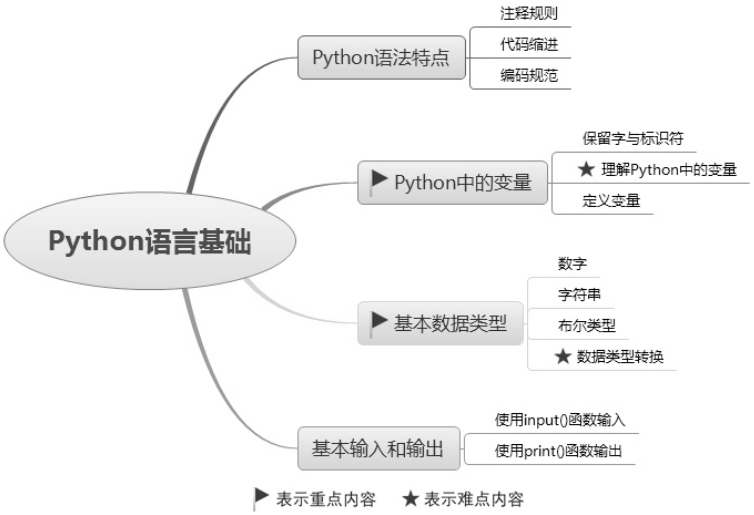
第 2 章

Python 语言基础

熟练掌握一种编程语言，最好的方法就是充分了解它，并掌握其基础知识，另外，还需要亲自体验，多编写代码，方可熟能生巧。

从本章开始，我们将正式踏上 Python 开发之旅，体验 Python 带给我们的简单、快乐。本章将先对 Python 的语法特点进行详细介绍，然后介绍 Python 中的保留字、标识符、变量、基本数据类型，以及数据类型间的转换，最后介绍如何通过输入和输出函数进行交互。

本章知识架构及重难点如下。



2.1 Python 语法特点

学习 Python 需要了解它的语法特点，如注释规则、代码缩进、编码规范等。下面将对学习 Python 时首先需要了解的这些语法特点进行详细介绍。

2.1.1 注释规则

注释类似于语文课本中古诗文的注释，如图 2.1 所示。据此，所谓注释，就是在代码中添加标注性的文字，进而帮助程序员更好地阅读代码。注释的内容将被 Python 解释器忽略，并不会在执行结果中体现出来。

在 Python 中，通常包括 3 种类型的注释，分别是单行注释、多行注释和文件编码声明注释。这些注释在 IDLE 中的效果如图 2.2 所示。

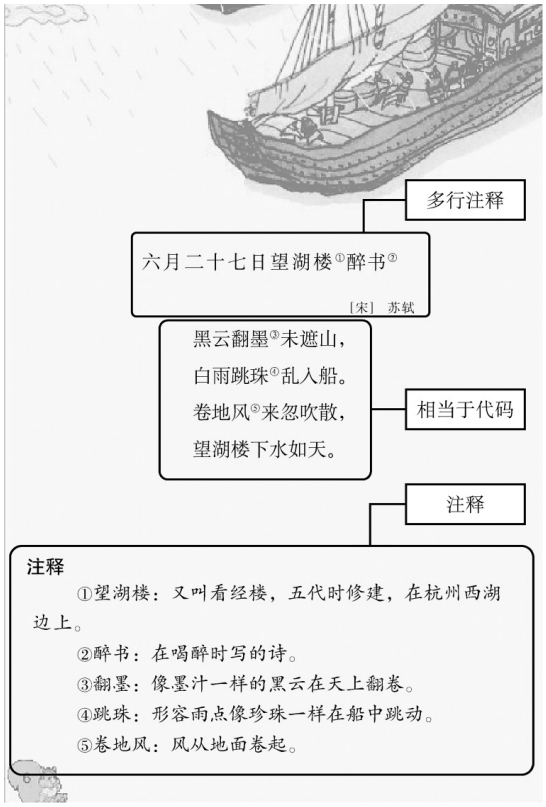


图 2.1　古诗文的注释

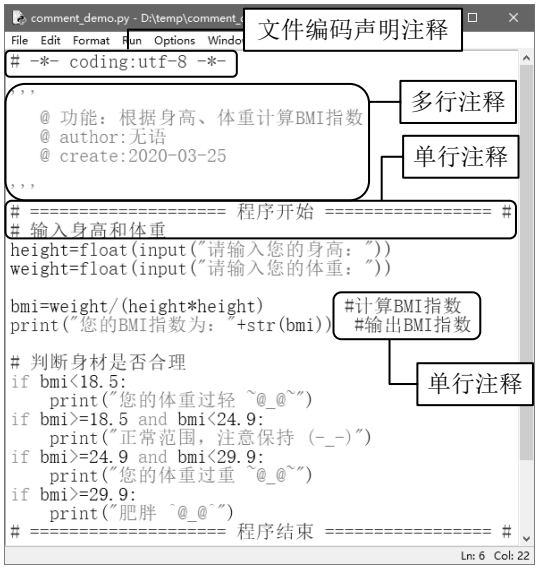


图 2.2　Python 中的注释

1．单行注释

在 Python 中，使用#作为单行注释的符号。从符号#开始直到换行为止，其后面所有的内容都作为注释的内容而被 Python 编译器忽略。

语法如下：

```
# 注释内容
```

单行注释可以放在要注释的代码的前一行，也可以放在要注释的代码的右侧。例如，下面的两种注释形式都是正确的。

第一种形式：

```
# 要求输入身高，单位为 m，如 1.70
height=float(input("请输入您的身高："))
```

第二种形式：

```
height=float(input("请输入您的身高："))          # 要求输入身高，单位为 m，如 1.70
```

上述两种形式的代码，其运行结果都将如图 2.3 所示。

```
请输入您的身高：1.70
>>>
```

图 2.3　运行结果

说明

在添加注释时，一定要有意义，即注释能充分体现代码的作用。例如，图 2.4 中的注释就是冗余的注释。如果将其注释修改为如图 2.5 所示的注释，则可很清楚地知道代码的作用。

```
bmi=weight/(height*height)        #Magic，请勿改动
```

图 2.4　冗余的注释

```
bmi=weight/(height*height)      # 用于计算BMI指数，公式为“体重/(身高×身高)”
```

图 2.5　推荐的注释

单行注释可以出现在代码的任意位置，但是不能分隔关键字和标识符。例如，下列代码注释是错误的：

```
height=float(#要求输入身高 input("请输入您的身高："))
```

说明

在 IDLE 开发环境中，可以通过选择主菜单中的 Format→Comment Out Region 菜单项（也可直接使用快捷键 Alt+3），将选中的代码注释掉；也可通过选择主菜单中的 Format→UnComment Region 菜单项（也可直接使用快捷键 Alt+4），取消添加的单行注释。

2．多行注释

在 Python 中并没有一个单独的多行注释标记，而是将包含在一对三引号（即'''……'''或者"""……"""）中，并且不属于任何语句的内容则认为是注释。对于这样的代码，将被解释器忽略。由于这样的代码可以分为多行编写，因此也作为多行注释。

语法格式如下：

```
'''
    注释内容 1
    注释内容 2
    ……
'''
```

或者

```
"""
    注释内容 1
    注释内容 2
    ……
"""
```

误区警示

在使用三引号作为注释时，需要注意，三引号必须成对出现，如果只写一半三引号，那么当程序运行时，将会提示 EOF while scanning triple-quoted string literal 错误。

多行注释通常用来为 Python 文件、模块、类或者函数等添加版权、功能等信息。例如，下列代码将使用多行注释为 demo.py 文件添加版权、功能及修改日志等信息：

```
'''
    @ 版权所有：吉林省明日科技有限公司©版权所有
    @ 文件名：demo.py
    @ 文件功能描述：根据身高、体重计算 BMI 指数
    @ 创建日期：2021 年 1 月 31 日
    @ 创建人：无语
    @ 修改标识：2021 年 2 月 2 日
    @ 修改描述：增加根据 BMI 指数判断体重是否合理的功能代码
    @ 修改日期：2021 年 2 月 2 日
'''
```

注意

如果三引号'''……'''或者"""……"""出现在语句中，那么就不是注释，而是字符串，这一点要注意区分。例如，图 2.6 中的代码即为多行注释，而图 2.7 中的代码即为字符串。

```
'''
    @ 功能：根据身高、体重计算BMI指数
    @ author:无语
    @ create:2021-03-25
'''
```

图 2.6　三引号为多行注释

```
print('''根据身高、体重计算BMI指数''')
```

图 2.7　三引号为字符串

3．文件编码声明注释

在 Python 3 中，默认采用的文件编码是 UTF-8。这种编码支持世界上大多数语言的字符，也包括中文。如果不想使用默认编码，就需要在文件的第一行声明文件的编码，也就是需要使用文件编码声明注释。

语法格式如下：

```
# -*- coding:编码 -*-
```

或者

```
#coding=编码
```

在上述语法中，编码为文件所使用的字符编码类型，如果采用 GBK，则设置为 gbk 或 cp936。例如，指定编码为 GBK，可以使用以下中文注释：

```
# -*- coding:gbk -*-
```

说明

在上述代码中，“-*-”没有特殊的作用，只是为了美观才加上的。所以上述代码也可以使用“# coding:gbk”代替。

另外，以下代码也是正确的中文注释：

```
#coding=gbk
```

2.1.2　代码缩进

Python 不像其他程序设计语言（如 Java 或者 C 语言）那样采用大括号“{}”分隔代码块，而是采用代码缩进和冒号“:”区分代码之间的层次。

说明

缩进可以使用空格或者 Tab 键实现。其中，如果使用空格，则通常情况下采用 4 个空格作为一个缩进量；而如果使用 Tab 键，则采用一个 Tab 键作为一个缩进量。通常情况下建议采用空格进行缩进。

在 Python 中，对于类定义、函数定义、流程控制语句，以及异常处理语句等，行尾的冒号和下一行的缩进表示一个代码块的开始；而缩进结束，则表示一个代码块的结束。

例如，以下代码中的缩进即为正确的缩进。

```
height=float(input("请输入您的身高："))          # 输入身高
weight=float(input("请输入您的体重："))          # 输入体重
bmi=weight/(height*height)                       # 计算 BMI 指数

# 判断体重是否合理
if bmi<18.5:
    print("您的 BMI 指数为："+str(bmi))          # 输出 BMI 指数
    print("体重过轻 ~@_@~")
if bmi>=18.5 and bmi<24.9:
    print("您的 BMI 指数为："+str(bmi))          # 输出 BMI 指数
    print("正常范围，注意保持 (-_-)")
if bmi>=24.9 and bmi<29.9:
    print("您的 BMI 指数为："+str(bmi))          # 输出 BMI 指数
    print("体重过重 ~@_@~")
if bmi>=29.9:
    print("您的 BMI 指数为："+str(bmi))          # 输出 BMI 指数
    print("肥胖 ^@_@^")
```

Python 对代码的缩进要求非常严格，同一个级别的代码块的缩进量必须相同。如果不采用合理的代码缩进，将抛出 SyntaxError 异常。例如，代码中有的缩进量是 4 个空格，还有的是 2 个空格，这样就会导致出现 SyntaxError 错误，如图 2.8 所示。

在 IDLE 开发环境中，一般以 4 个空格作为代码的基本缩进单位。不过也可以选择 Options→Configure IDLE 菜单项，在打开的 Settings 对话框的 Fonts/Tabs 选项卡中修改代码的基本缩进量，如图 2.9 所示。

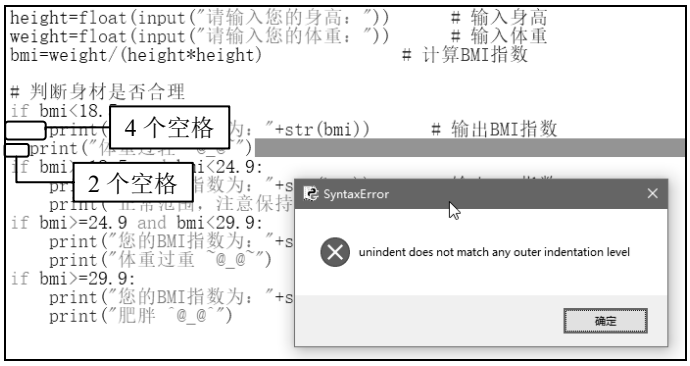


图 2.8　缩进量不同导致的 SyntaxError 错误

图 2.9　修改基本缩进量

2.1.3　编码规范

下面给出两段实现同样功能的代码，如图 2.10 所示。

```
'''@ 功能：根据身高、体重计算BMI指数
   @ author:无语
   @ create:2021-03-25'''
# 输入身高和体重
height=float(input("请输入您的身高："))
weight=float(input("请输入您的体重："))
bmi=weight/(height*height)# 计算BMI指数
print("您的BMI指数为："+str(bmi))# 输出BMI指数
# 判断体重是否合理
if bmi<18.5:
    print("您的体重过轻 ~@_@~")
if bmi>=18.5 and bmi<24.9:
    print("正常范围，注意保持 (-_-)")
if bmi>=24.9 and bmi<29.9:
    print("您的体重过重 ~@_@~")
if bmi>=29.9:
    print("肥胖 ^@_@^")
```

```
'''
   @ 功能：根据身高、体重计算BMI指数
   @ author:无语
   @ create:2021-03-25
'''
# 输入身高和
height=float(input("请输入您的身高："))
weight=float(input("请输入您的体重："))
bmi=weight/(height*height)          # 计算BMI指数
print("您的BMI指数为："+str(bmi))  # 输出BMI指数
# 判断体重是否合理
if bmi<18.5:
    print("您的体重过轻 ~@_@~")
if bmi>=18.5 and bmi<24.9:
    print("正常范围，注意保持 (-_-)")
if bmi>=24.9 and bmi<29.9:
    print("您的体重过重 ~@_@~")
if bmi>=29.9:
    print("肥胖 ^@_@^")
```

图 2.10　两段功能相同的 Python 代码

大家在学习时，愿意看到图 2.10 中的左侧代码还是右侧代码？答案应该是一致的，大家肯定都喜

欢阅读图 2.10 中右侧的代码，因为它看上去更加规整，这是一种最基本的代码编写规范。遵循一定的代码编写规则和命名规范可以使代码更加规范化，对代码的理解与维护起到至关重要的作用。

本节将对 Python 代码的编写规则以及命名规范进行介绍。

1．编写规则

Python 中采用 PEP 8 作为编码规范，其中 PEP 是 Python Enhancement Proposal 的缩写，翻译过来是 Python 增强建议书，而 PEP 8 表示版本，它是 Python 代码的样式指南。下面给出 PEP 8 编码规范中的一些应该严格遵守的条目。

- ☑ 每个 import 语句只导入一个模块，尽量避免一次导入多个模块。图 2.11 为推荐的写法，而图 2.12 为不推荐的写法。
- ☑ 不要在行尾添加分号“;”，也不要用分号将两条命令放在同一行。例如，图 2.13 中的代码为不规范的写法。

```
import os
import sys
```

图 2.11　推荐的写法

```
import os, sys
```

图 2.12　不推荐的写法

```
height = float(input("请输入您的身高："));
weight = float(input("请输入您的体重："));
```

图 2.13　不规范的写法

- ☑ 建议每行不超过 80 个字符，如果超过，建议使用小括号“()”将多行内容隐式地连接起来，而不推荐使用反斜杠“\”进行连接。例如，如果一个字符串文本在一行上显示不下，那么可以使用小括号“()”将其分行显示，代码如下：

```
print("我一直认为我是一只蜗牛。我一直在爬，也许还没有爬到金字塔的顶端。"
      "但是只要你在爬，就足以给自己留下令生命感动的日子。")
```

例如，以下通过反斜杠“\”进行连接的做法是不推荐使用的。

```
print("我一直认为我是一只蜗牛。我一直在爬，也许还没有爬到金字塔的顶端。\
但是只要你在爬，就足以给自己留下令生命感动的日子。")
```

不过以下两种情况除外：导入模块的语句过长；注释里的 URL。

- ☑ 使用必要的空行可以增加代码的可读性。一般在顶级定义（如函数或者类的定义）之间空两行，而方法定义之间空一行。另外，在用于分隔某些功能的位置也可以空一行。
- ☑ 通常情况，运算符两侧、函数参数之间、逗号“,”两侧建议使用空格进行分隔。
- ☑ 应该避免在循环中使用+和+=操作符累加字符串。这是因为字符串是不可变的，这样做会创建不必要的临时对象。推荐的做法是将每个子字符串加入列表中，然后在循环结束后使用 join() 方法连接列表。
- ☑ 适当使用异常处理结构提高程序容错性，但不能过多依赖异常处理结构，适当的显式判断还是必要的。

说明

在编写 Python 程序时，建议严格遵循 PEP 8 编码规范。完整的 Python 编码规范请参考 PEP 8。

2．命名规范

命名规范在编写代码中起到很重要的作用，虽然不遵循命名规范，程序也可以运行，但是使用命名规范可以更加直观地了解代码所代表的含义。下面将介绍 Python 中常用的一些命名规范。

- ☑ 模块名尽量短小，并且使用全部小写字母，可以使用下画线分隔多个字母。例如，game_main、game_register、bmiexponent 都是推荐使用的模块名称。
- ☑ 包名尽量短小，并且使用全部小写字母，不推荐使用下画线。例如，com.mingrisoft、com.mr、com.mr.book 都是被推荐使用的包名称，而 com_mingrisoft 则是不被推荐使用的。
- ☑ 类名采用单词首字母大写形式（即 Pascal 风格）。例如，定义一个借书类，可以命名为 BorrowBook。

说明

Pascal 是以纪念法国数学家 Blaise Pascal 而命名的一种编程语言，Python 中的 Pascal 命名法就是根据该语言的特点总结出来的一种命名方法。

- ☑ 模块内部的类采用下画线“_”+Pascal 风格的类名组成。例如，在 BorrowBook 类中的内部类，可以使用_BorrowBook 命名。
- ☑ 函数、类的属性和方法的命名规则同模块类似，也是全部采用小写字母，多个字母间用下画线“_”分隔。
- ☑ 常量命名时采用全部大写字母，可以使用下画线。
- ☑ 使用双下画线“__”开头的实例变量或方法是类私有的。

2.2 Python 中的变量

2.2.1 保留字与标识符

在学习变量之前，先了解什么是保留字和标识符。

1．保留字

保留字是 Python 中已经被赋予特定意义的一些单词，开发程序时，不可以把这些保留字作为变量、函数、类、模块和其他对象的名称来使用。Python 中的保留字如表 2.1 所示。

表 2.1 Python 中的保留字

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			

注意

Python 中所有保留字是区分字母大小写的。例如，if 是保留字，但是 IF 就不属于保留字，如图 2.14 所示。

```
>>> true="真"
>>> True="真"
SyntaxError: can't assign to keyword
>>>
```

```
>>> if = "守得云开见月明"
SyntaxError: invalid syntax
>>> IF = "守得云开见月明"
>>>
```

图 2.14　Python 中的保留字区分字母大小写

Python 中的保留字可以通过在 IDLE 中输入以下两行代码予以查看：

```
01　import keyword
02　keyword.kwlist
```

执行结果如图 2.15 所示。

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', '
def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if',
 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'retu
rn', 'try', 'while', 'with', 'yield']
>>>
```

图 2.15　查看 Python 中的保留字

误区警示

如果在开发程序时，使用 Python 中的保留字作为模块、类、函数或者变量等的名称，如下面代码为使用 Python 保留字 if 作为变量的名称：

```
01　if = "坚持下去不是因为我很坚强，而是因为我别无选择"
02　print(if)
```

运行时则会出现如图 2.16 所示的错误提示信息。

```
if = "坚持下去不是因为我很坚强，而是因为我别无选择"
print(if)
```

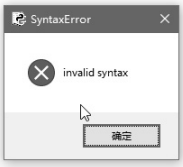


图 2.16　使用 Python 保留字作为变量名时的错误信息

2．标识符

标识符可以简单地理解为一个名字，比如每个人都有自己的名字，它主要用来标识变量、函数、类、模块和其他对象的名称。

Python 语言标识符命名规则如下。

（1）由字母、下画线“_”和数字组成，并且第一个字符不能是数字。当前 Python 中只允许使用

ISO-Latin 字符集中的字符 A~Z 和 a~z。

（2）不能使用 Python 中的保留字。

例如，下面是合法的标识符：

```
USERID
name
model2
user_age
```

下面是非法的标识符：

```
4word          # 以数字开头
try            # Python 中的保留字
$money         # 不能使用特殊字符$
```

注意

Python 的标识符中不能包含空格、@、%和$等特殊字符。

（3）区分字母大小写。在 Python 中，标识符中的字母是严格区分大小写的，两个同样的单词，如果大小写格式不一样，那么所代表的意义是完全不同的。例如，下面 3 个变量是完全独立、毫无关系的，就像 3 个长得比较像的人，彼此之间都是独立的个体。

```
01  number = 0      # 全部小写
02  Number = 1      # 部分大写
03  NUMBER = 2      # 全部大写
```

（4）Python 中以下画线开头的标识符有特殊意义，一般应避免使用相似的标识符。

- ☑ 以单下画线开头的标识符（如_width）表示不能直接访问的类属性。另外，也不能通过 from xxx import *导入。
- ☑ 以双下画线开头的标识符（如__add）表示类的私有成员。
- ☑ 以双下画线开头和结尾的是 Python 中专用的标识。例如，__init__()表示构造函数。

说明

在 Python 中允许使用汉字作为标识符，如“我的名字="明日科技"”，在程序运行时并不会出现错误，如图 2.17 所示。但建议读者尽量不要使用汉字作为标识符。

```
>>> 我的名字="明日科技"
>>> print(我的名字)
明日科技
>>>
```

图 2.17　使用汉字作为标识符

2.2.2　理解 Python 中的变量

在 Python 中，严格意义上变量应该称为“名字”，也可以理解为标签。当把一个值赋给一个名字（如把值“学会 Python 还可以飞”赋给 python）时，python 就称为变量。在大多数编程语言中，都将其称为“把值存储在变量中”。意思是在计算机内存中的某个位置，字符串序列“学会 Python 还可以飞”已经存在。你不需要准确地知道它们到底在哪里，只需要告诉 Python 这个字符串序列的名字是 python，然后就可以通过这个名字来引用这个字符串序列。这个过程就像上门取快递一样，内存就像

一个巨大的货物架，在 Python 中使用变量就像是给快递盒子加标签，如图 2.18 所示。

你的快递存放在货物架上，上面附着写有你名字的标签。当你来取快递时，并不需要知道它们存放在这个大型货架的具体哪个位置，只需要提供你的名字，快递员就会把你的快递交还给你。实际上，你的快递可能并不在原先所放的位置。不过快递员会为你记录快递的位置。要取回你的快递，只需要提供你的名字。变量也一样，你不需要准确地知道信息存储在内存中的哪个位置，只需要记住存储变量时所用的名字，再使用这个名字即可。

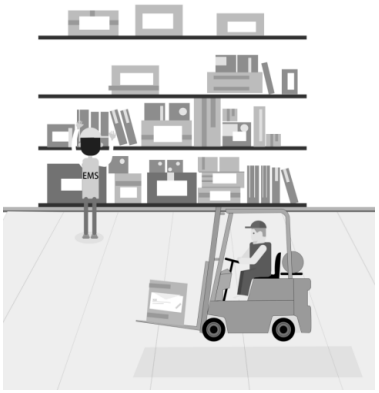


图 2.18　货物架中贴着标签的快递

2.2.3　定义变量

在 Python 中，不需要先声明变量名及其类型，直接赋值即可创建各种类型的变量。需要注意的是，对于变量的命名并不是任意的，应遵循以下几条规则。

- ☑ 变量名必须是一个有效的标识符。
- ☑ 变量名不能使用 Python 中的保留字。
- ☑ 慎用小写字母 l 和大写字母 O。
- ☑ 应选择有意义的单词作为变量名。

为变量赋值可以通过等号“=”来实现。语法格式如下：

```
变量名 = value;
```

例如，创建一个整型变量，并为其赋值为 1024，可以使用下列语句：

```
number = 1024                    # 创建变量 number 并赋值为 1024，该变量为数值型
```

这样创建的变量就是数值型的变量。如果直接为变量赋值一个字符串值，那么该变量即为字符串类型，如下列语句：

```
nickname = "碧海苍梧"             # 字符串类型的变量
```

误区警示

在 Python 中，输入代码时，除非在字符串中有全角空格，否则一定不要用全角空格。这个错误比较隐蔽，不容易被发现，所以我们要养成好的编码习惯。

另外，Python 是一种动态类型的语言，也就是说，变量的类型可以随时变化。例如，在 IDLE 中，创建变量 nickname，并赋值为字符串“碧海苍梧”，然后输出该变量的类型，可以看到该变量为字符串类型，再为变量赋值为数值 1024，并输出该变量的类型，可以看到该变量为整型。执行过程如下：

```
>>> nickname = "碧海苍梧"          # 字符串类型的变量
>>> print(type(nickname))
<class 'str'>
>>> nickname = 1024               # 整型的变量
>>> print(type(nickname))
<class 'int'>
```

说明

在 Python 语言中，使用内置函数 type()可以返回变量类型。

在 Python 中，允许多个变量指向同一个值。将两个变量都赋值为数字 2048，再分别应用内置函数 id()获取变量的内存地址，将得到相同的结果。执行过程如下：

```
>>> no = number = 2048
>>> id(no)
50766992
>>> id(number)
50766992
```

在上述代码中，id()为 Python 的内置函数，使用它可以返回变量所指的内存地址。

注意

常量就是在程序运行过程中，值不能改变的量，诸如现实生活中的居民身份证号码、数学运算中的π值等，这些都是不会发生改变的，它们都可以定义为常量。在 Python 中，并没有提供定义常量的保留字。不过在 PEP 8 规范中定义了常量的命名规范由大写字母和下画线组成，但是在实际项目中，常量首次赋值后，还是可以被其他代码修改。

2.3 基本数据类型

在内存中可以使用多种类型存储数据。例如，一个人的姓名可以用字符型存储，年龄可以使用数值型存储，而婚否可以使用布尔类型存储。这些都是 Python 中提供的基本数据类型。下面将对这些基本数据类型进行详细介绍。

2.3.1 数字

在程序开发时，经常使用数字记录游戏的得分、网站的销售数据和网站的访问量等信息。在 Python 中，提供了数字类型用于保存这些数值，并且它们是不可改变的数据类型。如果修改数字类型变量的值，那么会先把该值存储到内容中，然后修改变量让其指向新的内存地址。

在 Python 中，数字类型主要包括整数、浮点数和复数。下面分别介绍。

1．整数

整数用来表示整数数值，即没有小数部分的数值。在 Python 中，整数包括正整数、负整数和 0，并且它的位数是任意的（当超过计算机自身的计算功能时，会自动转用高精度计算），如果要指定一个非常大的整数，只需要写出其所有位数即可。

整数类型包括十进制整数、八进制整数、十六进制整数和二进制整数。下面分别进行介绍。

（1）十进制整数。十进制整数的表现形式大家都很熟悉。例如，以下数值都是有效的十进制整数：

```
3141592653589793238462 6
```

```
66666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
66666666666666666666666666666666666666666
-2017
0
```

在 IDLE 中，执行的结果如图 2.19 所示。

```
>>> 3.1415926535897932384626
3.141592653589793
>>> 66666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
66666666666666666666666666666666666666666
>>> -2048
-2048
>>> 0
0
>>>
```

图 2.19　有效的整数

注意

不能以 0 作为十进制数的开头（0 除外）。

（2）八进制整数。由 0～7 组成，进位规则是“逢八进一”，并且以 0o/0O 开头的数，如 0o123（转换成十进制数为 83）、−0o123（转换成十进制数为−83）。

（3）十六进制整数。由 0～9、A～F 组成，进位规则是“逢十六进一”，并且以 0x/0X 开头的数，如 0x25（转换成十进制数为 37）、0Xb01e（转换成十进制数为 45086）。

（4）二进制整数。只有 0 和 1 两个基数，进位规则是“逢二进一”，如 101（转换为十制数为 5）、1010（转换为十进制为 10）。

2．浮点数

浮点数由整数部分和小数部分组成，主要用于处理包括小数的数，如 1.414、0.5、−1.732、3.141592653589793238462 6 等。浮点数也可以使用科学记数法表示，如 2.7e2、−3.14e5 和 6.16e−2 等。

注意

在使用浮点数进行计算时，可能会出现小数位数不确定的情况。例如，计算 0.1+0.1 时，将得到想要的 0.2，而计算 0.1+0.2 时，将得到 0.30000000000000004（想要的结果为 0.3），执行过程如下：

```
>>> 0.1+0.1
0.2
>>> 0.1+0.2
0.30000000000000004
```

对于这种情况，所有语言都存在这个问题，暂时忽略多余的小数位数即可。

【例 2.1】根据身高、体重计算 BMI 指数。**（实例位置：资源包\TM\sl\02\01）**

在 IDLE 中创建一个名称为 bmiexponent.py 的文件，然后在该文件中定义两个变量：一个用于记录身高，单位为 m；另一个用于记录体重，单位为 kg。根据公式“BMI=体重/（身高×身高）”计算 BMI 指数，代码如下：

```
height = 1.70                                  # 保存身高的变量，单位为 m
print("您的身高：" + str(height))
weight = 48.5                                  # 保存体重的变量，单位为 kg
print("您的体重：" + str(weight))
bmi=weight/(height*height)                     # 用于计算 BMI 指数，公式为“体重/（身高×身高）”
print("您的 BMI 指数为："+str(bmi))             # 输出 BMI 指数
# 判断体重是否合理
```

```
if bmi<18.5:
    print("您的体重过轻 ~@_@~")
if bmi>=18.5 and bmi<24.9:
    print("正常范围，注意保持 (-_-)")
if bmi>=24.9 and bmi<29.9:
    print("您的体重过重 ~@_@~")
if bmi>=29.9:
    print("肥胖 ^@_@^")
```

说明

在上述代码中，str()函数用于将数值转换为字符串；if 语句用于进行条件判断，将在 4.2 节中进行详细介绍。

运行结果如图 2.20 所示。

```
您的身高：1.7
您的体重：48.5
您的BMI指数为：16.782006920415228
您的体重过轻 ~@_@~
>>>
```

图 2.20　根据身高、体重计算 BMI 指数

3．复数

Python 中的复数与数学中的复数的形式完全一致，都是由实部和虚部组成，并且使用 j 或 J 表示虚部。当表示一个复数时，可以将其实部和虚部相加。例如，一个复数实部为 3.14，虚部为 12.5j，那么这个复数为 3.14+12.5j。

2.3.2　字符串

字符串就是连续的字符序列，可以是计算机所能表示的一切字符的集合。在 Python 中，字符串属于不可变序列，通常使用单引号（' '）、双引号（" "）或者三引号（''' '''或""" """）括起来。这 3 种引号形式在语义上没有差别，只是在形式上有些差别。其中，单引号和双引号中的字符序列必须在一行上；而三引号中的字符序列可以分布在连续的多行上。例如，定义 3 个字符串类型变量，并且应用 print()函数输出，代码如下：

```
title = '我喜欢的名言警句'                                    # 使用单引号，其中的字符序列必须在一行上
mot_cn = "命运给予我们的不是失望之酒，而是机会之杯。"# 使用双引号，其中的字符序列必须在一行上
# 使用三引号，其中的字符序列可以分布在多行上
mot_en = '''Our destiny offers not the cup of despair,
but the chance of opportunity.'''
print(title)
print(mot_cn)
print(mot_en)
```

执行结果如图 2.21 所示。

误区警示

字符串开始和结尾使用的引号形式必须一致。另外，当需要表示复杂的字符串时，还可以进行引号的嵌套。例如，下面的字符串也都是合法的。

```
'在 Python 中也可以使用双引号（" "）定义字符串'
'''(··)nnn'也是字符串"
"""---' "_"***"""
```

【例 2.2】输出字符画——坦克。**（实例位置：资源包\TM\sl\02\02）**

在 IDLE 中创建一个名称为 ascii_art.py 的文件，然后在该文件中输出一个表示字符画的字符串，由于该字符画有多行，所以需要使用三引号作为字符串的定界符。关键代码如下：

```
print('''
                                ▸  学编程，你不是一个人在战斗~~
                                |
                           __\--__|_
II=======OOOOO[/  ★007___|
                  _____\_____|/-----.
              /___mingrisoft.com___|
               \◎◎◎◎◎◎◎◎◎⊙/
                  ~~~~~~~~~~~~~~~~
''')
```

运行结果如图 2.22 所示。

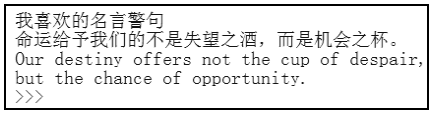


图 2.21　使用 3 种形式定义字符串

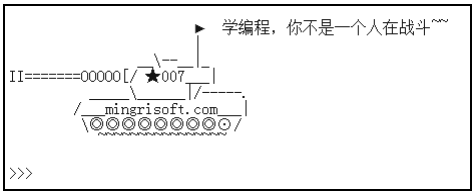


图 2.22　输出字符画

Python 中的字符串还支持转义字符。所谓转义字符，是指使用反斜杠“\”对一些特殊字符进行转义。常用的转义字符及其说明如表 2.2 所示。

表 2.2　常用的转义字符及其说明

转 义 字 符	说　　明	转 义 字 符	说　　明
\	续行符	\'	单引号
\n	换行符	\\	一个反斜杠
\0	空	\f	换页
\t	水平制表符，用于横向跳到下一制表位	\0dd	八进制数，dd 代表字符，如\012 代表换行
\"	双引号	\xhh	十六进制数，hh 代表字符，如\x0a 代表换行

注意

在字符串定界符引号的前面加上字母 r（或 R），那么该字符串将原样输出，其中的转义字符将不进行转义输出。例如，输出字符串“"失望之酒\x0a 机会之杯"”将转义字符换行输出；而输出字符串“r"失望之酒\x0a 机会之杯"”，则原样输出，执行结果如图 2.23 所示。

```
>>> print("失望之酒\x0a机会之杯")
失望之酒
机会之杯
>>> print(r"失望之酒\x0a机会之杯")
失望之酒\x0a机会之杯
>>>
```

图 2.23　转义输出和原样输出的对比

2.3.3 布尔类型

布尔类型主要用来表示真或假的值。在 Python 中，标识符 True 和 False 被解释为布尔值。另外，Python 中的布尔值可以转换为数值，其中 True 表示 1，而 False 则表示 0。

说明

在 Python 中，可以对布尔类型的值进行数值运算，例如，“False + 1”的结果为 1。但是不建议对布尔类型的值进行数值运算。

在 Python 中，对所有的对象都可以进行真值测试。其中，只有下面列出的 4 种情况得到的值为假，其他对象在 if 或者 while 语句中都表现为真。

- ☑ False 或 None。
- ☑ 数值中的零，包括 0、0.0、虚数 0。
- ☑ 空序列，包括字符串、空元组、空列表、空字典。
- ☑ 自定义对象的实例，该对象的__bool__方法返回 False，或者__len__方法返回 0。

2.3.4 数据类型转换

Python 是动态类型的语言（也称为弱类型语言），不需要像 Java 或者 C 语言一样在使用变量前必须先声明变量的类型。虽然 Python 不需要先声明变量的类型，但有时仍然需要用到类型转换。例如，在例 2.1 中，要想通过一个 print()函数输出提示文字“您的身高：”和浮点型变量 height 的值，就需要将浮点型变量 height 转换为字符串；否则将显示如图 2.24 所示的错误。

```
Traceback (most recent call last):
  File "D:\demo.py", line 11, in <module>
    print("您的身高：" + height)
TypeError: can only concatenate str (not "float") to str
>>>
```

图 2.24　字符串和浮点型变量连接时出错

在 Python 中，提供了如表 2.3 所示的函数进行各数据类型间的转换。

表 2.3　常用的数据类型转换函数及其作用

函　数	作　用
int(x)	将 x 转换成整数类型
float(x)	将 x 转换成浮点数类型
complex(real [,imag])	创建一个复数
str(x)	将 x 转换为字符串
repr(x)	将 x 转换为表达式字符串
eval(str)	计算在字符串中的有效 Python 表达式，并返回一个对象
chr(x)	将整数 x 转换为一个字符
ord(x)	将一个字符 x 转换为它对应的整数值
hex(x)	将一个整数 x 转换为一个十六进制的字符串
oct(x)	将一个整数 x 转换为一个八进制的字符串
bin(x)	将一个整数 x 转换为一个二进制字符串

【例 2.3】模拟超市的抹零结账。**（实例位置：资源包\TM\sl\02\03）**

假设某超市因为找零麻烦，特设抹零行为。现编写一段 Python 代码，实现模拟超市的这种带抹零的结账行为。

在 IDLE 中创建一个名称为 erase_zero.py 的文件，然后在该文件中，首先将各个商品金额累加，计算出商品总金额，并转换为字符串输出；然后再应用 int()函数将浮点型的变量转换为整型，从而实现抹零，并转换为字符串输出。关键代码如下：

```
01  money_all = 56.7 + 72.9 + 88.5 + 26.6 + 68.8        # 累加总计金额
02  money_all_str = str(money_all)                      # 转换为字符串
03  print("商品总金额为：" + money_all_str)
04  money_real = int(money_all)                         # 进行抹零处理
05  money_real_str = str(money_real)                    # 转换为字符串
06  print("实收金额为：" + money_real_str)
```

运行结果如图 2.25 所示。

```
商品总金额为：313.5
实收金额为：313
>>>
```

图 2.25　模拟超市抹零结账行为

误区警示

在进行数据类型转换时，如果把一个非数字字符串转换为整型，将产生如图 2.26 所示的错误。

```
>>> int("17天")
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    int("17天")
ValueError: invalid literal for int() with base 10: '17天'
```

图 2.26　将非数字字符串转换为整型产生的错误

2.4　基本输入和输出

从第 1 章的 Hello World 程序开始，我们一直在使用 print()函数向屏幕上输出一些字符，这就是 Python 的基本输出函数。除了 print()函数，Python 还提供了一个用于进行标准输入的函数，即 input()。input()函数用于接收用户通过键盘输入的内容。下面将对这两个函数进行详细介绍。

2.4.1　使用 input()函数输入

在 Python 中，使用内置的函数 input()可以接收用户通过键盘输入的内容。input()函数的基本用法如下：

```
variable = input("提示文字")
```

其中，variable 为保存输入结果的变量，双引号内的文字是用于提示用户要输入的内容的。例如，想要接收用户输入的内容，并保存到变量 tip 中，可以使用以下代码：

```
tip = input("请输入文字：")
```

在 Python 3.x 中，无论输入的是数字还是字符都将被作为字符串读取。如果想要接收数值，需要把接收到的字符串进行类型转换。例如，想要接收整型的数字并保存到变量 age 中，可以使用以下代码：

```
age = int(input("请输入数字："))
```

【例 2.4】根据身高、体重计算 BMI 指数（改进版）。**（实例位置：资源包\TM\sl\02\04）**

在 2.3.1 节的例 2.1 中，实现根据身高、体重计算 BMI 指数时，身高和体重是固定的，下面将其修改为使用 input()函数进行输入，修改后的代码如下：

```
height = float(input("请输入您的身高（单位为 m）："))    # 输入身高，单位为 m
weight = float(input("请输入您的体重（单位为 kg）："))   # 输入体重，单位为 kg
bmi=weight/(height*height)                              # 用于计算 BMI 指数，公式为“体重/（身高×身高）”
print("您的 BMI 指数为："+str(bmi))                      # 输出 BMI 指数
# 判断体重是否合理
if bmi<18.5:
    print("您的体重过轻 ~@_@~")
if bmi>=18.5 and bmi<24.9:
    print("正常范围，注意保持 (-_-)")
if bmi>=24.9 and bmi<29.9:
    print("您的体重过重 ~@_@~")
if bmi>=29.9:
    print("肥胖 ^@_@^")
```

运行结果如图 2.27 所示。

```
请输入您的身高（单位为m）：1.70
请输入您的体重（单位为kg）：49
您的BMI指数为：16.955017301038065
您的体重过轻 ~@_@~
>>>
```

图 2.27　根据身高和体重计算 BMI 指数

2.4.2　使用 print()函数输出

在 Python 中，默认情况下，使用内置的函数 print()可以将结果输出到 IDLE 中或者标准控制台上。其基本语法格式如下：

```
print(输出内容)
```

其中，输出内容可以是数字和字符串（使用引号括起来），此类内容将直接输出；也可以是包含运算符的表达式，此类内容将计算结果输出。例如：

```
a = 10                                          # 变量 a，值为 10
b = 6                                           # 变量 b，值为 6
print(6)                                        # 输出数字 6
print(a*b)                                      # 输出变量 a*b 的结果 60
print(a if a>b else b)                          # 输出条件表达式的结果 10
print("做对的事情比把事情做对重要")              # 输出字符串“做对的事情比把事情做对重要”
```

说明

在 Python 中，默认情况下，一条 print()语句输出后会自动换行，如果想要一次输出多个内容，而且不换行，可以将要输出的内容使用英文的逗号分隔。例如，以下代码将在一行中输出变量 a 和 b 的值。

```
print(a,b)                                      # 输出变量 a 和 b，结果为 10 6
```

在输出时，也可以把结果输出到指定文件中。例如，将一个字符串“命运给予我们的不是失望之酒，而是机会之杯。”输出到 D:\mot.txt 中，代码如下：

```
01  fp = open(r'D:\mot.txt','a+')                                  # 打开文件
02  print("命运给予我们的不是失望之酒，而是机会之杯。",file=fp)      # 输出到文件中
03  fp.close()                                                     # 关闭文件
```

说明

在上述代码中应用了打开和关闭文件等文件操作的内容，关于这部分内容的详细介绍请参见本书第 13 章，这里了解即可。

执行上述代码后，将在 D:\目录下生成一个名称为 mot.txt 的文件，该文件的内容为文字，即“命运给予我们的不是失望之酒，而是机会之杯。”，如图 2.28 所示。

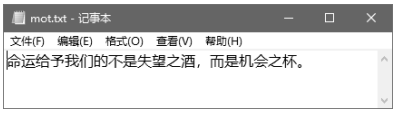


图 2.28　文件 mot.txt 中的内容

2.5　实践与练习

（答案位置：资源包\TM\sl\02\实践与练习\）

综合练习 1：程序员计算器　作为程序员，经常与二进制数、十进制数、八进制数和十六进制数打交道，例如将十进制数分别转换为对应的二进制数、八进制数和十六进制数。本任务要求编写 Python 代码，实现将输入的十进制数分别转换为对应的二进制数、八进制数和十六进制数。（提示：可以使用 bin()、oct()和 hex()函数实现）

综合练习 2：给电影打分　《肖申克的救赎》是一部经典的影片，在国内外评价均很高。编写一个程序，对该部电影进行评价。评分只能输入数字 1～9，输出根据用户打分形成的星级（★）评价，打几分就输出几个星（★）。（提示：输出多个相同字符时，可以使用*号，如想要输出 3 个 A，可以使用 print('A'*3)）参考输出结果如下：

请您为一部名为《肖申克的救赎》的电影打分（只能输入数字 1～9）：5

您为《肖申克的救赎》电影的评价是 ★★★★★