**第2章****Python快速入门**

本章将针对Python的主要特性做一个简单讲解,帮助读者迅速了解Pythono如果在学习 本章内容过程中遇到困扰，不要担心。因为这些内容都会在后续章节进行单独的详细介绍，这里， 只是带大家快速入门。

程序的输趣和箍人

输入和输出是Python程序最基本的操作，在前面第1章的内容中，其实我们已经接触了 程序的输出，这里将具体讲解程序的输入和输出。

**2.1**

在Python中，提供了一个专门输出的函数print()9该函数的声明如下:

print (\*objects, sep = \* ! , end = ! \n1 r file = sys . stdout, flush = False) 可选参数关键字的相关介绍如下:

**-sep**：在值之间插入的字符串，默认是一个空格。

* **end**：追加在后面值的字符串，默认是一个换行符。

**° file**：类似文件(流)的对象,默认是sys.stdouto

* **flush**：是否强制刷新文件(流)，默认是FMs©。注flush参数是从Python 3.3版本 开始添加的。

**print()函数可以输出各种类型的数据，常见的输出场景如下：**

(1 )直接输出字符串和数值类型。示例如下：

>>> print(120)

120

>>> print (? abc')

abc

(2)输出变量，包括数值、布尔、列表、字典等变量都可以直接输出。

>>> print(a)

123

»> b = [1,2,3,4,5]

»> print (b)

[1, 2； 3, 4, 5]

1. 格式化输出，该输出效果类似于C语言中的printf()o示例如下:

>>> s = \* helloJ

>>> n = len(s)

>>> print(\* The length of %s is %d!%(sr n))

The length of hello is 5

1. 不换行输岀。默认情况下，使用p『inf()输出数据后默认会换行，例如，使用p『hit() 函数输出列表中的值，效果如下所示。

>>> for x in range(0 r 3):

... print(x)

0

1

2

如果希望在Python3中输出的数据不换行，可以将函数写成下列uprint(x, end =")”样式, 示例如下:

>>> for x in range(0 z 3):

print (xf end = \* ')

012

1. 一次输出多个值。使用print。可以一次输出多个数据，示例如下：

>>> print(5r 1r 'python \*)

5 1 python

***2.2***

现在，我们已经可以用p，iiit()输出想要的结果了。但是，如果要让用户从计算机输入一些 字符怎么办？ Python提供了一个input函数，该函数可以让用户输入字符串，并存放到一个 变量里。例如，输入用户的名字，示例如下所示。

>>> name = input ()

Billy

上述代码中,在命令行输入name = input()并按[Enter】键后,命令行就会等待用户输入 To这时，可以输入任意字符,然后按[Enter】键完成输入。输入完成后,不会有任何提示, Python交互式命令行又回到〉>>状态了。那我们刚才输入的内容到哪里去了？答案是存放到 umne变量里了。可以直接输入name查看变量内容，示例如下：

>>> neime

'Billy5

**2.3Pythom 注释**

在实际开发中，不管是个人开发还是团队合作开发，为了让别人更容易理解代码的功能，

使用注释是非常有效的方法。

**2.3.1緡淺繹德嬢涇籌**

Python注释是以符号#开头的。当我们在代码中使用#注释时，它右边的任何数据都会被 当作注释，这样程序运行的时候，Pythoii解释器会直接忽略注释。下面我们看一段示例代码歹 具体如下：

print ( 5 hello world! \*) # 使用 print 输出 hello world!

上述代码中,#号右边的内容在执行的时候是不会被输出的。

注释还有一个巧妙的用途,就是一些代码我们不想运行,但又不想删除,就可以用注释暂 时屏蔽掉。示例如下:

#暂时不想运行下面这段代码

* print(!hello world!J)
* print(?hello python!\*)

当为Pythg代码添加注释时，这里有两点建议：

(1 )最需要写注释的是代码中那些技巧性的部分。对于复杂的操作，应该在其操作开始 前写上若干行注释。对于不是一目了然的代码，应在其行尾添加注释。

(2)为了提高注释的可读性，行尾添加的注释应该至少离开代码2个空格。

**2么2黨灣掌籍窜**

Python有一种独一无二的注释方式，那就是使用文档字符串。文档字符串是包、模块、类 或者函数里的第一个语句，它使用三重双引号组织，其包裹的内容可以通过对象的\_doc\_成 员自动提取§并且被pydoc所用。下面看一下标准的p『int函数中的文档字符串，示例如下：

def print (self z \*argsf sep= 5 ' , end= ! \n ! f file=None):

H H H ,

print (value, *……* sep= ? \* end= 5 \n \ file=sys . stdout7 flush=False)

Prints the values to a stream, or to sys.stdout by default.

Optional keyword arguments:

file: a file-like object (stream) ; defaults to the current sys. stdout. sep: - string inserted between values, default a space.

end: string appended after the last value, default a newline a

flush: whether to forcibly flush the stream.

n n n

pass

大多数计算语言都会使用花括号来表示一个语句块，而Pythoii不同，Python没有使用花 括号或者其他定界符来表示一个语句块。

缩进是Pythg表示语句块的唯一途径，在Python中，对缩进的要求是非常严格的。一个 语句块中的所有语句必须使用相同的缩进，表示一个连续的逻辑行序列。源文件的第一行不需 要缩进(不允许以任何空格开始)。例如，下面的这段代码会报错。

if True:

pisint ( "Answer ° )

print ("True")

else :

print ("Answer")

print ( "False" ) #缩进不一致,会导致运行错误

以上程序会报错。这是因为在控制语句中，不仅要保证语句块内语句有一定的缩进，而且 要保证整个语句块在控制语句有相同的缩进，如此才能表示这些代码属于同一个控制语句的代 码块。

**2.3.2譴譬鬚邂圖牆**

标准Python风格中每个缩进级别使用4个空格，所以尽量不要使用Tab制表符！因为不 同编辑器处理制表符的方式不同，有些会将它视为4个空格，有些会将它视为4个以上的空格， 从而产生与代码缩进相关的错误。当然，可以在编辑器中设置Rb键显示的空格数量。

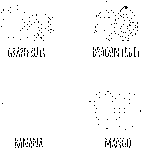
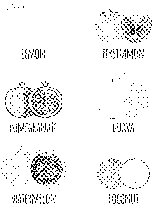
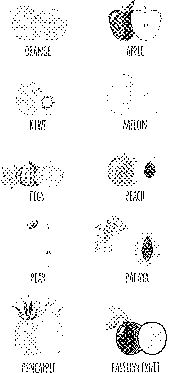
**2.3.3**

在Python中,如果希望在多行中编写一行较长的文本或代码,可以将所有的文本使用括 号包含起来,示例如下:

demo\_str= ("Python是一种面向对象、解释型计算机程序设计语言，"

"由Guido van Ros sum于198 9年底发明。第一个公开发行"

"版发行于1991年。°)



**2O4痂识綺和关链字**

现实生活中，人们常用一些名称来标记事物， 例如,现实生活中每种水果都有一个名称来标识, 具体如图24所示。

同理，若希望在程序中标识一些事物，需要 开发人员自定义一些符号和名称,这些符号和名 称叫作标识符。变量名、函数名都是标识符。

Python中的标识符由字母、数字和下画线 组成，其命名方式需要遵守一定的规则，下面对 这些规则进行讲解。

(1)标识符由字母、数字和下画线组成，且

臉飒对

不能以数字开头。示例如下:

图2-1现实生活中的标识符

fromNol2 #合法的标识符

from#12 #不合法的标识符,标识符不能包含#符号

2nd0bj #不合法的标识符,标识符不能以数字开头

（2） Python中的标识符是区分大小写的。例如，andy和Andy是不同的标识符。

（3） Python中的标识符不能使用关键字。例如，if不能作为标识符。

除此之外，为了规范标识符，关于标识符的命名有以下两点建议：

（1） 见名知意：起一个有意义的名字，尽量做到看一眼就可以知道标识符是什么意思， 从而提高代码的阅读性。比如，定义名字使用name来表示。

（2） 建议变量名使用“小写字母”加命名。例如，demojisto

**2.4.2襲纜享**

Python中一些具有特殊功能的标识符称为关键字。关键字是Python语言预先定义的,不 允许开发者使用的标识符。Python中的关键字如下所示。

|  |  |  |  |
| --- | --- | --- | --- |
| False | def | if | raise |
| None | del | import | return |
| True | elif | in | try |
| and | else | is | while |
| as | except | lambda | with |
| assert | finally | nonlocal | yield |
| break | for | not | class |
| from | or | continue | global |
| pass |  |  |  |
| Python中的每个关键字都代表不同的含义。读者可以在命令行输入help（）命令进入帮助系 | | | |
| 统，使用“help〉关键字”命令查看关键字的相关信息。 | | |  |
| >>> help () |  | #进入帮助系统 |  |
| help> keywords |  | #查看所有的关键字列表 |  |
| help> return |  | #查看return这个关键字的说明 | |
| help> quit |  | #退出帮助系统 |  |

**2.5变量**

**2.5J** '

在学习Python的过程中会用到许多数据,为了方便在接下来的程序中引用这些数据,需 要使用不同的名字分别表示这些数据，在以上叙述中，“数据”通常被称为变量的值，"数据 的名字”则称为变量名。例如，使用x表示数值10的写法如下：

x = 10

上述代码中，X.表示的是变量名，10表示的是变量的值。Pythoii作为一门动态类型语言， 不需要预先声明变量类型，变量的类型和值在赋值的同时确定，变量赋值是通过等号来执行的。

接下来，通过几个简单案例来演示变量赋值的几种方式。

(1 )变量nam©\_01的值是“小明”，代码如下：

name\_01 ='小明'

内存分配情况如图A2所示。

(2)变量namR\_01的值是“小明”，变量眼me\_02的值是“小红”，代码如下： name\_01 ="小明， '

name\_02 =早小红'

内存分配情况如图2・3所示。

(3 )变量name\_01的值是“小明”，变量iMme\_02的值等于name\_019代码如下:

name\_01 ='小明“

name\_02 = name\_01

内存分配情况如图2『4所示。

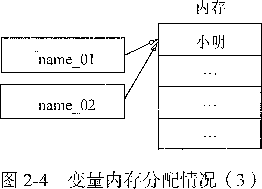
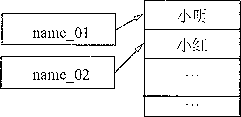
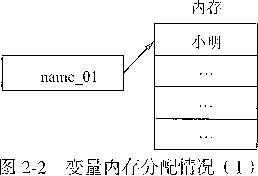


图2-3变量内存分配情况(2)

内存

‘ 除此之外，变量还支持增量赋值，即通过复合运算符进行赋值，看下面的例子：

n = n \* 10

将上面的例子改成增量赋值方式即

n \*= 10

需要注意的是，Python不支持C语言中的自增1(++)和自减1(一)运算符，这是因为+和- 也是单目运算符，Python会将—n解释为-(-n)从而得到n?同样++11的结果也是no

2亂2

希望一个变量可以存储新的数据，不需要删除原有变量就可以直接赋值•例如下面的代码？

»> a = 10

〉〉〉a = 3

»> a

3

从上述代码可以看出，再次对变量&进行赋值后，原有变量的值被改变了，这是因为，当 对变量务进行再次赋值时，它会指向内存中的一个新值，原先变量的值会被释放，最终被内存 回收。

变量a在内存中的分配情况如图2・5所示。

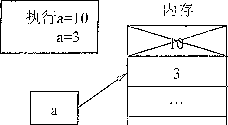
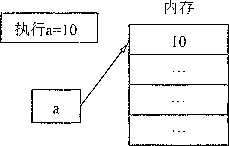


图2-5变量a的内存分配情况

**'2o6戮字奨型**

Python支持种3种数字类型，分别是整型（血）、浮点型（float）.复数类型（complex）*。* 下面是一些数字类型的例子。

t M （M）

1010 23 T2 0x84

2.浮点型（利。觥）

2.45 3.2E-10 -9.32 -2.4E-19

3n蘆數襲翼（complex）

5.1+2.3j 9+2j -2.3+5j

需要说明的是，在Python 2中，数字类型还有一种类型long,该类型表示的是长整型，它 所能表示的整数范围是远远超过C语言的长整型。事实上，Python中的长整型仅受限于计算机 的虚拟内存总数。但是，在Python 3中，long类型和血类型都被合并为inf类型了。

在第3章中，我们可以看到和数字类型相关的详细讲解。

*2.1*

布尔类型是特殊的整型，它的值只有两个，分别是True和False。如果将布尔值进行数值 运算，True会被当作整型1, False会被当作整型0。

在Python 3中,以下对象的布尔值都是Falseo

® Noneo

° False （布尔型）。

。0 （整型0）o

。0J （浮点型0 ） o

° 0J+0.0j （复数 0）o

®"（空字符串）。

° []（空列表）O

° （）（空元组）。

° {}（空字典）。

•用户自定义的类实例中实现的\_bool\_或者\_len\_()方法。

除了上述对象之外的所有其他对象的布尔值都为Trueo

，2W字简醺

Python中的字符串被定义为一个字符集合,它被引号所包含,引号可以是单引号、双引号 或者三引号(三个连续的单引号或者双引号)。字符串具有索引规则，第一个字符的索引是0, 第二个字符的索引是1,依此类推。

下面是字符串的示例代码M

str\_01 = 5 Python'

str\_02 = "Python"

str\_03 = \* , \* Python \* , 5

Python中的字符串有很多实用的操作方法，可以使用索引来对字符串进行查找和取值操作， 示例如下:

>>> demo\_str = , hello itheima9

>>> demo\_str[0]

>>> demo\_str[0:6]

Jhello 1

>>> demo\_str[6:]

5 itheima \*

>>> demo\_str[:6]

fhello 9

除此之外,还可以使用+或者\*对字符串进行连接和重复操作,示例如下:

>>> str\_\_01 = \* hello \*

>>> str\_02 = \* python9

>>> str\_01+str\_02

\* hellopython'

>>> str\_03 = str\_01\*3

>>> str\_\_03

? hellohellohello'

在第4章中，有字符串的详细讲解。

2O9死缱\*表

序列(sequence )是一组有顺序的元素的集合,它包含一个或者多个元素,也可以为空。

最常用的序列有两种类型,分别是元组(tuple)和列表(list),我们可以将列表和元组当 作普通的数据的结合，它们可以保存任意数量的任意类型的值，我们将这些值称作列表或者元 组的元素。不同的是，列表使用中括号口定义，元素的个数和值是可以随意修改的，而元组使 用小括号()定义,元素不可以被修改。下面是列表和元组的示例代码:

demo\_list = [ 1 f 2 ! hello 1 ] # 这是一个歹寸表

demo\_tuple = (1F 2 f 5 hello ! ) # 这是一个元组

34 w Pyth©o褻勰鑑蔭 映嚴寥Python

列表和元组都可以通过索引取值，这一点与字符串的使用方式相同。接下来，针对列表和 元组的操作进行简单介绍。

使用索引获取列表中的元素，并对列表元素进行修改，示例如下：

>>> demo\_list = [lf2f3f4z5,6]

>>> demo\_list

[lz 2, 3, 4, 5, 6]

>>> demo\_list[2]

3

>>> demo\_\_list [2 :] [3, 4, 5, 6]

>>> demo\_list[:5]

丄 2, 3, 4, 5]

»> demo\_list [-1]

6

>>> demo\_list[::2] 丄 3, 5]

>>> demo\_list[2] = 0

>>> demo\_list

[1, 2, 0, 4, 5/ 6]

使用索引获取元组中的元素，示例如下：

>>> demo\_tuple = (1Z2Z3Z4Z5^6)

»> demo\_tuple *[2]*

*3*

»> demo\_tuple [ : 5 ] (L 2’ 3, 4, 5)

»> demo\_tuple [ 3 ] = 2

Traceback (most recent call last):

File n<stdin>n r line 1z in <module>

TypeError: 1 tuple ? obj ect does not support item assignment

由以上示例可知须只能对元组进行读取操作，不能进行修改和删除操作。关于列表、元组 的相关知识，将会在第5章中详细讲解。

2d®字典

字典是Pythoii中的映射数据类型，由形如key:value的键值对组成。字典可以存储不同类 型的元素，元素使用花括号来包含。通常情况下，字典的键会以字符串或者数值的形式来表示, 而值可以是任意类型。示例如下：

demo\_dict = { "name" : ° zhangsan0 z "age" : 18 } # 这是一个字典

上述代码中，变量demo\_dict是一个字典类型，它存储了两个元素，第1个元素的键为 name,值为zhangsari；第2个元素的键为age,值为18。

2o 11 集念(set)

在Python中，集合是一个无序、不重复的数据组合，它的主要作用如下:

1. 去重。把一个列表变成集合，会自动去重。
2. 关系测试。测试两组数据之间是否存在交集、差集、并集等关系。 Python中可以使用花括号或者set()函数来创建集合。

注意：如果要创建空的集合，必须使用set(),而不是表示空值的“{}”。。用于创建空字典。 集合的常见操作如下所示。

student = {'Tom\*, \* Jim1, 5Mary', \* Torn5, 5 Jack!, 1 Rose 1} print (student) #输出集合,重复的元素被自动去掉

#成员测试

if(!Rose 1 in student) : print ( 1 Rose 在集合中?) else :

print (1 Rose不在集合中

')

# set可以进行集合运算 a = set(s abracadabra 5) b = set('alacazam1) print(a) print(a - print(a | print(a & print(a A

a和b的差集

a和b的并集

a和b的交集

a和b中不同时存在的元素

b)

b)

b)

b)

上述代码的输出结果如下所示。

{\* Rose'f 'Tom \*, 'Mary \*f 5 Jim1f 5 Jack \*}

Rose在集合中

{ 1 rs , f' \*ds }

{，r ' !d' }

( ?c\ 'd1 , " , 'a5 , !bS !z? }

{?a? }

{'丄?r S 'cT < ,z? }

a o if语甸

在Pythori中，if语句是一种条件判断语句，它的语法格式如下所示。

if条件表达式: 满足条件执行的语句

上述格式中，如果表达式的值为True，则下面满足条件的语句会被执行，否则会执行if语 句外的代码。

示例如下：

i = 1

x = 1

if i>0:

x = x+l

print(x)

上述代码执行后,程序运行到if的时候，因为判断条件成立，所以会执行X = x+1,此时,

X的值为2。priirt（x）位于if语句之夕卜，当执行完if语句后,print语句会执行，输岀x的值为2。 if^else语句也是一种if语句，其语法格式如下所示。

if条件表达式:

满足条件执行的语句

else:

不满足if条件表达式执行的语句

if-else语句的示例如下所示。

i = 1

if i < 0:

print （'id、于（P ）

else :

print （ vi 大于 0 5 ）

Python还支持elif （意为uelse-if ” ）语句,语法如下:

i f条件表达式1:

满足条件表达式1执行的语句

elif条件表达式2:

满足条件表达式2执行的语句

else :

都不满足执行的语句 elif语句的示例如下所示。

i = 10

if i > 0:

print（i + 1）

elif i == 0:

print（i \*10）

else :

print（i-1）

该示例中的if语句由三个分支，程序根据条件判断结果执行不同的分支，且只会执行三个 分支中的一个分支。

if语句可以嵌套使用，示例如下：

i = 10

if i > 1: print （ 1 i 大于 1'） if i > 5:

print （ ! i 大于 5 1）

上述代码中，“ifi>5:”下面的代码相对于该if缩进了 4个空格，表明其隶属于该if,而 不是外层的if o

wMte糖环

while循环的格式是: while循环条件:

循环执行的语句

上述格式中，只要循环条件成立，那么whi尼循环会不停地执行隶属于它的语句，直到循 环条件不成立为止。示例如下：

i = 1

while i < 5:

i = i+1

、 print(i)

上述代码的循环过程如下所示:

第1次，

第2次，

第3次，

第4次，

第5次，

i=l, i = 2, i = 3,

i = 4,

i = 5,

判断i <5, 判断i v5,

判断i <5, 判断i<5, 判断i < 5,

循环条件成立， 循环条件成立项 循环条件成立， 循环条件成立，

执行i = i+1, 执行i = i+1, 执行i = i+1, 执行i = i+1,

此时i的值为2； 此时i的值为3； 此时i的值为4； 此时i的值为5；

循环条件不成立，循环终止。

2ol4 for W

在Python中，for循环可以遍历任何序列，例如，列表、字符串等。版循环的语法格式如 下所示。

for 变量 in 序列： 循环中执行的语句

例如，使用for循环遍历列表，代码如下：

list\_demo = [1,2,3,4,5,6]

for i in list\_\_demo:

print(i)

上述代码中，fo『循环遍历的次数，也就是列表的长度，等于6。for循环下的p『int语句也 被执行6次。输岀结果如下所示。

1

2

3

4

5

6

考虑到要使用的数值范围可能会经常发生变化，Python W了一个内建函数range()9它返 回的是一个表示数值范围的对象(可迭代对象)。例如，range(5)表示的是0〜5 (不包括5 )的 整数，即 0, 1, 2, 3, 4。

看一段示例代码：

for i in range (5):

print (i)

输出结果：

38*虹* Pythoo氯離鷄蔭 黒霉攣Python

3

4

关于while循环、for循环、range()函数等详细用法，详见第6章。

2。15可选代幫象 •

从理论上来说，循环对象和fg循环调用之间还有一个中间层，该层将循环对象转换成可 迭代对象。这一转换通过使用iter()函数实现。但从逻辑层面上，常常可以忽略这一层，所以 循环对象和可迭代对象常常相互指代对方。

前面学过的列表、元组、字符串、集合等都是可迭代对象，所谓可迭代对象指的是可以返 回一个迭代器的对象，如果不清楚哪个是可迭代对象，可以通过Python内建的谕『()函数测试。

»> iter (丄2,3])

<list\_iterator object at 0xl018e66a0>

»> iter ( s 123 v )

<str\_iterator object at 0xl018e6748>

>>> iter({1,2,3})

<set\_iterator object at 0xl018e4798>

>>> iter(123)

Traceback (most recent call last):

File n<stdin>"r line 1r in <module>

TypeError: 'intJ obj ect is not iterable

上述代码中,iter()函数为列表返回了 list\_iterator对象,为字符串返回了 str\_iterator对象, 为集合set\_itemtow对象。但是，对于数字类型，当使用血此函数判断时，提示int类型的对象 不是可迭代的对象。

，曳滞读写

Python具有基本的文本文件读写功能。Pythoii的标准库提供了更丰富的读写功能。文本文 件的读写主要通过open()所构建的文件对象来实现。

1刨建黨緋对家

打开一个文件，并使用一个对象来表示该文件：

对象名=open (文件名,模式)

最常用的模式有：

e r :打开只读文件，该文件必须存在。

* r+ :打开可读写的文件，该文件必须存在。

-w :打开只写文件，若文件存在则文件长度清为零，即该文件内容会消失。若文件不存 在则建立该文件。

* w+：打开可读写文件，若文件存在则文件长度清为零，即该文件内容会消失。若文件 不存在则建立该文件。

-a :以追加的方式打开只写文件。若文件不存在，则会建立该文件歹如果文件存在，写 入的数据会被添加到文件末尾，文件原本的内容会被保留。

-a+ :以追加的方式打开可读写的文件。若文件不存在，则会建立该文件，如果文件存在, 写入的数据会被添加到文件尾后，文件原本的内容会被保留。

以上表示模式的字符可以与“b”组合成新的模式，如也、wb+、ab+等组合，加入b字符

用来告诉函数库以二进制方式打开文件，

而非使用纯文本方式打开。

2.文件对象的方法

读取数据：

#读取N bytes的数据

content = f o read(N)

content = f e readline() content = f » readlines()

#读取一行

#读取所有行，存储在列表中，每一行都是一个元素

写入数据旨

f o write(\* I like apple!\n \*)

#将？工like apple! ,写入文件并换行

#不要忘记关闭文件

关闭文件：

f.close ()

文件的详细知识将在第9章中详细讲解。

X17错误和异常

作为一名编程开发人员，难免会在开发过程中遇到一些问题。例如，程序报错或者抛出异常。 不过，好在编译器具有检查语法错误的强大功能，它可以快速检查到语法错误，并详细显示异 常信息，这样，开发人员就可以快速定位到出现问题的位置。

Python中发生的异常都对应一个类，这里列出一些常见的异常类，如表2』所示。

表2』常见异常类

|  |  |
| --- | --- |
| 异常类 | 相关描述 |
| NameError | 尝试访问一个没有定义的变量 |
| ZeroDivisionError |  |
| SyntaxError | 语法错误 |
| IndexError | 索引超出序列范围 |
| KeyError | 请求一个不存在的字典关键字 |
| lOError |  |
| ArributeError | 尝试访问未知的对象属性 |
| VkiueError | 传给函数的参数类型不正确 |

如果希望捕获这些异常，可以使用tryYxcept语句来实现，完整的捕获异常语法格式如下 所示。

try:

40 *間* Pythoo 舞饑疆鹳 MW Python

try代码块

except Exceptionlf Exception2 r .. .7 Argument:

exception代码块

…… #其他异常

else :

未捕获到异常的代码块

finally:

无论是否有异常,都会被执行的代码块

示例如下:

try:

1/0

except ZeroDivisionError:

print （" 不能为 0"）

else :

print （"其他情况"）

finally:

print （°到底啦〜“）

当然可以不必完全按照上述格式完全写下来，可以省略鱼lally语句，可以省略©庇语句， 也可以省略except语句,而保留finallyo

错误和异常的相关知识将在第12章中详细讲解。

:X18函數

在Python中,函数使用关键字def定义，它的后面需要紧跟一个函数名，函数名后面紧跟 的是函数所需要的参数（参数可选）。接下来，看一个Python中的函数，示例如下：

»> def add\_number （a^b）:

.。. return a+b

以上代码定义了一个函数，其中，add\_number是函数名，函数名后圆括号中的a、b是函 数的参数，圆括号后有一个冒号，这点与if和wh加循环的方式相同。在冒号下面，有一段缩 进的代码，这段代码表示的是函数体。本例中的函数体是一个&tum语句，它返回的是函数调 用的结果。如果没有return,那么函数返回的是Noneo

Pythoii中的函数调用通过“函数名（）”实现。例如，调用以上定义的add\_Number（）函数， 调用方式如下所示。

»> add\_number （10 z 2 0）

30

函数的更多知识将在第8章进行详细讲解。

类aw蒙

Python支持面向对象编程，类和对象是面向对象编程的两个核心概念。其中，类是对一类 事物的抽象描述，对象是实际存在的事物。例如，我们手中的书就是对象，坐着的凳子也是对象。

在描述一个事物(对象)时，需要考虑两个方面：

1. 它是什么样的(属性或者特征)。
2. 它可以做什么(•行为)。

在Pythoii中，一个对象的特征被称为对象的属性，一个对象的行为被称为对象的方法， 我们把具有相同特征和行为的事物归为一 “类”，例如，人类、动物类等。

类的定义使用class实现，看下面的例子：

»> class Person (object) : # 类

* . . def speak (self) : # 方法
* print(? hello worldf)

对象是通过类创建出来的，创建对象、调用对象中方法的示例如下所示。

»> p = Person () #创建一个p对象

»> p. speak () #通过p对象调用speak方法

hello world

关于面向对象的更多知识，将在第10、11章进行专门的讲解。

ago模块

从本质上来说，函数和对象都是为了更好地组织已经有的程序，以方便重复利用。模块 (module )也是为了同样的目的。在Python中,一个.py文件就构成一个模块,模块名即文件名。 通过在当前文件中导入模块，可以调用模块文件中包含的功能。 ’

创建firstly文件，文件中的内容如下:

def laugh():

print ? HaHaHaHa ,

创建secondly文件，在该文件中导入first模块，并调用fW模块中的函数。secondly文 件中的内容如下：

import first # 将 first 文件引入

for i in range(10):

first. laugh ()

在second.py中，使用了 firstly中定义的laugh()函数。

引入模块后，可以通过“模块•对象”的方式来调用引入模块中的某个对象。上面例子中， ftrst为导入的模块，laugh()是所调用的函数。

Python中还有其他的导入方式:

import a as b #引入模块a,并将模块a重命名为b

from a import functionl # 从模块 3 中引入 functionl 对象

使用ufrom...import xxxn语句导入a中内容时,不用再说明模块,即直接使用functionl, 而不是 a. function lo

from a import \* #从模块a中引入所有对象

调用為中对象时，不用再说明模块，即直接使用对象，而不是為•对象。

关于模块的详细知识，将在第13章进行详细讲解。