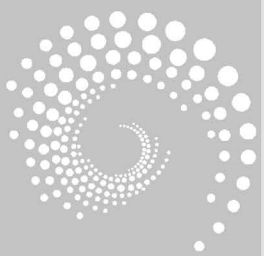
第3章

Python语句

CHAPTER 3



Python 语句是 Python 程序的基本组成单元，一个大型的 Python 程序都是由一个个 Python 语句构成的，所谓万丈高楼平地起，学好 Python 基本语句是熟练编写 Python 程序的前提和基础。与其他语言一样，Python 语言可以实现顺序、选择和循环的程序结构。

3.1　输出语句格式控制语句


视频讲解

例 3-1　基本输出语句格式举例(In[1])。

1. 基本输出语句

```
In[1]:
  print('hello world!')
  print(5)
```

程序运行结果如下。

```
hello world!
5
```

2. 使用%格式控制符对输出格式进行控制

%格式控制符是用于控制打印输出的格式的特殊字符,在 Python 中常用的%格式控制符如表 3-1 所示。

表 3-1　Python 中常用的%格式控制符

格 式 符 号	转　　换
%s	字符串
%d	有符号的十进制整数
%f	浮点数
%c	字符
%u	无符号十进制整数
%x	十六进制整数(输出小写十六进制)
%X	十六进制整数(输出大写十六进制)
%e	科学记数法(输出的字母部分为小写)
%E	科学记数法(输出的字母部分为大写)

现举一个例子说明 Python 中%格式控制符的用法。

例 3-2　%格式控制符举例(In[2])。

```
In[2]:
  name = 'hello world!'
  print('Her name is  %s' % name)
  t = 8.0754
  print('%5.2f' % t)
  name = '张红'
  age = 23
  print('我的名字是%s,我的年龄是%d' %(name,age))
```

程序运行结果如下。

```
Her name is hello world!
8.08
我的名字是张红,我的年龄是 23
```

3. 使用 format 方法对格式进行控制

例 3-3 format 格式控制符举例(In[3]～ In[7])。

(1) 通过关键字进行格式化。

```
In[3]:
  name = '张红'
  age = '23'
  print('姓名:{名字},年龄:{年龄}'.format(名字 = name,年龄 = age))
```

程序运行结果如下。

```
姓名:张红,年龄:23
```

(2) 通过位置进行格式化。

```
In[4]:
  print('姓名:{1},年龄:{0}'.format(age,name))
```

程序运行结果如下。

```
姓名:张红,年龄:23
```

```
In[5]:
  print('{:.2f}'.format(3.1415926))
```

程序运行结果如下。

```
3.14
```

```
In[6]:
  #针对数字,使用千分位分隔符
  print('{:,}'.format(25000.01245))
```

程序运行结果如下。

```
25,000.01245
```

(3) 转义字符的使用。

对于某些字符(串)进行特殊处理,需要用到转义字符。在常用的特殊转义字符中,/n 代表换行; /t 表示制表符,其作用相当于一个 Tab 键(4 个空格键)。

```
In[7]:
  name = '张红'
  age = '23'
  sex = '男'
  stu_no = '20222305008'
  #print('{姓名}'.format(名字 = '张红'))
  print('学号:{0}\n 姓名:{1}\n 性别:{2}\n 年龄:{3}'.format(stu_no, name,sex,age))
```

程序运行结果如下。

```
学号:20222305008
姓名:张红
性别:男
年龄:23
```

4. 使用 f 方法对输出格式进行控制

f 格式化是 Python 3.6 之后版本才有的功能。通过在字符串前添加 f 或者 F 进行格式化处理,功能相当于%或.format()。

例 3-4　f 方法格式控制符举例(In[8])。

```
In[8]:
  name = '肖伦'
  age = '23'
  sex = '男'
  stu_no = '20222305017'
  print(f'学号:{stu_no}\n 姓名:{name}\n 性别:{sex}\n 年龄:{age}')
```

程序运行结果如下。

```
学号:20222305017
姓名:肖伦
性别:男
年龄:23
```

3.2　选择语句

选择语句主要是 if 语句,分为三个,分别是单项选择语句、双向选择语句和多项选择语句。

例 3-5　选择语句举例(In[9]～ In[12])。

(1) if 单项选择语句。满足选择条件则执行程序块,不满足则不执行。

```
In[9]:
  x = -1
  if x < 0:
    print('x 小于 0')
```

程序运行结果如下。

```
x 小于 0
```

(2) if 双向选择语句。满足选择条件则执行语句块 1,否则执行语句块 2。

格式:

```
if 条件 1:
        语句块 1
     else:
        语句块 2
In[10]:
  x = 5
  if x < 0:
    print('x 小于 0')
  else:
    print('x 不小于 0')
```

程序运行结果如下。

```
x 不小于 0
```

上述程序也可以修改为以下程序：

```
In[11]:
  x = -1
  print('x 小于 0') if x<0 else print('x 不小于 0')
```

程序运行结果如下。

```
x 小于 0
```

(3) if 多向双向选择语句。

格式：

```
if 条件 1:
  语句块 1
elif 条件 2:
  语句块 2
elif 条件 3:
  语句块 3
else:
  语句块 n
```

该语句的执行过程如下。

(1) 检查条件 1(if 语句)。

如果条件 1 为真,则执行语句块 1,然后跳过整个 if-elif-else 结构。

如果条件 1 为假,则继续检查下一个条件。

(2) 检查条件 2(elif 语句)。

如果条件 2 为真,则执行语句块 2,然后跳过整个 if-elif-else 结构。

如果条件 2 为假,则继续检查下一个条件。

(3) 继续检查后续条件(可选)。

如果前面的条件都为假,则继续检查后续的 elif 条件。

如果某个 elif 条件为真,则执行相应的代码块,然后跳过整个 if-elif-else 结构。

(4) 执行最后的 else 代码块(可选)。

如果前面的所有条件都为假,则执行 else 语句后的代码块。

整个过程是顺序进行的,一旦找到为真的条件,执行相应的代码块,并跳过其余的条件。如果没有任何条件为真,则执行最后的 else 代码块(如果存在的话),或者整个语句结束。

例 3-6 多条件选择语句举例(In[12])。

```
In[12]:
  score = int(input('请输入你的分数:'))
  if score<60:
    print('成绩不及格')
  elif 60<=score<70:
    print('成绩及格')
  elif 70<=score<80:
    print('成绩中等')
  elif 80<=score<90:
    print('成绩良好')
  elif(90<=score<=100):
```

```
    print('成绩优秀')
  else:
    print('输入有误')
```

运行程序后，提示输入成绩。

```
请输入你的分数:80
```

再次运行程序，输出结果如下。

```
成绩良好
```

3.3　循环语句

循环语句的作用是多次执行同一段程序。

例 3-7　循环语句举例(In[13]～In[15])。

```
In[13]:
  fruits = ['banana','apple','mangguo']
  for fruit in fruits:
    print(fruit)
```

程序运行结果如下。

```
banana
apple
mangguo
```

```
In[14]:
  for i in range(1,10,2):
    print(i)
```

以上程序的执行流程如下。

(1) 给 i 赋初值 1。

(2) 判断 i＜10 是否成立，如果成立则执行循环体，转(3)。如果不成立，则退出循环，转(4)。

(3) 执行 i＝i＋2(其中 2 为步长)，判断 i＜10 是否成立，如果成立，则执行循环体后转(3)继续执行。如果不成立，则退出循环，转(4)。

(4) 执行后续语句。

程序运行结果如下。

```
1
3
5
7
9
```

```
In[15]:
  totle = 0
  for i in range(1,101):
    totle = totle + i
  print(totle)
```

程序运行结果如下。

```
5050
```

以上程序的功能是使用 for 循环求解 1～100 之和。

例 3-8 循环遍历字典举例(In[16])。

```
In[16]:
  d = {'name':'Divid','age':65,'gender':'male','department':'math'}
  for key in d:
    print(key,':',d[key])
```

程序运行结果如下。

```
name : Divid
age : 65
gender : male
department : math
```

以上程序的功能是遍历字典后输出字典的键值对。

3.4 while 语句

while 语句的作用是当条件成立,则执行循环体,否则退出循环体继续执行循环体后的语句。

例 3-9 while 语句举例(In[17])。

```
In[17]:
  totle = 0
  x = 1
  while x < 101:
    totle + = x
    x + = 1
  print(totle)
```

以上程序使用 while 语句实现整数 1～100 的求和运算,程序运行结果如下。

```
5050
```

3.5 break 语句

break 语句的作用是跳出 break 语句所在位置的最内层循环。

例 3-10 break 语句举例(In[18])。

```
In[18]:
  s = 0
  for i in range(1,101):
    s + = i
    if i > = 10:
      break
  print(s)
```

在上述程序段中，当 i>＝10 时，就退出 for 循环，整个程序实现 1～10 的相加。

程序运行结果如下。

```
55
```

3.6　pass 语句

pass 语句的作用是跳过该语句块，而这个语句块是应该有语句的，可能在编写程序时起到预留位置的作用。

例 3-11　pass 语句举例(In[19])。

```
In[19]:
  count = 0
  while count < 10:
    count = count + 1
    if count % 3 = = 0:
      pass
    else:
      print(count)
```

程序运行结果如下。

```
1
2
4
5
7
8
10
```

3.7　continue 语句

continue 语句的作用是跳过和 continue 同一级别的后续语句。

例 3-12　continue 语句举例(In[20])。

```
In[20]:
  count = 0
  while count < 10:
    count = count + 1
    if count % 3 == 0:
      continue
      print(count)
    else:
      print(count)
```

程序运行结果如下。

```
1
2
4
```

```
5
7
8
10
```

在上述程序段中,一旦 count 能被 3 整除,则执行 continue 语句并跳过其后的 print(count)语句;否则,执行 else 语句后面的 print(count)语句,输出 11 之内不能被 3 整除的整数。

3.8 二元运算符和比较运算符

二元运算符用于进行算术运算,比较运算符用于判断两个 Python 对象的大小。Python 中常用的二元运算符和比较运算符如表 3-2 所示。

表 3-2 Python 中常用的二元运算符和比较运算符

二元操作符	描 述
a+b	a 与 b 之和
a-b	a 与 b 之差
a * b	a 与 b 之积
a/b	a 除以 b
a//b	a 整除 b
a&b	a 与 b 按位相与
a|b	a 与 b 按位相或
a^b	a 与 b 按位相异或
a==b	a 与 b 相等则返回 True,否则返回 False
a!=b	a 与 b 相等则返回 False,否则返回 True
a<=b,a<b	a 小于或等于 b,a 小于 b
a>=b,a>b	a 大于或等于 b,a 大于 b
a is b	a 与 b 是同一个 Python 对象,则返回 True
a is not b	a 与 b 不是同一个 Python 对象,则返回 True

习题 3

本书提供在线测试习题,扫描下面的二维码,可以获取本章习题。

在线测试