组

（1）使用园括号()创建元组

元组创建很简单，只需要使用园括号将有序元素括起来，并用逗号隔开。

3.2.5 元组

```
# -*- coding: UTF-8 -*-
season = ('春','夏','秋','冬')          #创建season元组
constant = (('r',10),3.1416)          #创建constant元组
emptytuple = ()                       #创建空元组
tup1 = 12,                            #创建一个元素元组
tup2=(12)
print(season,constant,emptytuple,tup1,tup2)
```

3.2.5 元组

1. 创建元组

（2）使用tuple()函数创建元组

tuple()函数能够将其他数据结构对象转换成元组类型。

具体示例见教材。

```
# -*- coding: UTF-8 -*-
constant = tuple((('r',10),3.1416))    #创建constant元组
tup1 = tuple((12,))                    #创建一个元素元组
print(constant,tup1)                   #输出元组
```

3.2.5 元组

2. 元组索引

元组与列表类似，也分为正索引（也称为索引）和负索引两种。

3. 元组元素的提取和切片操作

（1）通过索引访问提取元素

与列表索引访问提取元素一样，只要传入元素索引，就可以获得对应的元素。

（2）元组切片操作

元组切片操作与列表类似，只需要传入所提取子序列起始元素的索引，终止元素的索引和提取的步长值即可。

具体示例见教材。

3.2.5 元组

2. 元组索引

元组与列表类似，也分为正索引（也称为索引）和负索引两种。

3. 元组元素的提取和切片操作

（1）通过索引访问提取元素

与列表索引访问提取元素一样，只要传入元素索引，就可以获得对应的元素。

3.2.5 元组

（2）元组切片操作

元组切片操作与列表类似，只需要传入所提取子序列起始元素的索引，终止元素的索引和提取的步长值即可。

```
# -*- coding: UTF-8 -*-
season = ('春','夏','秋','冬')          #创建元组
tup1 = season [1:3]                    #提取第2至3元素开始所有元素
tup2 = season [0:5:2]                  #从第1个元素开始，步长为2提取元素
print(tup1,tup2)
```

3.2.5 元组

4. 元组解包

将元组中的各个元素赋值给多个不同变量的操作通常称为元组解包，元组解包的格式如下。

var1,var2,...varn=tuple

```
# -*- coding: UTF-8 -*-
season = ('春','夏','秋','冬')        #创建元组
a,b,c,d = season                      #元组解包
print('a=',a,'b=',b,'c=',c,'d=',d)    #输出a,b,c,d
a,b,c,d = '春','夏','秋','冬'          #多重赋值语句，右边也可看成元组
print('a=',a,'b=',b,'c=',c,'d=',d)    #输出a,b,c,d
constant = tuple((('r',10),3.1416))   #创建元组
x,y = constant                        #元组解包
print('x=',x,'y=',y)                  #输出x,y
(x,y),z = constant                    #元组解包
print('x=',x,'y=',y,'z=',z)           #输出x,y,z
```

3.2.5 元组

5. 元组中常用方法和函数

元组中常用函数和方法见表3-7。

【例3-4】说明Python中列表与元组的运用

```
# -*- coding: UTF-8 -*-
title = ('姓名','电话','Email')          #创建元组
cont = ['李芳',13862345616,'lifang@163.com']  #创建列表
str1,str2,str3 = title
con1,con2,com3 = cont
str4 = "--------"
str5 = str4*4
print('%s' % (str5))
print(' │ %s|   %s   |    %s   |' % (str1,str2,str3))
print('%s' % (str5))
print(' │ %s|%d|%s|' % (con1,con2,com3))
print('%s' % (str5))
cont[0] = '李志'
cont[1] = 18573323312
cont[2] = 'lizhi@sina.com'
con1,con2,com3 = cont
print(' │ %s|%d|%s|' % (con1,con2,com3))
```

3.2.6 字典

字典主要是通过键来访问对应的元素，为了获取指定键所关联的值，将键放到一对方括号中，如“值=字典对象[键]”。

1. 创建字典

（1）使用花括号｛｝创建字典

字典是用花括号({})括起来的一系列键值对，其中每个键值(key=>value)对的键与值之间用冒号(:)分隔，各键值对之间用逗号(,)隔开，字典的格式如下。

{key1:value1,key2:value2…,keyn:valuen}

3.2.6 字典

```
# -*- coding: UTF-8 -*-
contacts = {'张艳':13873321234,'杨丽':13623456712}    #创建contacts
字典
student = {'Name':'张艳','Age':20,3:13873321234}    #创建student字典
emptydict = {}                                    #创建空字典
dict1 = {1:'空调',2:'1.5p',2:1989}                    #创建dict1字典
print(contacts)                                   #输出字典
print(student)                                    #输出字典
print(emptydict,dict1)                            #输出字典
```

3.2.6 字典

（2）使用dict()函数创建字典

```
# -*- coding: UTF-8 -*-
 dict((['key1','value1'],['key21','value2'],'ab',...))
or
  dict([['key1','value1'],['key21','value2'],'ab',...[])
```

#key 只能是数字，字符，元组

#value 可以是数字，字符，列表，元组和字典

#value 代表任意2个字符

3.2.6 字典

（2）使用dict()函数创建字典

```
# -*- coding: UTF-8 -*-
student = dict((['Name','张艳'],('Age',20),'xy',('d',{1:2})))
contacts = dict([(('tel','email'),[13873321234,'zhangyan@163.com'])])
dict1 = dict(Name='杨丽',Age=18,a=(1,2))
dict2 = dict()
print(student)
print(contacts)
print(dict1,dict2)
```

3.2.6 字典

2. 获取字典的元素

（1）通过键获取字典元素

（2）使用字典get()函数

具体示例见教材。

3. 字典常用函数和方法

（1）添加字典元素

通过向字典中传入一个新的键，并对这个新键进行赋值操作，可实现向字典中添加一个元素的功能，其格式如下。

```
dict_name[new_key]=new_value
```

字典中的update方法可实现将两个字典中的键值对进行合并。

3.2.6 字典

2. 获取字典的元素

（1）通过键获取字典元素

（2）使用字典get()函数

```
# -*- coding: UTF-8 -*-
student = dict((['Name','张艳'],('Age',20)))
if 'Name' in student:
    print(student['Name'])          #输出键为'Name'的对应值
if 'Age' in student:
    print(student['Age'])           #输出键为'Age'的对应值
```

3.2.6 字典

2. 获取字典的元素

（1）通过键获取字典元素

（2）使用字典get()函数

```
# -*- coding: UTF-8 -*-
dict1 = dict(Name='杨丽',Age=18,a=(1,2))
print(dict1.get('Name'))
print(dict1.get('age','字典中没有该键'))    #注意键Age的大小写
```

3.2.6 字典

3. 字典常用函数和方法

（1）添加字典元素

通过向字典中传入一个新的键，并对这个新键进行赋值操作，可实现向字典中添加一个元素的功能，其格式如下。

```
dict_name[new_key]=new_value
```

字典中的update方法可实现将两个字典中的键值对进行合并。

3.2.6 字典

3. 字典常用函数和方法

（1）添加字典元素

通过向字典中传入一个新的键，并对这个新键进行赋值操作，可实现向字典中添加一个元素的功能，其格式如下。

```
dict_name[new_key]=new_value
```

字典中的update方法可实现将两个字典中的键值对进行合并。

3.2.6 字典

3. 字典常用函数和方法

（1）添加字典元素

```
# -*- coding: UTF-8 -*-
student = dict((['Name','张艳'],('Age',20)))
stu_copy = student.copy()                    #创建1个字典副本
stu_copy['tel'] = 13873321234                #给字典副本添加键为tel的值
dict1 = dict(Name='杨丽',ClassName='软件一班')
stu_copy.update(dict1)                       #合并字典,相同键Name的值会被
替换
print(stu_copy)
```

3.2.6 字典

（2）删除字典元素

使用del语句可以删除某个键值对，另外，字典也包含pop语句，只要传入键，函数就能将对应的值从字典中抽离，若要清空字典内容，可以使用字典方法的clear()函数。

3.2.6 字典

```
# -*- coding: UTF-8 -*-
student = dict((['Name','张艳'],('Age',20),'xy',('d',{1:2})))
del student['x']                    #删除键为x的值
print(student)
student.pop('d')                    #删除键为d的值
print(student)
stu_copy = student.copy()
print(stu_copy.clear())             #清空字典
```

3.2.6 字典

（3）修改字典元素，可使用键访问赋值来修改。

（4）查询和获取字典元素信息

除了可以使用字典元素提取的方法进行查询外，还可以使用in()函数进行判断。在字典方法中，可以使用keys获取字典中的所有键，values获取字典中的所有值，itmes得到字典中的所有键值对。

3.2.6 字典

```
# -*- coding: UTF-8 -*-
student = dict((['Name','张艳'],('Age',20)))
all_keys = student.keys()          #获取所有键
print(all_keys)
all_values = student.values()      #获取所有值
print(all_values)
all_itmes = student.items()        #获取所有键值对
print(all_itmes)
```

3.2.6 字典

```
keys = list(student.keys())          #获取所有键
print(keys)
values = list(student.values())      #获取所有值
print(values)
key_value= keys[values.index(20)]    #查找值为20所对应的健
print(key_value)
if 'Name' in student:
    print(student['Name'])           #输出键Age所对应的值
```

例3-5

```
# -*- coding: UTF-8 -*-
con_dict = {'姓名':'李芳','电话':13862345616,  'Email':'lifang@163.com'}        #创
字典
all_keys = con_dict.keys()                  #获取字典中所有键
str1,str2,str3 = all_keys                  #将键赋值变量
str4 = "--------"
str5 = str4*4
print('%s' % (str5))
```

```
print('%s' % (str5))
print('｜%s|%s|%s|' % (con_dict['姓名'],
                      con_dict['电话'],con_dict['Email']))
print('%s' % (str5))
con_dict['姓名'] = '李志'                    #修改字典中键为'姓名'的值
con_dict['电话'] = 18573323312
con_dict['Email'] = 'lizhi@sina.com'
print('｜%s|%s|%s|' % (con_dict['姓名'],
                      con_dict['电话'],con_dict['Email']))
```

3.2.7 集合

集合(set)就是一个由唯一元素组成的非排序的集合体。

1. 创建集合

创建可变集合方法是使用花括号{ }或set()函数。其格式如下。

value0, value1, value2…}　#花括号中元素是数字、字符串、元组

set(value)　　#value是列表或元组

3.2.7 集合

```
# -*- coding: UTF-8 -*-
student = {'王明',20,('李志',18)}        #用{}创建可变集合
student1 = set([1,2,5,1,(1,4),2])        #用set创建可变集合,重复值会
删除
str = set('asd')                          #用set创建可变集合
set()                                     #创建空集合
print(student)
print(student1,str)
student2 = frozenset(('王明',20,'张宁',19))    #创建不可变集合
```

3.2.7 集合

2. 集合的运算

见表3-8集合运算方法和表3-9集合运算符,。

3. 集合常用函数和方法

见表3-10集合常用函数和方法

【例3-6】说明Python中集合的运用。

```
# -*- coding: UTF-8 -*-
s1 = set('hello')          #创建由字符串字母组成集合
s2 = set(('hello',))       #创建一个英文单词集合
print(s1,s2)
s1 = set(['hello','world'])
s2 = set(('hello','Python'))
print(s1,s2)
s1.add('!')                #给s1集合添加字符
s1.update(s2)              #给s1集合添加s2集合
print(s1,s2)
```

3.2.7 集合

```
#s1.s2集合运算
print(s1-s2)
print(s1&s2)
print(s1|s2)
print(s1>=s2)
print(s1<=s2)
print(s1^s2)
```

3.3.1 if 条件语句

if 条件语句的基本语法格式如下。

if 条件表达式:

语句块

if语句执行过程，首先判断条件表达式（也就是布尔表达式）是否成立，即条件表达式是否为真（True），如果为真（True），则执行冒号（:）下面的语句块，否则，就不执行该语句块。if语句控制结构图如图3-4所示。

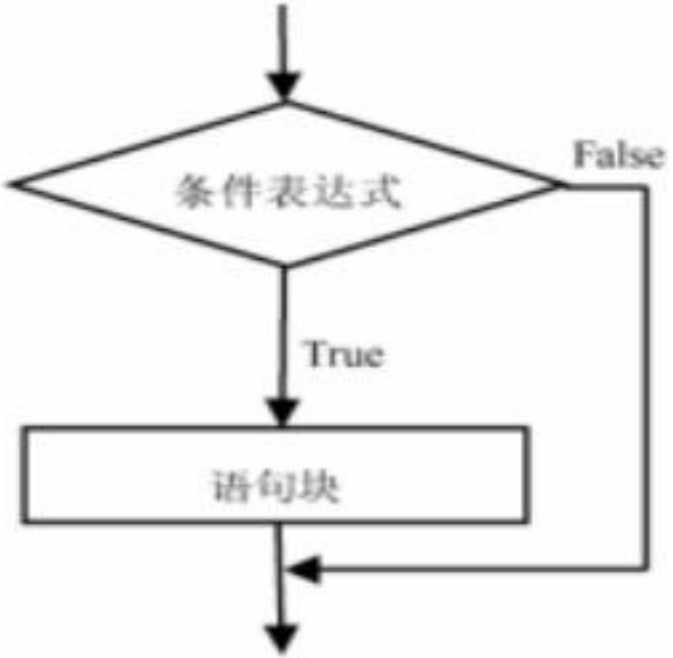


图 3-4 if 语句的控制结构

3.3.1 if 条件语句

除了单路分支语句外，还会经常遇到多路分支语句。多路分支语句的语法格式如下。

```
if 条件表达式1:
    分支一语句块
elif 条件表达式2:
    分支二语句块
else:
    分支三语句块
```

3.3.1 if 条件语句

```
# -*- coding: UTF-8 -*-
#input从标准输入中得到的值是一个字符串
grade = input("\n请输入学生成绩:")
if int(grade)>=85 :  #用int(grade)强制转换为整型
    print("优秀学生！")
elif int(grade)<60 :
    print("不合格学生！")
else :
    print("合格学生！")
```

3.3.2 while 循环控制语句

Python中的循环控制语句有while 语句和for语句，其中，while语句是常用的循环控制语句之一，它的语法格式如下。

```
while (条件表达式):
    语句块
```

while语句的执行过程，首先判断条件表达式是否为真，如果结果为真，则执行条件表达式冒号（:）后的语句块，执行完毕后再次判断条件表达式是否为真，如果仍然为真，则再次执行冒号（:）后的语句块，如此反复，直到条件表达式为假。循环控制语句的控制结构如图3-5所示。

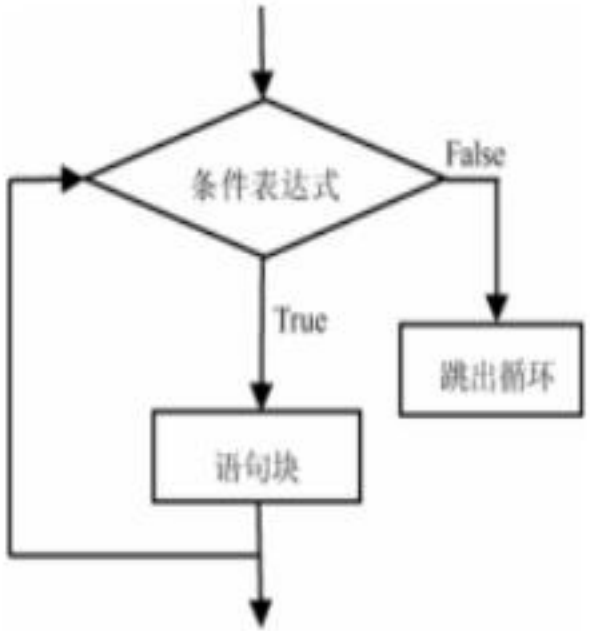


图 3-5 while 循环控制语句的控制结构

3.3.2 while 循环控制语句

```
# -*- coding: UTF-8 -*-
str = input("\n请输入数字n:")
s,i = 0,1
while (i<=int(str)) :
    s = s + i
    i += 1
print('1至 %d 之和为: %d' % (int(str),s))
```

3.3.2 while 循环控制语句

```
# -*- coding: UTF-8 -*-
var = 1
while (var==1) :
  str = input("\n请输入数字n:")
  s,i = 0,1
  while (i<=int(str)) :
     s = s + i
     i += 1
  print('1至 %d 之和为: %d' % (int(str),s))
```

3.3.2 while 循环控制语句

```
# -*- coding: UTF-8 -*-
s,i = 0,1
while (i<=100) :
    s = s + i
    i += 1
else:
   print('1至 %d 之和为: %d' % (i-1,s))
```

3.3.3 for 循环控制语句

在Python中，for循环是一个通用的序列迭代器，可以遍历任何有序列的序列，如字符串、列表、元组等。

Python中的for语句接收可迭代对象，如序列和迭代器作为其参数，每次循环可以调取其中的一个元素。Python的for循环看上去像伪代码，非常简洁。

for 循环控制语句的格式如下。

```
for <变量> in <序列>:
    语句块
```

。

3.3.3 for 循环控制语句

```
# -*- coding: UTF-8 -*-
str = 'Python'
for i in str:
    print(i)

for i in ('c','java','Python','php'):
    print("循环数据 " + i)
```

3.3.3 for 循环控制语句

```
# -*- coding: UTF-8 -*-
for i in ('c', 'java', 'Python', 'php'):
    print("循环数据 " + i)
else:
    print('循环完成！ ')
```

3.3.4 range()函数作用

range()函数是Python的内置函数，它可以返回一系列连续添加的整数，生成一个列表对象。range()函数常用在for循环中可作为索引使用。

在Python 3.0中range()函数是一个迭代器。range()函数语法格式如下。

range(start,end[,step])

1、函数中的参数说明

2、在Python中，range()函数的作用

3.3.4 range()函数作用

```
str = ['C','Java','Python','PHP']
for i in range(len(str)):          #遍历序列的索引
    print(i, str[i])

a = list(range(8))                 #创建列表
print(a)
```

3.3.5 break、continue、pass语句

1. break语句

在while和for循环中，break语句的作用是终止循环语句，即循环条件没有False条件或者序列还没被完全递归完，就停止执行循环语句，跳出循环。在嵌套循环中，break语句可以停止执行最深层的循环，并开始执行下一行代码。

在Python中，break语句的语法为break。

3.3.5 break、continue、pass语句

```
# -*- coding: UTF-8 -*-
s,i = 0,1
while (i) :
    s = s + i
    i += 2
    if (i > 100):
        break
print(s)
```

3.3.5 break、continue、pass语句

2. continue语句

在while和for循环中，continue语句作用是告诉Python跳过当前循环的剩余语句，然后继续进行下一轮循环。continue语句与break语句不同，continue语句是跳出本次循环，而break语句是跳出整个循环。

continue语句的语法格式为continue。

3.3.5 break、continue、pass语句

```
# -*- coding: UTF-8 -*-
s = i = 0
while (i<100) :
    i += 1
    if (i%2==0):    #如果n是偶数，执行continue语句
        continue    # continue语句会直接继续下一轮循环，后续的语句不会执行
    s = s + i
    print(i,s)
print(s)
```

3.3.5 break、continue、pass语句

3. pass语句

pass是空语句，作用是保持程序结构的完整性。pass语句不做任何事情，一般用做占位语句。pass语句的语法格式为pass。

3.3.5 break、continue、pass语句

```
# -*- coding: UTF-8 -*-
# 输出 Python 的每个字母
for letter in 'Python':
   if letter == 'h':
      pass
      print('这是 pass 块')
   print('当前字母 :', letter)
```

3.4.1 自定义函数

在Python中定义函数方法是以def关键词开头，其后紧接函数标识符名称和圆括号()。园括号内包含将要在函数体中使用的形式参数（简称形参），各参数之间用逗号分割，定义语句以冒号（:）结束。函数体编写另起一行，函数体的缩进为4个空格或者一个制表符，函数定义的语法格式如下。

```
def 函数名(参数列表):
    函数体
```

3.4.1 自定义函数

```
# -*- coding: UTF-8 -*-
#定义函数
def sum_function(n):
    #求1到n数字的累加和
    sum,i=0,1
    while (i<=n) :
      sum = sum + i
      i += 1
    print('1至 %d 之和为: %d' % (n,sum))
    return sum
```

3.4.2 设置函数参数

参数对于函数而言，只是作为函数的输入，目的是调用函数时，可以传递不同的值给函数，然后得到相应的结果。

在定义函数时，函数名后面，园括号中的变量名称为形式参数，或者称为"形参"；在调用函数时，函数名后面，园括号中的变量名称叫做实际参数，或者称为"实参"。

Python中的函数参数主要有位置参数、关键字参数、默认参数和可变参数。

3.4.2 设置函数参数

默认参数

```
# -*- coding: UTF-8 -*-
#定义函数
def score_fun(normal,exam,n_Perc=0.4,e_Perc=0.6):
    #求期评成绩
    score = normal*n_Perc+exam*e_Perc
    return score
#调用
score = score_fun(90,80)
print (score)
print (score_fun(90,80))
```

3.4.2 设置函数参数

位置可变参数：*args，它们放在参数的最后面

传递参数时可以在原有参数后添加0到多个位置可变参数。

```
# -*- coding: UTF-8 -*-
#定义函数
def example_fun(x,*args):
    print('x=',x)
    print('args=',args)

#调用函数
example_fun(67,78,'asd','xyz')
```

3.4.2 设置函数参数

任意数量的关键字可变参数**kargs。。参数名称前必须有两个星号**，其位置必须放在所有参数最后面

```
# -*- coding: UTF-8 -*-
#定义函数
def example_fun(x,*args,**kwargs):
    print('x=',x)
    print('args=',args)
    print('kwargs=',kwargs)

#调用函数
afun=example_fun(67,78,'asd','xyz',a=1,b=2)
print (afun)
```

3.4.3 返回函数值

函数可以处理一些数据，并返回一个或一组值。函数返回的值称为返回值。如果想要保存或者调用函数的返回值，需要用到return语句。

Python对函数返回值的数据类型没有限制，包括列表或字典等复杂的数据结构。当程序执行到函数中的return语句时，就会将指定的值返回并结束函数，后面语句不会被执行。

3.4.4 调用自定义函数

Python中使用“函数名()”的格式对函数进行调用，根据参数传入方式不同，总共有3种函数调用方式，分别为

位置参数调用： score_fun(90,80)， 参数顺序不能换

关键字参数调用，关键字参数必须在位置参数后面：

score_fun(exam=90,norm=80) ;score_fun(90,exam=80)

可变参数调用：

3.4.4 调用自定义函数

可变参数调用：

用*args 可变参数列表可以将元组或列表转换为参数，然后传入参数；

用**kwargs 关键字参数列表可以将字典转换为关键字参数，然后传入参数。

3.4.4 调用自定义函数

```
# -*- coding: UTF-8 -*-
args=[1,6,2]
a=list(range(*args))
print(a)
def contacts(name,**kwargs):
    print('name:',name,'tel:',kwargs)

contacts('李芳
',tel='13834561234',email='lifang@163.com')
```

3.4.5 局部变量和全局变量

1. 局部变量

在定义函数时，往往需要在函数内部对变量进行定义和赋值，在函数体内定义的变量为局部变量，局部变量只能在其被赋值的函数内部访问。

3.4.5 局部变量和全局变量

```
# -*- coding: UTF-8 -*-
def total(x, y):
    #求两个数之和
    sum = x + y   # sum是局部变量
    print('函数内是局部变量 : ', sum)
    return sum

#调用total函数
total(10,20)
print(sum)     #输出<built-in function sum>，函数体外不能访问局部变量
```

3.4.5 局部变量和全局变量

2. 全局变量

定义在函数体外的变量称为全局变量。全局变量可以在函数体内被调用。

具体示例见教材。

```
# -*- coding: UTF-8 -*-
total = 10
def fun():
    sum = total + 100   #调用了全局变量total
    return sum

#调用fun函数
a = fun()          #a=110，在函数体内调用了全局变量total
print(a)
```

```
# -*- coding: UTF-8 -*-
total = 10
def fun():
    total = 0          #定义total局部变量
    total = total + 100   #全局变量total被局部变量覆盖
    return total

#调用fun函数
a = fun()          #a=100
print(a)
```

```
# -*- coding: UTF-8 -*-
total = 10
def fun():
    global total
    total = total + 100   #调用了全局变量total
    return total
```

3.4.6 函数嵌套

Python允许在函数中定义另外一个函数，这就是函数的嵌套。定义在其他函数内部的函数称为内建函数，而包含有内建函数的函数称为外部函数。其中，内建函数中的局部变量独立于外部函数，如果想使用外部函数的变量，则需要声明该变量为全局变量。

具体示例见教材

3.4.6 函数嵌套

```
# -*- coding: UTF-8 -*-
def mean(args):
    def sum(x):
        total = 0
        for i in x:
            #print(i)
            total += i
        return total
    return sum(args)/len(args)

#调用mean函数
args=list(range(6))
a = mean(args)
print(a)
```