模块 3

Python 输入与输出

学习前请思考

(1)在 Python 中,如何实现基本的输入与输出?
(2)在 Python 中,如何写代码的注释?
(3)在 Python 中,如何进行代码的缩进?
(4)在 Python 中,编码规范和命名规范有哪些?

3.1 基本的输入与输出

Python 实现输入与输出有很多方式,基本的输入与输出是使用 print()函数和 input()函数来实现的。

3.1.1 使用 print()函数

输出的最简单方法是使用 print 语句,可以用逗号分隔零个或多个表达式,如下面的代码所示:

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

上述语句将会在屏幕上输出如下结果:

```
Python is really a great language, isn't it?
```

如果希望更多地控制输出格式,而不是简单地以空格分割,可以采用用户自己控制的方式来输出,如使用字符串切片、连接操作,以及字符串包含的一些有用的操作来实现,如下面的代码所示:

```
>>>for x in range(1, 11):
        print(str(x).rjust(2), str(x * x).rjust(3), end='')
        print(str(x * x * x).rjust(4))
```

输入完成后，按两次 Enter 键，输出的结果如下：

```
1  1   1
2  4   8
3  9  27
4 16  64
5 25 125
6 36 216
7 49 343
8 64 512
9 81 729
10 100 1000
```

在上述代码中，字符串对象的 str. rjust()方法的作用是使字符串靠右，并默认在左边填充空格，所占长度由参数指定，类似的方法还有 str. ljust()和 str. center()。这些方法并不会输出任何东西，它们仅仅返回新格式的字符串，新格式字符串再由 print 输出。

3.1.2 使用 input()函数

Python 2 中有两个内置的函数可从标准输入(默认来自键盘)读取数据，这两个函数分别是 input()和 raw_input()。但在 Python 3 中，raw_input()函数已被弃用。input()函数的返回值是字符串。

例如：

```
x=input("请输入 x=")
请输入 x=111

y=input("请输入 y=")
请输入 y=222

z=x+y
print("x+y=",z)
```

程序的运行结果如下：

```
x+y=111222
```

可以看到 input 的返回值永远是字符串，当需要返回 int 型时，需要使用 int(input())的形式。例如：

```
x=int(input("请输入 x="))
请输入 x=111

y=int(input("请输入 y="))
请输入 y=222

z=x+y
print("x+y=",z)
```

程序的运行结果如下：

```
x+y= 333
```

3.2　注释

Python 中有多种注释，包括单行注释、多行注释和批量注释，中文注释也是常用的。Python 中的注释也有自己的规范。注释可以起到一个备注的作用，当进行团队合作时，某个人编写的代码经常会被多人调用，为了让别人更容易理解代码的用途，使用注释是非常有必要的。

3.2.1　单行注释

#被用作单行注释符号。在代码中使用#时，它右边的任何数据都会被忽略而被当作注释。例如：

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
#定义一个函数，用来输出“Hello,Mary”字符串。
def sayHello():
    print('Hello','Mary',sep=',',end='\n',flush=True)
```

3.2.2　多行注释

多行注释使用三个单引号，形式是 '''内容'''；也可以使用三个双引号，形式是"""内容"""。例如：

```
'''
输出 Hello,Mary\t

'''
def sayHello():
    print('Hello','Mary',sep=',',end='\t',flush=True)
```

3.2.3　中文编码声明注释

在编写 Python 代码时，难免会出现或是用到中文，这时需要在文件开头加上中文编码

声明注释。如果开头不声明保存编码的格式，那么系统会默认使用 ASCII 码保存文件，这时如果代码中有中文就会出错，即使中文是包含在注释里面的。因此，加上中文编码声明注释很重要。例如：

```
#coding=utf-8 或 #coding=gbk
```

以上两种形式都可以代表中文注释，更多人使用 utf-8。

3.3 代码缩进

在 Python 中，对逻辑行行首的空白是有规定的，逻辑行行首的空白不对，就会导致程序执行出错。这是 Python 语言与其他语言的一个很重要的不同点。

缩进的空白是有要求的，下面是一些缩进的方法：

(1)一般情况下逻辑行首不应该出现空白。

(2)if 语句的缩进方法。

(3)while 循环的缩进方法。

具体的缩进方法如下所示：

```
#一般情况下,行首应该不留空白
import sys
#缩进的方法有两种,可以按空格,也可以按 Tab 键
#if 语句的缩进方法
a=7
if a>0:
    print "hello"   #前面的空格是按 Tab 键
#while 语句的缩进方法
a=0
while a<7:
    print a      #前面的空格是按 Tab 键
    a+=1      #前面的空格是按 Tab 键
```

3.4 规范

3.4.1 编码规范

1. 缩进

4 个空格的缩进(编辑器都可以完成此功能)，不要使用 Tab 键，更不能混合使用 Tab 键和空格键。

2. 换行

每行最大长度为 79，换行可以使用反斜杠，最好使用圆括号。换行点要在操作符的后边

按 Enter 键。

3. 空行

(1)类和 top-level 函数定义之间空两行。

(2)类中的方法定义之间空一行。

(3)函数内逻辑无关段落之间空一行。

(4)其他地方尽量不要再空行。

3.4.2　命名规范

总体原则,新编代码必须按下面命名风格进行,现有库的编码尽量保持原有风格。

(1)尽量单独使用小写字母 l、大写字母 O 等容易混淆的字母。

(2)模块命名尽量短小,使用全部小写的方式,可以使用下划线。

(3)包命名尽量短小,使用全部小写的方式,不可以使用下划线。

(4)类的命名使用 CapWords 的方式,模块内部使用的类采用_CapWords 的方式。

(5)异常命名使用 CapWords+Error 后缀的方式。

(6)全局变量尽量只在模块内有效,类似 C 语言中的 static。实现方法有两种,一是__all__机制,二是加一个下划线作为前缀。

(7)函数命名使用全部小写的方式,可以使用下划线。

(8)常量命名使用全部大写的方式,可以使用下划线。

(9)类的属性(方法和变量)命名使用全部小写的方式,可以使用下划线。

(10)类的属性有 3 种作用域 public、non-public 和 subclass API,可以理解成 C++中的 public、private 和 protected,non-public 属性前加一条下划线作为前缀。

(11)类的属性若与关键字名字冲突,加一条下划线作为后缀,尽量不要使用缩略等其他方式。

(12)为避免与子类属性命名冲突,在类的一些属性前加两条下划线作为后缀。例如,若类 Foo 中声明了__a,则只能通过 Foo._Foo__a 来进行访问,以避免歧义。

(13)类的方法第一个参数必须是 self,而静态方法第一个参数必须是 cls。

3.5　实训

3.5.1　输出唐诗《静夜思》

【实训描述】

本实训输出唐诗《静夜思》。

【实训过程】

```
#-*- coding: utf-8 -*-
print ('''静夜思
床前明月光,
```

```
疑是地上霜。
举头望明月，
低头思故乡。'")
```

程序的运行结果如图 3-1 所示。

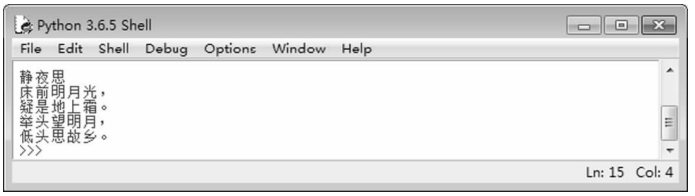

图 3-1 “输出唐诗《静夜思》”程序运行结果

3.5.2 输出大写字母的 ASCII 状态值

【实训描述】

本实训输出大写字母的 ASCII 状态值。

【实训过程】

```
#*_*coding:utf-8*_*
while True:
    c=input("请输入单个大写字母:")
    if len(c)>=2:
        print("字符长度超出范围,请输入单个字符!")
    else:
        print(c+"的 ASCII 码为",ord(c))
```

程序的运行结果如图 3-2 所示。

图 3-2 “输出大写字母的 ASCII 状态值”程序运行结果