第5章

Python编程基础

Python 既支持面向过程的编程，也支持面向对象的编程。Python 是完全面向对象的编程语言，程序中的数字、字符串、函数、模块都是对象，并且 Python 具备封装、继承、多态和面向对象特征。本章将介绍 Pyt 任务一：掌握 hon 编程的相关基础知识，主要内容包括：

任务一：Python 的内置数据类型。

任务二：Python 函数。

任务三：运算符与表达式。

任务四：控制语句。

本章教学目的

理解 Python 内置数据类型、内置函数、运算符、表达式及控制语句的基本语法，掌握其应用方法，为后续深入学习夯实基础。

5.1 Python 的内置数据类型

Python 语言强调一切皆对象，Python 中的一切数据都是对象，每个对象都属于某种数据类型，Python 中数据类型等同于类。内置数据类型是内置于 Python 解释器中的标准类型，用户可以直接使用。一般把内置数据类型分为基本数据类型和复杂数据类型。除了内置数据类型外，还有在标准库及第三方库中定义的数据类型，用户也可以自定义数据类型。

5.1.1 基本数据类型

1. 基本数据类型

基本数据类型包括整数（int）、浮点数（float）、复数（complex）、布尔型（bool）、NoneType 等。

虽然同样都是数值，但在 Python 语言里，整数、浮点数和复数是三种不同的数字类型。

（1）整数是不带小数点的数据，如 5、10。

（2）浮点数是使用 C 中的双精度实现的带小数点的实数，如 1.3 或 2.4E-6（科学记数法 $2.4 * 10^{-6}$ 在 Python 里的表示方式），其小数点后的有效位通常是 15 位。

（3）复数由实部和虚部组成，实部和虚部都是浮点数，如 2.3＋4.5j 就是一个复数（虚部

加小写字母 j 来表示）。

sys.floatinfo 中提供了有关运行程序机器的浮点数字的精度和内部表示形式的信息。要从复数 z 中提取实部和虚部，可以使用 z.real 和 z.imag 来分别获取。

（4）布尔型存储的是逻辑值，只有 True 和 False 这两个取值，分别对应逻辑真值和逻辑假值，当布尔型参与算术运算或比较运算时，True 被看作整数 1，False 被看作整数 0。

（5）类型 NoneType 中的唯一值就是 None，它通常用于表示缺少值，当函数没有使用 return 返回结果时，调用函数得到的返回值就是 None。此外，还可以指定函数的形式参数的默认值为 None。

2. 混合数据类型算术运算规则

Python 完全支持混合数据类型的算术运算，其规则如下。

当一个算术运算符要对两个具有不同数据类型的操作数做运算时，“更窄”类型的操作数会自动加宽成另一个“更宽”的数据类型，并由此得到“更宽”的数据类型的结果。其中整数比浮点数窄，浮点数比复数窄，所以，2＋3.0 得到的结果是 5.0，5.0 是浮点数。

Python 解释器中有许多内置函数，可直接调用，内置函数 type(a)的作用就是输出对象 a 的类型信息。

在 Python Shell 上执行以下代码，观察输出，了解基本数据类型的使用和变化情况。

说明：代码中以“＃”开始直至行末的信息为注释，是对程序中代码的解释说明。注释的作用是方便阅读理解程序和读懂程序，对程序的执行没有任何影响。

```
>>>a=1                #为对象 a 赋值
>>>type(a)            #查看对象 a 的数据类型
<class 'int'>
>>>b=2.3               #为对象 b 赋值
>>>type(b)            #查看对象 b 的类型
<class 'float'>
>>>a+b                #查看 a+b 的结果
3.3
>>>c=4+5j             #为对象 c 赋值
>>>type(c)            #查看对象 c 的类型
<class 'complex'>
>>>a+b+c              #查看 a+b+c 的结果
(7.3+5j)
>>>d=True             #为对象 d 赋值
>>>type(d)            #查看对象 d 的类型
<class 'bool'>
>>>a+b+c+d            #查看 a+b+c+d 的结果
(8.3+5j)
>>>e=None             #为对象 e 赋值
>>>type(e)            #查看对象 e 的类型
<class 'NoneType'>
```

5.1.2　复杂数据类型

复杂数据类型包括字符串(str)、元组(tuple)、列表(list)、集合(set)、字典(dict)等。

1. 字符串

字符串是由 0 个或多个 Unicode 字符组成的不可变的序列。

字符串可以采用多种方式表示。

(1) 由成对的单引号' '引起来的文本序列,其中允许嵌入双引号且不必转义;

(2) 由成对的双引号" "引起来的文本序列,其中允许嵌入单引号且不必转义;

(3) 由成对的三个单引号或三个双引号引起来的文本序列,其中允许嵌入单引号和双引号且不必转义,允许文本跨越多行。

在 Python Shell 上执行以下代码,通过内置函数 print(a)来观察输出,了解字符串类型的基本使用。

内置函数 print(a) 的作用就是输出对象 a 的值。

```
>>>str1='"Come on, Benny." Henry said.'              #用成对的单引号表示字符串
>>>print(str1)                                      #输出 str1 的值
"Come on, Benny." Henry said.
>>>str2="Now we'd better go back to house and pack." #用成对的双引号表示字符串
>>>print(str2)                                      #输出 str2 的值
Now we'd better go back to house and pack.
>>>#用成对的 3 个单引号表示字符串
>>>str3='''
"Watch is a lazybones," Jessie said fondly.
"He does anything but watch."
"That's his job, but he loved our boxcar from the first,"
Violet said, smiling down at him.
'''
>>>print(str3)                                      #输出 str3 的值
"Watch is a lazybones," Jessie said fondly.
"He does anything but watch."
"That's his job, but he loved our boxcar from the first,"
Violet said, smiling down at him.
>>>
```

2. 元组

元组是不可变序列,一般以“(”开头,以“)”结束(也可以不要这一对小括号),可用于存储多个元素。元素值的数据类型可以不同,元素之间以“,”间隔,元素值不能更改。

元组可以通过以下多种方式构建。

(1) 使用一对括号表示空元组,如 ( );

(2) 只包含一个元素的元组使用逗号结尾,如 a,或 (a,);

(3) 用逗号间隔元素,如 a,b,c 或 (a,b,c);

（4）使用内置函数 tuple( )。

在 Python Shell 上执行以下代码，观察输出，了解元组类型的基本使用。

```
>>>t1=()                          #将空元组赋值给 t1
>>>type(t1)                       #查看对象 t1 的类型
<class 'tuple'>
>>>t2=1,                          #将单个元素构成的元组赋值给 t2
>>>print(t2)                      #输出元组 t2 的值
(1,)
>>>type(t2)                       #查看对象 t2 的类型
<class 'tuple'>
>>>t3=(1,2.3,True,None)           #将多个不同数据类型的元素构成的元组赋值给 t3
>>>type(t3)                       #查看对象 t3 的类型
<class 'tuple'>
>>>t4=tuple()                     #利用内置方法 tuple( )构建元组
>>>type(t4)                       #查看对象 t4 的类型
<class 'tuple'>
```

3. 列表

列表也是由多个元素构成的序列，[1,2,3]表示一个列表，元素之间以“,”间隔，元素值的数据类型也可以不同，列表和元组最大的区别就是，列表可以变更，不仅列表中的元素可以改变，列表的长度也可以改变。

列表可以通过下面几种方式构建。

（1）使用一对中括号表示空列表，如[ ]；

（2）使用中括号，[ ]中元素用逗号间隔，如[a,b,c]；

（3）使用内置函数 list( )。

在 Python Shell 上执行以下代码，观察输出，了解列表类型的基本使用。

```
>>>list1=[]                       #将空列表赋值给 list1
>>>type(list1)                    #查看对象 list1 的类型
<class 'list'>
>>>list2=[1,2,False,3.4]          #将多个不同类型元素构成的列表赋值给 list2
>>>print(list2)                   #输出对象 list2 的值
[1, 2, False, 3.4]
>>>type(list2)                    #查看对象 list2 的类型
<class 'list'>
>>>list3=list()
>>>type(list3)                    #查看对象 list3 的类型
<class 'list'>
```

4. 集合

集合也是由多个元素构成的序列，以“{”开头，以“}”结束，元素之间以“,”间隔，元素值的数据类型也可以不同。

集合具有以下特点。

(1) 类似数学上的集合概念,集合内的元素不可以重复;

(2) 集合内的元素无先后顺序;

(3) 不能改变集合内原有元素的值,但是可以通过集合对象的 add( )、discard( )、remove( )、pop( )等方法增减元素,并由此改变集合的长度(即集合中元素的个数);

(4) 集合内的元素不能是列表、集合或字典等可变更对象;

(5) 集合可以用{a,b,c}表示;

(6) 创建空集合时,只能使用 set( ),不能用{ }创建空集合。

在 Python Shell 上执行以下代码,观察输出,了解集合类型的基本使用。

```
>>>set1={1,1,2,2}                    #将有重复元素的集合赋值给 set1
>>>print(set1)                        #输出 set1 的值
{1, 2}                                #集合内只保留了不重复的元素
>>>set2={1,2.3,False,True,4,None}     #对 set2 赋值
>>>print(set2)                        #输出 set2,观察其元素的无序性
{False, 1, 2.3, 4, None}              #由于 True 和 1 的哈希值都为 1,故被视为相同元素
>>>set3={1,[2,3]}                    #将列表对象作为集合的元素,会导致出错
Traceback (most recent call last):
  File "<pyShell# 80>", line 1, in <module>
    set3={1,[2,3]}
TypeError: unhashable type: 'list'
>>>set4=set()                        #用 set()创建空集合
>>>type(set4)                        #查看 set4 的类型
<class 'set'>
>>>print(set4)                       #输出 set4 的值,空集合输出为 set(),区别于空元组的输出
set()
>>>set5={ }                          #用{ }对 set5 赋值
>>>type(set5)                        #查看 set5 的类型,是字典类型,不是集合类型
<class 'dict'>
```

5. 字典

字典是一种映射类型,字典以“{”开头,以“}”结束,元素之间以“,”间隔,其中每一个元素都是一个“key:value”形式的“键值对”,即每一个元素由两个对象“key(键)”和“value(值)”组成,形成“键→值”的映射关系,类似数学中的函数。

字典具有以下特点。

(1) 字典元素的“键值对”是将“键”对象的哈希值映射到“值”对象,故字典中元素的“键”不能重复。

(2) 字典中元素的“键”不可改变。数字、字符串可以作为元素的“键”;但那些值可以改变的对象,如列表、字典等,不能用作元素的“键”。

(3) 字典中元素的“值”可以是任意类型的对象。

(4) 字典是可以变更的,可以通过元素的“键”,修改元素的“值”,也可以通过字典对象的 setdefault( )、pop( )等方法向字典中添加、删除元素。

(5) 创建字典对象时，可以使用内置函数 dict( )，可以用{ }创建空字典。

在 Python Shell 上执行以下代码，观察输出，了解字典类型的基本使用。

```
>>>dict1={'name':'张飞','gender':'男','age':18}     #将字典对象赋值给 dict1
>>>type(dict1)                                      #输出对象 dict1 的类型
<class 'dict'>
>>>print(dict1)                                     #输出 dict1 的值
{'name': '张飞', 'gender': '男', 'age': 18}
>>>dict2={1:('张丽','张芳'),'t2':['王刚','王伟']}
                                                   #字典中元素的键、值可以是不同类型的
>>>print(dict2['t2'])                  #利用字典对象名[key] 可以得到所映射的 value
['王刚', '王伟']
>>>dict3={ }                                        #将空字典赋值给 dict3
>>>type(dict3)                                      #查看 dict3 的类型
<class 'dict'>
>>>print(dict3)                                     #输出 dict3 的值
{}
```

5.1.3 复杂数据类型的基本操作

5.1.2 节介绍的字符串(str)、元组(tuple)和列表(list)类型都是序列数据类型，即上述类型对象中的元素是有顺序的，可以通过索引(或下标)访问其中的一个或多个元素。

注意：

(1) 索引值用 0 表示该对象中的第一个元素，字符串内用 0 表示第一个字符。

(2) 集合(set)和字典(dict)对象中的元素是无序的，因此不可以通过索引来访问。

对序列数据类型，可以进行切片操作，其功能是取出被操作对象(字符串、元组、列表等)的一部分。

语法格式为：

```
objectname[startindex : endindex : step]
```

其中，startindex 是取元素的起始索引。endindex 是结束索引(不包含该值)。step 称为步长，即索引值每次变化的量，若 step＞0，从左到右取元素，进行正向切片；若 step＜0，则从右到左取元素，进行反向切片。startindex、endindex 和 step 都是可选参数，若都不设置，则取出整个序列。

字典对象可以通过“键”访问其对应的“值”，也可以通过该对象的其他若干方法获得其元素值。

每种数据类型都提供了多种方法实现对该类型对象的访问或修改，具体的方法名及使用方法请参考帮助文档。下面代码中所使用的方法都有注释说明其作用。

在 Python Shell 上执行以下代码，观察输出，了解复杂数据类型的基本操作。

```
>>>str1='abcdefg'       #对字符串对象 str1 赋值
>>>print(str1[0])       #输出 str1 中第一个字符
```

```
a
>>>print(str1[-1])        #输出 str1 中最后一个字符
g
>>>print(str1[0:5:2])     #输出 str1 中索引[0,5)范围内索引值为 0,2,4 上的字符
ace
>>>print(str1[0:5])       #输出 str1 中索引[0,5)范围内的所有字符
abcde
>>>print(str1[::])        #输出 str1 中的所有字符
abcdefg
>>>print(str1 [::-1])     #逆序输出 str1 中的所有字符
gfedcba
>>>str1.index('d')        #返回 str1 中字符'd'的索引
3
>>>str1.capitalize()      #返回 str1 中首字母大写后的字符串
'Abcdefg'
>>>str1.upper()           #将 str1 中的所有字母大写后返回
'ABCDEFG'
>>>str2='ab,c,d,e'
>>>str2.split(',')        #将 str2 中的元素以“,”作为分隔符进行拆分,得到列表并返回
['ab', 'c', 'd', 'e']

t1=('赵亮','孙周',18,19,True,False,None)   #对元组对象 t1 赋值
>>>print(t1[1])           #输出 t1 中的第二个元素
孙周
>>>print(t1[2:5])         #对 t1 切片并输出
(18, 19, True)
>>>t1[0]='着凉'           #试图更改元组中的元素,失败
Traceback (most recent call last):
  File "<pyShell# 147>", line 1, in <module>
    t1[0]='着凉'
TypeError: 'tuple' object does not support item assignment
>>>print(t1.count 18)     #输出 t1 中值为 18 的元素个数
1

list1=['赵亮','孙周',18,19,True,False,None]   #对列表对象 list1 赋值
>>>print(list1[1:3])      #对 list1 切片
['孙周',18]
>>>list1[0]='着凉'        #更改列表中的元素
>>>print(list1)           #输出 list1
['着凉','孙周',18,19,True,False,None]
>>>list1.append(3.7)      #向 list1 中追加元素
>>>print(list1)
['着凉', '孙周',18,19,True,False,None, 3.7]
```

```
>>>set1={1,2,3,4}              #对集合对象 set1 赋值
>>>set2=set([3,4,5])           #用 set 方法创建 set2 对象
>>>set1.issubset(set2)         #判断 set1 是否是 set2 的子集
False
>>>set2.add(6)                 #向 set2 中添加元素 6
>>>set1.remove1                #从 set1 中移除元素 1
>>>print(set1)                 #输出对象 set1 的值
{2, 3, 4}
>>>print(set2)                 #输出对象 set2 的值
{3, 4, 5, 6}
>>>set1-set2                   #集合的差集运算
{2}
>>>set1.update(set2)           #向 set1 中添加 set2 中的所有元素
>>>print(set1)                 #输出对象 set1 的值
{2, 3, 4, 5, 6}

>>>stu1={"name":"张雪","gender":"女","age":20}                    #对字典对象 stu1 赋值
>>>stu1.get("name")            #获取 stu1 中“键”name 所对应的值
'张雪'
>>>stu1.setdefault("major","软件工程")                            #向 stu1 中增加“键值对”
'软件工程'
>>>stu1.keys()                 #获取 stu1 中所有的“键”
dict_keys(['name', 'gender', 'age', 'major'])
>>>stu1.values()               #获取 stu1 中所有的“值”
dict_values(['张雪', '女', 20, '软件工程'])
>>>stu1.items()                #获取 stu1 中所有的元素(键值对)
dict_items([('name', '张雪'), ('gender', '女'), ('age', 20), ('major', '软件工程')])
```

每种数据类型都提供了多种方法对该类型对象进行各种操作，具体的方法名及使用方法可参考安装 Python 解释器时一起安装的帮助文档。

在 IDLE 的 Python Shell 上输入代码时，会出现如图 5.1 所示的自动代码提示。该提示中列出了整数对象 int1 可以使用的方法。

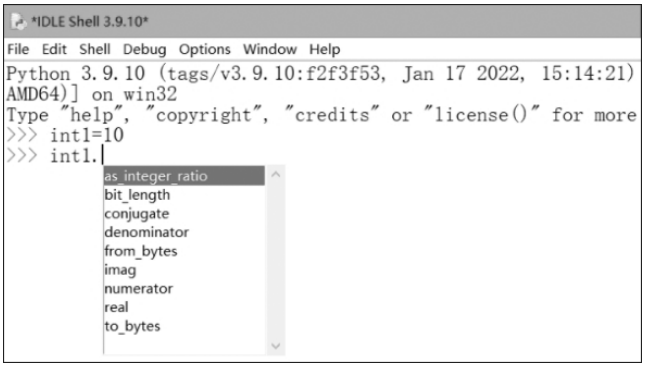

图 5.1　在 IDLE 的 Python Shell 上出现的自动代码提示

当选择了某个方法并输入左括号“(”时，窗口中会出现关于该方法的简要说明，如图 5.2 所示，图中说明 bit_length()方法会返回用二进制表示整数 int1 所需要的位数。

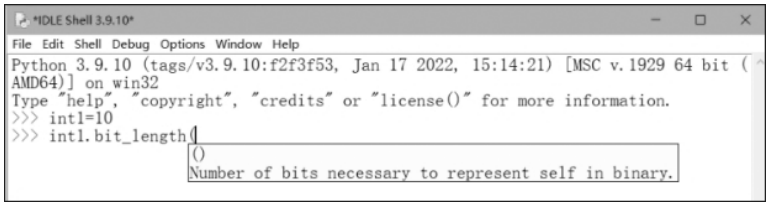

图 5.2　在 IDLE 的 Python Shell 上出现的简要方法说明

若需要查看该方法的详细说明，可以在该界面下按 F1 键打开帮助文档，如图 5.3 所示，单击菜单“Help”下菜单项“Python Docs F1”，最终该代码的运行结果如图 5.3 所示，该代码表示利用二进制来表示十进制的整数 10，需要 4 位二进制数来表示。

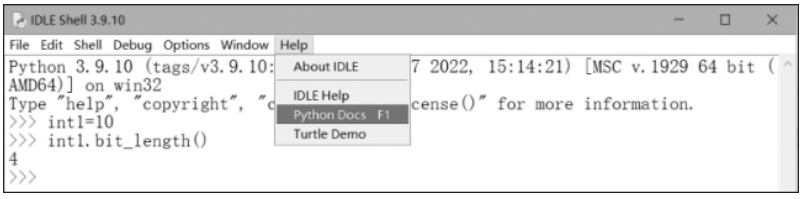

图 5.3　在 IDLE 的 Python Shell 上运行代码

图 5.4 为列表对象 list1 出现的自动代码提示，该提示列出了该对象可以使用的方法。当可用方法多于 10 个，而无法完全显示时，可以拖动提示窗口的滚动条查看完整内容。

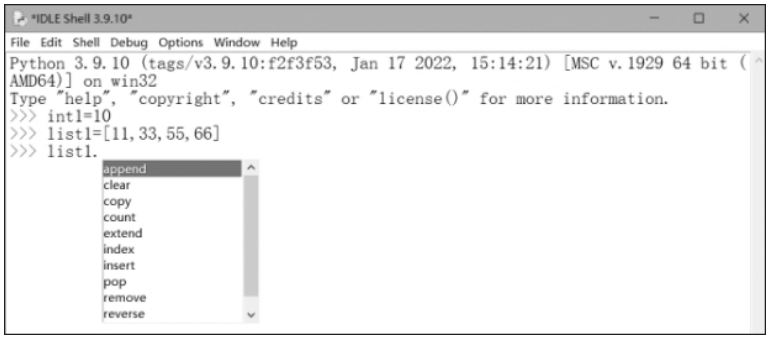

图 5.4　在 IDLE 的 Python Shell 上出现的列表对象的自动提示

针对其他数据类型的对象，请读者参照上述方法自行查看、学习。

5.2　Python 函数

5.2.1　Python 函数分类

Python 中的函数分为内置函数、库函数和自定义函数。

1. 内置函数

内置函数是由 Python 解释器提供的，在 Python 内置模块中定义好的，用户无须定义也无须导入就可以直接使用的函数，例如前面使用过的 print( )、type( )等函数。

Python 中的内置函数如图 5.5 所示。图中的内置函数按字母顺序纵向排列。其中很多函数可以“顾名思义”，如 help( )是用于提供帮助信息的函数，max( ) 、min( )是计算最大值、最小值的函数，sorted( )是排序函数，sum( )是求和函数。

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

图 5.5 Python 中的内置函数

Python 中有一些内置函数，也是相应类的构造方法，可用于创建该类型的对象，如 bool( )、complex( )、dict( )、float( )、int( )、list( )、object( )、range( )、set( ) 、slice( )、str( )、tuple( )、type( )等，分别对应布尔型、复数、字典、浮点数、整数、列表、对象、序列、集合、切片、字符串、元组、类型等不同的数据类型。

对于 Python 中的每一个内置函数的具体用法，读者可自行查看帮助文档详细了解。学习方法如图 5.6 所示。

(1) 根据箭头 1 所示先找到 Python 标准库(The Python Standard Library)；

(2) 选择箭头 2 所示的内置函数(Built-in Functions)，就能看到如图 5.5 所示的内置函数列表；

(3) 单击其中的任意一个函数，就会显示相应的详细说明。

图 5.6 中同时列出了内置函数列表中前面几个函数的帮助说明。

2. 库函数

Python 库函数分为标准库函数和第三方库函数。

1) 标准库函数

安装 Python 解释器时自带的库就是标准库(也叫作标准模块)，标准库中定义的函数就是标准库函数(也简称为标准函数)。标准函数无须定义，但需要使用 import 语句导入相应标准库后再使用。

在这里介绍几种常用的标准库：math、time、random、os。