第 1 章

认识 Python

Chapter 1

【学习目标】

(1) 了解 Python 特点和用途。

(2) 掌握 Python 环境的搭建。

(3) 了解综合案例内容及要求。

任务 1.1　了解 Python

1.1.1　Python 语言的特点

Python 语言诞生于 1989 年,是由吉多·范罗苏姆开发的一种面向对象、解释型、弱类型的脚本语言,也是一种功能强大而完善的通用型语言。

它的主要特点如下。

(1) 通用性。Python 语言几乎可以用于任何与程序设计相关应用的开发。

(2) 语法简洁、易学。Python 语言主要用来精确表达问题逻辑,更接近自然语言,只有 33 个保留字,十分简洁。

(3) 面向对象。Python 语言既支持面向过程,也支持面向对象,提供了类、对象、继承、重载、多态等编程机制。

(4) 丰富的扩展库。Python 提供了丰富的标准库,可以满足各种编程场景的应用,如数据分析与挖掘、图像处理、网络爬虫等。

1.1.2　了解 Python 语言的应用领域

Python 具有丰富和强大的库。它常被戏称为“胶水语言”,因为使用 Python 能够把用其他语言制作的各种模块(尤其是 C/C++)很轻松地联结在一起。常见的一种应用情形是使用 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,用更合适的语言加以改写,比如 3D 游戏中的图形渲染模块,性能要求特别高,就可以先用 C/C++重写,而后封装为 Python 可以调用的扩展类库。Python 的应用领域非常广泛,其主要应用包括但不限于以下领域。

(1) 系统编程。提供 API(Application Programming Interface,应用程序编程接口),能方便进行系统维护和管理,Linux 下标志性语言之一,是很多系统管理员理想的编程工具。

(2) 图形处理。提供 PIL、Tkinter 等图形库支持,能方便地进行图形处理。

(3) 数学处理。NumPy 扩展提供大量标准数学库的接口。

(4) 文本处理。Python 提供的 re 模块能支持正则表达式,同时还提供 SGML、XML 分析模块,许多程序员利用 Python 可方便地进行 XML 程序的开发。

(5) 数据库编程。程序员可通过遵循 Python DB-API(数据库应用程序编程接口)标准的模块与 SQL Server、Oracle、Sybase、DB2、MySQL、SQLite 等数据库进行通信。Python 自带 Gadfly 模块,提供了一个完整的 SQL 环境。

(6) 网络编程。提供丰富的模块支持 Sockets 编程,能方便快速地开发分布式应用程序。很多大规模软件开发计划,如 Zope、Mnet、BitTorrent 及 Google 都在广泛地使用 Python。

(7) Web 编程。应用的开发语言,支持最新的 XML 技术。

(8) 多媒体应用。Python 的 PyOpenGL 模块封装了 OpenGL 应用程序编程接口,利用该模块能进行二维和三维图像处理。另外,PyGame 模块可用于编写游戏软件。

(9) 数据分析与处理。Python 拥有一系列比较完善的数据分析与处理的标准库,其中 Matplotlib 经常会被用来绘制数据图表,它是一个 2D 绘图工具,可以完成直方图、散点图、折线图、条形图等的绘制。Pandas 是基于 Python 的一个数据分析工具,该工具是为了解决数据分析任务而创建的,拥有大量类库和一些标准的数据模型,提供了可高效操作大型数据集所需的工具。

任务 1.2 Python 语言开发环境搭建与使用

Spyder 是一个简单的 Python 开发集成环境,Python 的 Anaconda 版本中则集成了对 Spyder 的支持。Anaconda 是一个开源的 Python 发行版本,是一款集成的 Python 环境,安装 Anaconda 后就默认安装了 Python、IPython、集成开发环境 Spyder 和众多的包和模块。如在安装过程中选择一键安装,软件包则包含了 Python 等 180 多个科学包及依赖项,其最大的特点就是可以便捷获取包,且能方便地对包及其版本进行管理。

(1) 下载 Anaconda。Anaconda 可以从官网 https://www.anaconda.com/distribution/下载,也可以从清华大学镜像网站下载,对于速度而言,国内镜像下载比较快。清华大学的镜像网址为 https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/,选择 Anaconda3-5.0.0-Windows-x86_64.exe,如图 1-1 所示。

Anaconda3-5.0.0-Linux-ppc64le.sh	296.3 MiB	2017-09-27 05:31
Anaconda3-5.0.0-Linux-x86.sh	429.3 MiB	2017-09-27 05:43
Anaconda3-5.0.0-Linux-x86_64.sh	523.4 MiB	2017-09-27 05:43
Anaconda3-5.0.0-MacOSX-x86_64.pkg	567.2 MiB	2017-09-27 05:31
Anaconda3-5.0.0-MacOSX-x86_64.sh	489.9 MiB	2017-09-27 05:34
Anaconda3-5.0.0-Windows-x86.exe	415.8 MiB	2017-09-27 05:34
Anaconda3-5.0.0-Windows-x86_64.exe	510.0 MiB	2017-09-27 06:17
Anaconda3-5.0.0.1-Linux-x86.sh	429.8 MiB	2017-10-03 00:33
Anaconda3-5.0.0.1-Linux-x86_64.sh	524.0 MiB	2017-10-03 00:34
Anaconda3-5.0.1-Linux-x86.sh	431.0 MiB	2017-10-26 00:41
Anaconda3-5.0.1-Linux-x86_64.sh	525.3 MiB	2017-10-26 00:42
Anaconda3-5.0.1-MacOSX-x86_64.pkg	568.9 MiB	2017-10-26 00:42
Anaconda3-5.0.1-MacOSX-x86_64.sh	491.0 MiB	2017-10-26 00:42
Anaconda3-5.0.1-Windows-x86.exe	420.4 MiB	2017-10-26 00:44
Anaconda3-5.0.1-Windows-x86_64.exe	514.8 MiB	2017-10-26 00:45
Anaconda3-5.1.0-Linux-ppc64le.sh	285.7 MiB	2018-02-15 23:22

图 1-1 清华大学镜像网址文件选择页面

(2) 安装 Anaconda。下载完成后,双击安装文件,弹出的界面如图 1-2 所示。

单击 Next 按钮,打开如图 1-3 所示的对话框,在 Advanced Options 栏中不要选中 Add

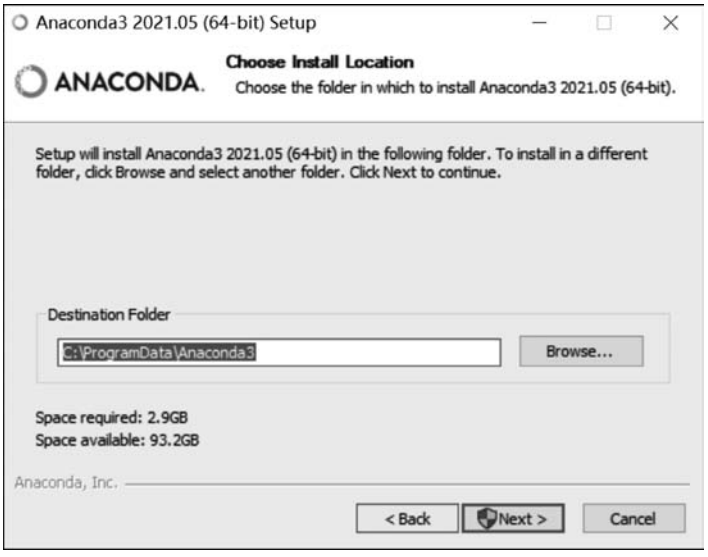

图 1-2 Anaconda3 安装界面

Anaconda3 to the system PATH environment variable(添加 Anaconda 至系统环境变量)复选框。因为如果选中该复选框,则将会影响其他程序的使用。如果使用 Anaconda,则通过打开 Anaconda Navigator 或者开始菜单中的 Anaconda Prompt(类似 Mac OS 中的“终端”)即可。

为了让其他相关程序,如一些 Python 开发环境,能够自动检测 Anaconda,请选中 Register Anaconda3 as the system Python 3.8 复选框。如图 1-3 所示。

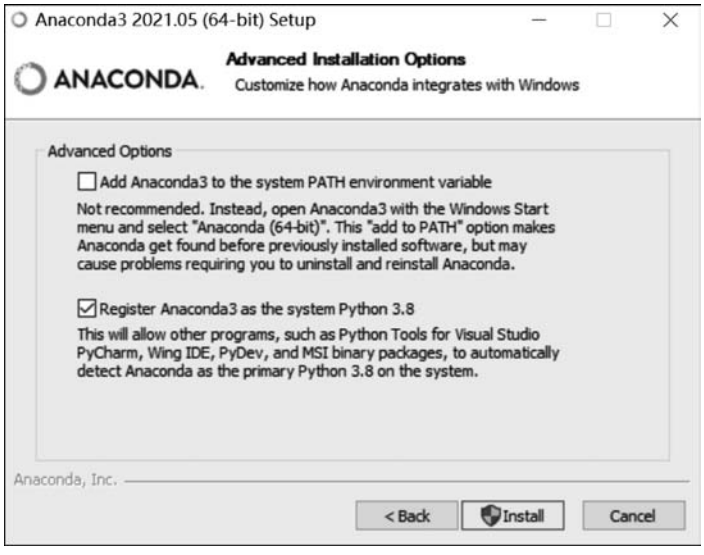

图 1-3 Advanced Installation Options 窗口

然后单击 Install 按钮开始安装。如果想要查看安装细节,则可以单击 Show Details 按钮。安装完成后则可以启动 Spyder。选择“开始”|Anaconda3|Spyder,便可以打开 Spyder 环境开发窗口,如图 1-4 所示。

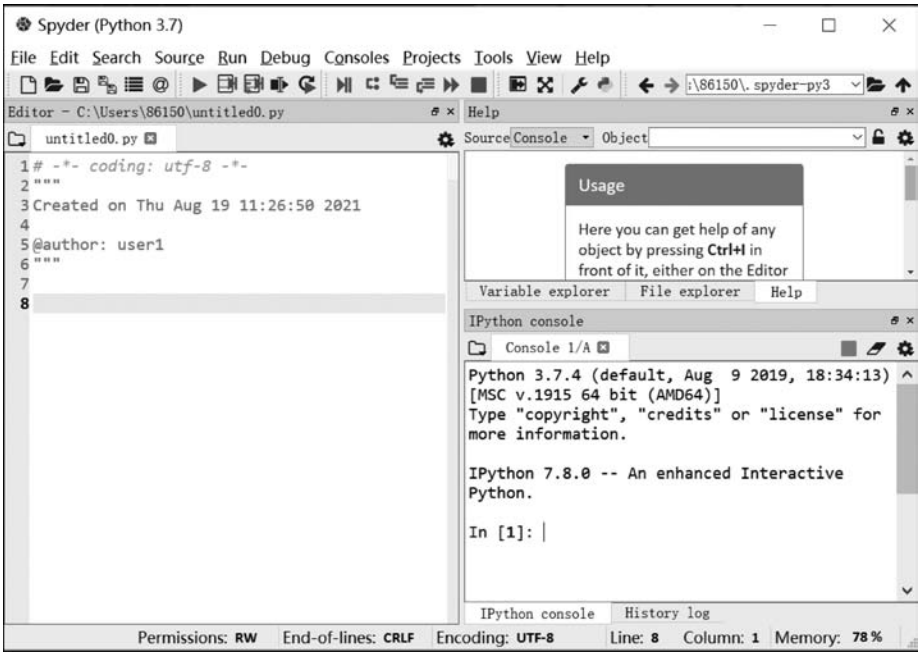

图 1-4 Spyder 开发窗口

在右下角的 Console 区输入 Python 命令后按 Enter 键，即可获得交互式结果。如图 1-5 所示，在 Console 区输入 print(3+4)命令后按 Enter 键，即可获得输出结果 7。

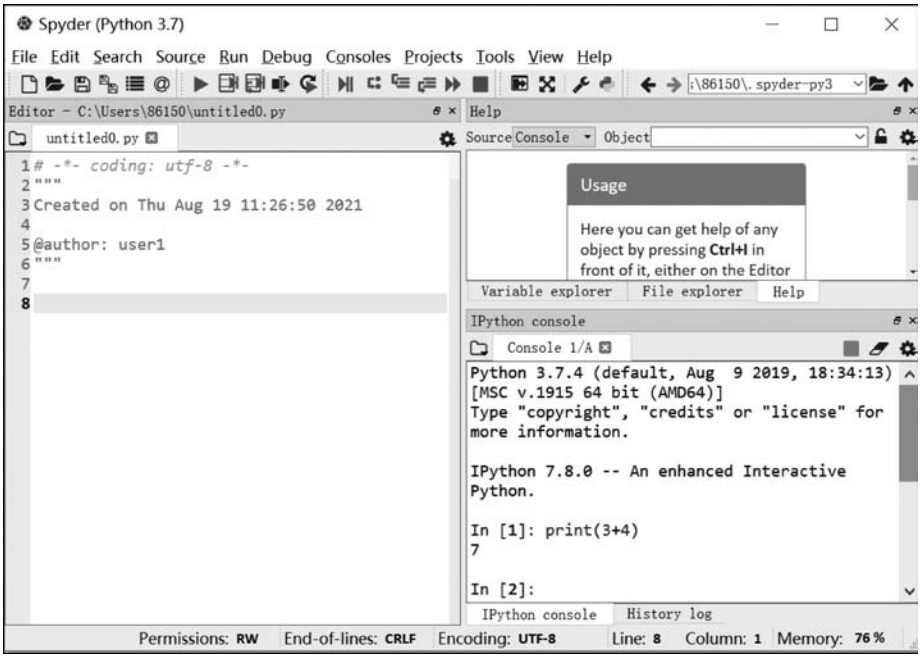

图 1-5 Console 交互命令区

位于窗口左侧编辑区的文件 untittled0.py 即为当前可编辑的 Python 程序文件，可以在其中写入 Python 程序后，单击菜单 Run|Run，会要求先保存文件，输入自定义文件名 lesson1

并进行保存后，即可在 Console 区看到运行结果，如图 1-6 所示。

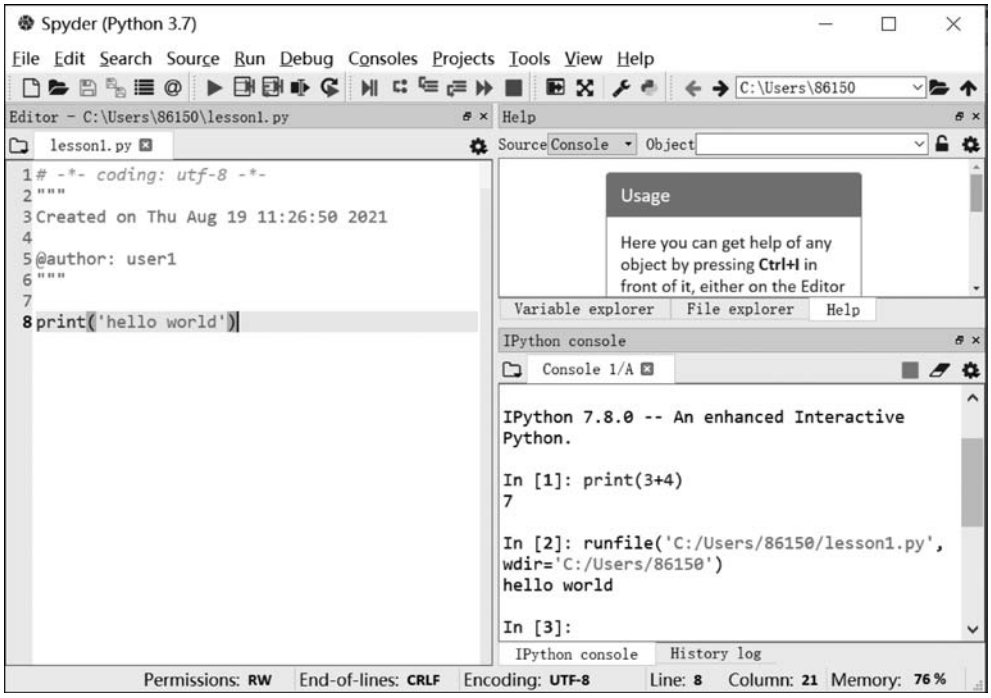

图 1-6 Python 程序编辑及运行结果

任务 1.3 Python 程序运行方式

Python 程序有两种运行方式，即交互式和文件式。

交互式是指利用 Python 解释器即时响应用户输入的代码并输出结果，在学习使用单个语句命令时，这种方式尤为方便易用。如 1.2 节中的图 1-5 展示了交互式的运行方式。

文件式是指先将代码编写成 Python 程序(扩展名为.py)，然后启动解释器将源代码转换为字节码(扩展名为.pyc)，之后将字节码转发到 Python 虚拟机中解释执行。如 1.2 节中的图 1-6 展示了 Python 程序的文件式运行方式。

任务 1.4 本书综合案例简介


任务 1.4

Python 支持很多开源扩展库，Turtle 是其中一个，它可以控制小海龟在屏幕上移动进行绘图。使用 Turtle 库中提供的函数可以实现在屏幕上绘制静态图形、动画的制作及程序与用户的实时交互。

为了综合运用 Python 知识，本书中提供了一个基于 Turtle 库的图形动画程序作为贯穿案例，其效果如图 1-7 所示。图中有四个中国结、一个长木板、一段文字及一个作为场景图形的小树林。其中四个中国结画法一致，但初始时大小位置不同，在运行时，四个中国结会按一定速度上下运动，到达上下边界后反向运动。通过按键 A、D 可以控制木板左右移动，在与中国结相遇时，中国结会停留在上面。当用户使用按键移动木板离开中国结范围时，中国结会继

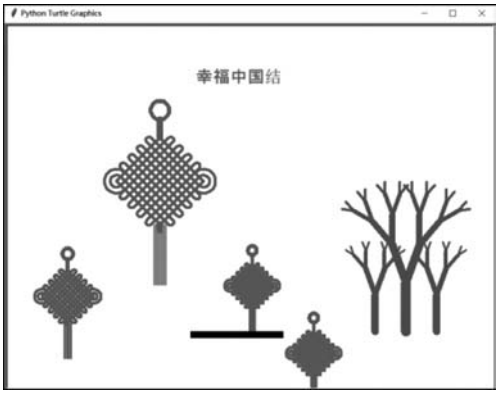

图 1-7　图形动画综合案例示意图

续下落。木板成为一个控制中国结是否下落的变因，从而完成用户对动画的控制，类似于游戏中玩家对游戏的控制。此程序涉及图形的绘制、动画的基本元素、动画交互的基本元素等相关知识，简单易学。整个案例的实现被分解成各个知识模块并贯穿于各个章节中，有助于基础知识的学习和运用，并可在此基础上扩展，从而能够制作复杂的动画和游戏程序。

在第 3 章中会使用模块化程序设计方法将单个中国结的绘制分解成主要函数，并调用 Turtle 库绘图函数实现各模块基本组成部分。在第 4 章会对各函数使用选择结构、循环结构进行扩充、完善，从而实现完整单个中国结的绘制及运动。在第 5 章会应用元组、列表、字典等数据结构扩展程序，用来实现多个中国结的生成及运动控制。在第 6 章会应用字符串、文件等实现中国结参数的动态输入获取，从而增加程序的灵活性。在完成案例开发的过程中既可学习 Python 的基本知识，又可以依据所学知识设计自己的动画和游戏程序，从而培养综合程序设计开发的能力。

小　　结

本章介绍了 Python 语言的特点和用途，并以 Anaconda 环境为例介绍了 Python 开发环境的搭建过程及使用方法，同时介绍了一个贯穿案例的运行效果及主要绘制过程。通过本章的学习，相信你已经掌握了 Python 语言工作环境的搭建方法，并了解了学习的主要目标，可以展开有趣的学习旅程了。

习　　题

一、填空题

1. Python 是由_____、_____、_____三个主要部分组成。
2. 编写 Python 语言程序，其扩展名为_____，编译后生成的文件扩展名为_____。

二、简答题

1. Python 程序的运行原理是什么？
2. Python 程序运行方式有哪些？
3. Python 有哪些特点？

第 2 章

Python 基础知识

Chapter 2

【学习目标】

(1) 掌握 Python 标识符的命名及注释规则。

(2) 掌握 Python 变量的定义及使用方法。

(3) 掌握 Python 的输入输出控制。

(4) 掌握 Python 的基本数据类型。

(5) 掌握 Python 的算术运算符及表达式的应用。

任务 2.1　学习 Python 基本语法

【任务描述】

使用 Python 语言编写程序实现中国结案例，需要按照 Python 语言的语法规则，创建变量、注释及进行基本运算等。

【任务分析】

(1) 掌握 Python 的注释规则。

(2) 掌握 Python 标识符的命名规则。

微课 2-1

2.1.1　注释

注释是对程序代码的解释和说明。在编写程序时，为了提高程序的可读性，让人一看就知道这段代码的作用，需要给某些重要的代码添加注释。Python 解释器会忽略注释，而且注释内容也不会影响程序的执行结果。Python 中的注释有两种，即单行注释和多行注释。

1. 单行注释

Python 中的单行注释以 # 开头。

【代码 2-1】 单行注释。使用 # 对一行代码进行单行注释。

```
1    # 输出中国结
2    print("中国结寓意真善美!")
```

代码说明： Python 解释器执行语句 print("中国结寓意真善美!")，输出“中国结寓意真善美!”。“# 输出中国结”是单行注释语句，不被执行。运行结果如下：

```
中国结寓意真善美!
```

2. 多行注释

Python 中用三个单引号(''')或者三个双引号(""")将多行注释括起来。

【代码 2-2】 使用三个单引号(''')的多行注释。

使用三个单引号(''')对一段代码进行多行注释。

```
'''
输出中国结
输出长木板
输出汉字
'''
print("中国结。")
print("长木板。")
print("汉字")
```

代码说明：使用三个单引号表示三行注释语句，即“输出中国结”“输出长木板”“输出汉字”。Python 解释器执行三条 print 语句，并输出结果。运行结果如下：

```
中国结。
长木板。
汉字
```

【代码 2-3】 使用三个双引号(""")的多行注释。

使用三个双引号(""")对一段代码进行多行注释。

```
"""
输出中国结
输出长木板
输出汉字
"""
print("中国结。")
print("长木板。")
print("汉字")
```

代码说明：使用三个双引号表示三行注释语句，即“输出中国结”“输出长木板”“输出汉字”。Python 解释器执行三条 print 语句，并输出结果。运行结果如下：

```
中国结。
长木板。
汉字
```

3. 注释的位置

为了使程序清晰易懂，注释量需要达到源程序的 20%以上，注释一般会被使用在 Python 源程序中的以下几个方面。

(1) 在 Python 文件的开头，添加安装路径和编码格式注释。

使用语句“#! /usr/bin/env python”，告诉操作系统，先到 env 里查找 Python 的安装路径，再调用解释器运行程序。

使用语句“# -*- coding: utf-8- *”，告诉 Python 编译器，源程序是使用 utf-8 编码的。

(2) 在安装路径和编码格式注释后,添加模块的功能说明。

```
"""
Shared support for scanning document type declarations in HTML and XHTML.
This module is used as a foundation for the html.parser module. It has no
documented public API and should not be used directly.
"""
```

(3) 在类名后,添加类的说明。

```
class JSONDecodeError(ValueError):
"""
Subclass of ValueError with the following additional properties:
    msg: The unformatted error message
    doc: The JSON document being parsed
    pos: The start index of doc where parsing failed
lineno: The line corresponding to pos
colno: The column corresponding to pos
"""
```

(4) 在函数名后,添加函数说明。

```
def __init__(self, filename=None, file=None, ** options):
"""
Construct a new TextFile object. At least one of 'filename'
        (a string) and 'file' (a file-like object) must be supplied.
        They keyword argument options are described above and affect
        the values returned by 'readline()'.
"""
```

(5) 在重要的代码后添加注释。

```
pos=rawdata.rindex("\n", i, j)         #  Should not fail
```

微课 2-2

2.1.2 Python 标识符

在编写 Python 程序时,需要给常量、变量、函数、模块等对象命名,这种用于标识对象的名字被称为标识符。在 Python 语言中,标识符必须遵守命名规则,即标识符可以包括英文、数字以及下划线(_),但不能以数字开头,且标识符区分大小写。例如,以下是合法的标识符:indent、check_environ、__init__、indent2、apple、APPLE。注意,apple 与 APPLE 是两个不同的标识符。

为了提高程序的可读性、可维护性、可重用性,降低程序出错的可能性,在遵守 Python 标识符命名规则的基础上,建议遵循以下约定:

- 项目的命名建议使用单词首字母大写,如 MyProject。
- 文件的命名建议使用小写,可以使用下划线,如 file_util。
- 模块的命名建议使用小写字母,且简短,尽量不使用下划线,如 util。
- 包的命名建议使用小写字母,且简短,不使用下划线,如 numpy。
- 类的命名建议使用单词首字母大写,如 StrictVersion。
- 私有类的命名建议使用下划线开头,如_StrictVersion。

- 常量的命名建议使用大写字母，如有多个单词，可以使用下划线隔开，如 PI=3.14159。
- 函数的命名建议使用小写字母，如有多个单词，可以使用下划线隔开，如 create_static_lib。
- 异常的命名建议使用 Error 作为后缀，如 KeyError。
- 全局变量的命名建议使用大写字母，如有多个单词，可以使用下划线隔开，如 MY_PATH。
- 普通变量的命名建议使用小写字母，如有多个单词，可以使用下划线隔开，如 realm。
- 只读对象的命名建议使用小写字母，并在开头和结尾添加下划线，如_id_。

在 Python 语言中，有些标识符已经被赋予了特定的含义，已被 Python 语言内部定义并保留使用，称为保留字。程序员在为变量、函数、类、模板等对象命名时，不能使用保留字。Python 的保留字有：and、as、assert、break、class、continue、def、del、elif、else、except、finally、for、from、False、global、if、import、in、is、lambda、nonlocal、not、None、or、pass、raise、return、try、True、while、with、yield。

Python 中的保留字同样区分大小写，如 if 是保留字，而 IF 不是保留字。

任务 2.2　变量与输入/输出控制

【任务描述】

在中国结案例中，通过变量记录中国结的尺寸并显示位置的 x 坐标、y 坐标。

【任务分析】

(1) 掌握变量的定义及使用方法。

(2) 掌握变量的输入输出方法。

微课 2-3

1. 变量的简单赋值

中国结的结心红线尺寸以及基准点的横纵坐标的值，根据不同的应用需求，可以灵活改变，这种值可以变化的量，称为变量。Python 语言实现变量的方法是，在计算机内存中开辟一个空间存放“值”，并将该内存空间的地址存放在另一个内存空间中，这个存放地址的内存空间就是变量。

现在通过 Python 语言实现一个结心红线尺寸为 100 且基准点横坐标为－200、基准点纵坐标为 200 的中国结。首先，按照标识符的命名规则，为中国结的基准点横纵坐标及结心红线尺寸分别取名为 x、y、size，然后为三个变量分别赋予一个值。变量赋值的语句格式为：变量名＝变量值。

【代码 2-4】 变量的简单赋值。

```
1    x=-200              # 变量 x 赋值为-200
2    y=200               # 变量 y 赋值为 200
3    size=100            # 变量 size 赋值为 100
```

代码说明：“＝”是赋值符号，它将右边的值赋给左边的变量，Python 语言中变量要赋值后才能在内存中创建。在 Python 语言中，变量没有数据类型，同一个变量可以赋值任何数据，变量通过对内存地址的引用实现值的读取和修改。变量的简单赋值结果如图 2-1 所示。

变量可以多次赋值，并保留最新一次的赋值。例如，需要将中国结的结心红线尺寸修改为 80，可以重新为变量 size 赋值，即 size＝80，变量多次赋值结果如图 2-2 所示。