单 元 3

Python 流程控制

Python 流程控制结构主要分为 3 种：顺序结构、选择结构和循环结构。顺序结构是按照每条语句的先后顺序依次执行每条语句，所有语句都执行且执行一次。选择结构则根据条件有选择地执行某语句块。循环结构重复执行某语句块若干次。

3.1 顺序结构

顺序结构是流程控制中最简单的一种结构。该结构的特点是按照语句的先后顺序依次执行，每条语句只执行一次。单元 2 中所有的实例全部是顺序结构。

顺序结构的程序设计方法如下所述。

(1) 根据要解决的问题确定变量的个数。

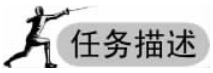
(2) 如果变量的值需要直接给出，如一个常量，需设计相应的赋值语句。

(3) 如果变量的值需要用户从键盘输入，需设计相应的输入语句。

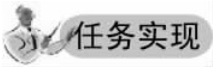
(4) 如果变量是保存运算的结果，需设计相应的处理语句，如把相应的数学公式转换为 Python 运算表达式，或编写 Python 函数调用语句等。

(5) 输出相应的信息和结果变量值。

任务 3-1 计算椭球的表面积和体积



编写一个 Python 程序，计算椭球的表面积和体积。



1. 设计思路

椭球在 xyz-笛卡儿坐标系中的方程是：$x^2/a^2+ y^2/b^2+z^2/c^2=1$。椭圆体的表面积 $S=\frac{4}{3}ab\pi$，椭圆体的体积 $V=\frac{4}{3}\pi abc$。根据公式和题目要求，可确定至少需要 5 个变量，其中 3 个存放从键盘输入的方程系数，另外 2 个存放计算后的表面积和体积。

2. 源代码清单

程序代码如表 3-1 所示。

表 3-1　任务 3-1 程序代码

#程序名称 task3_1.py	
序号	程 序 代 码
1	import math　#引入数学模块
2	print("请输入椭球方程的 3 个系数: ")
3	#从键盘输入方程的系数并转换为浮点型
4	a = float(input("请输入 a: "))
5	b = float(input("请输入 b: "))
6	c = float(input("请输入 c: "))
7	s = 4/3 * a * b * math.pi　#把公式转换为相应的 Python 语句,并计算
8	v = 4/3 * a * b * c * math.pi
9	#格式化输出数据.其中,大括号中的 0 表示第一个参数,.3f 表示小数点之后保留 3 位
10	print("x1 = {0:.3f},x2 = {1:.3f}".format(s,v))

任务 3-1 运行结果举例.txt

3.2　选择结构

在实际应用中,有时需要通过某个判断来决定任务是否执行或者执行的方式。对于这样的情况,仅有顺序结构控制是不够的,需要选择结构。

Python 中的 if 语句实现了选择结构控制,还可以使用 if-elif 结构来实现多分支控制。与其他程序设计语言相比,Python 中没有 switch 语句,但是可以通过其他方式获得类似 switch 语句功能的效果。

3.2.1　if-else 条件语句

Python 中的选择结构使用 if 和 else 关键字来构造,语法如下:

```
if 条件:
    条件为真时要执行的语句块
else:
    条件为假时要执行的语句块
```

选择结构根据条件的判断结果来决定执行哪个语句块。在任何一次运行中,两个分支的语句块只执行其中的一个。不可能两个语句块同时执行。选择结构执行完毕,继续执行其后的语句。

注意

（1）if 和 else 语句末尾的冒号不能省略。

（2）Python 通过严格的缩进来决定一个块的开始和结束，因此为真或为假的语句块都必须向右缩进相同的距离。

（3）条件可以是关系表达式或逻辑表达式，也可以是各种类型的数据。对于数值型数据(int，float，complex)，非零为真，零为假。对于字符串或集合类数据，空字符串和空集合为假，其余为真。

（4）else 分支可以省略。在单分支结构中，当条件为假时，继续执行 if 语句块之后的代码。else 不能单独使用。

（5）if 可以嵌套使用。

任务 3-2　输出最大的数

任务描述

编写一个 Python 程序，用户输入 3 个数，输出最大的数。

任务实现

1. 设计思路

分别定义 3 个变量来存放 3 个数。假定第一个变量是最大值，分别与第二个数和第三个数进行比较。如果发现逆序，则修改。最后输出结果。

2. 源代码清单

程序代码如表 3-2 所示。

表 3-2　任务 3-2 程序代码

#程序名称 task3_2.py	
序号	程 序 代 码
1	print("请输入 3 个数：")
2	#把键盘输入的字符串型的数据转换为浮点型并赋值给 a
3	a = float(input("请输入 a："))
4	b = float(input("请输入 b："))
5	c = float(input("请输入 c："))
6	maxNum = a　#假定 a 是最大值
7	#与 b 进行比较. 如果 b 大，修改当前最大值为 b
8	if (maxNum < b):　　#此处的冒号不可省略
9	maxNum = b　　#此语句必须缩进 4 格
10	if(maxNum < c):
11	maxNum = c
12	print("最大值是：{0:.3f}".format(maxNum))　#format 格式控制输出结果

任务 3-2 运行结果举例.txt

任务 3-3　计算一元二次方程的根

任务描述

编写一个 Python 程序，计算一元二次方程的根。

任务实现

1. 设计思路

根据题目，可确定至少需要 5 个变量，其中 3 个用来存放从键盘输入的方程系数，另外 2 个变量存放计算后得到的方程的根。还需要增加判断功能，如 a 不能为零，方程是否有实根。需要把如下公式转换为 Python 语句。

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

2. 源代码清单

程序代码如表 3-3 所示。

表 3-3　任务 3-3 程序代码

＃程序名称 task3_3.py

序号	程 序 代 码	
1	import math	＃导入数学函数模块
2	print("请输入一元二次方程的 3 个系数：")	
3	＃从键盘接收输入的数据，转换为浮点型并赋值给 a	
4	a = float(input("请输入 a："))	
5	b = float(input("请输入 b："))	
6	c = float(input("请输入 c："))	
7	＃判断是否是一元二次方程	
8	if(a == 0):	
9	＃a 为 0，不是一元二次方程	
10	print("二次系数不能为零，非一元二次方程！")	
11	＃exit()函数退出当前程序的运行	
12	exit(0)	
13	tmp = b * b - 4 * a * c	＃计算 delt
14	＃if 判断方程是否有实根	
15	if(tmp >= 0):	
16	x1 = (- b + math.sqrt(b * b - 4 * a * c))/(2 * a)	＃计算并输出方程的实根
17	x2 = (- b - math.sqrt(b * b - 4 * a * c))/(2 * a)	
18	print("x1 = {0:.3f},x2 = {1:.3f}".format(x1,x2))	

续表

序号	程 序 代 码
19	else:
20	print("该方程无实根")

任务 3-3 运行结果举例.txt

3.2.2 if-elif-else 判断语句

在实际中经常存在两种以上可能的选择，比如，学生成绩的等级转换，一个学生的成绩可以转换为五个不同的等级。Python 中提供了 if-else 语句的扩展。

if-elif-else 语法格式如下：

```
if 条件 1:
    条件 1 为真时执行的语句块 1
elif 条件 2:
    条件 1 为假且条件 2 为真时执行的语句块 2
⋮
elif 条件 n:
    条件 1 至条件 n－1 全部为假且条件 n 为真时执行的语句块 n
else:
    上述条件都不满足时执行的语句块 n＋1
```

执行过程说明如下：

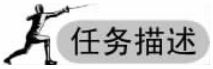
(1) 首先判断条件 1，如果其值为 True，执行语句块 1，然后结束整个选择结构。

(2) 如果条件 1 的值为 False，则判断条件 2；如果其值为 True，执行语句块 2，然后结束整个选择结构。

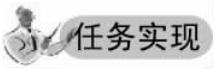
(3) 如果表达式 2 的值为 False，继续往下判断其他表达式的值。

(4) 如果所有表达式的值都为 False，则执行 else 之后的语句块 n＋1。

任务 3-4 成绩分等



编写一个 Python 程序，用户输入成绩，输出该成绩的等级。成绩等级划分原则为：90 分以上为“优秀”，80～90 分为“良好”，70～80 分为“中等”，60～70 分为“及格”，60 分以下为“不及格”。



1. 设计思路

定义一个变量来接收键盘输入的成绩。首先判断成绩是否有效，然后根据等级转换规则构造多分支判断结构并输出结果。

2. 源代码清单

程序代码如表 3-4 所示。

表 3-4　任务 3-4 程序代码

＃程序名称 task3_4.py	
序号	程 序 代 码
1	print("成绩等级换算")
2	grade = float(input("请输入学生的成绩"))
3	＃判断成绩是否在[0,100]分之间
4	if(grade < 0 or grade > = 100):
5	＃成绩无效,则退出,不再执行后面的代码
6	print("成绩无效")
7	exit(0)
8	＃构造多分支选择结构
9	if(grade > = 90):
10	＃成绩大于等于 90,输出"优秀"
11	print("{0}的成绩等级是{1}".format(grade,"优秀"))
12	elif(grade > = 80):
13	print("{0}的成绩等级是{1}".format(grade,"良好"))
14	elif(grade > = 70):
15	print("{0}的成绩等级是{1}".format(grade,"中等"))
16	elif(grade > = 60):
17	print("{0}的成绩等级是{1}".format(grade,"及格"))
18	else:
19	print("{0}的成绩等级是{1}".format(grade,"不及格"))

任务 3-4 运行结果举例.txt

3.2.3　if 语句的嵌套

在 if-else 语句的缩进块中可以包含其他 if-else 语句,称作嵌套 if-else 语句。在嵌套的选择结构中,根据对齐的位置来进行 else 与 if 的配对。

简单的形式如下:

```
if 条件 1:
    if 条件 2:
        条件 1 为真且条件 2 为真时执行的语句块 1
    else:
        条件 1 为真且条件 2 为假时执行的语句块 2
else:
    条件 1 为假时执行的语句块 3
```

执行过程说明如下:

（1）条件 1 为真时，判断条件 2。条件 1 为假时，执行语句块 3。

（2）如果条件 2 为真，执行语句块 1，然后结束整个选择结构。如果条件 2 为假，执行语句块 2，然后结束整个选择结构。

任务 3-5　判断三角形的类型

任务描述

编写一个 Python 程序，用户输入三角形的三条边的边长，判断是否是三角形及三角形的类型：直角三角形、钝角三角形和锐角三角形。

任务实现

1. 设计思路

根据题目，可确定至少需要 3 个变量，用来存放从键盘输入的三角形的三条边。首先判断是否是三角形，然后根据三角形的判断公式构造多分支选择结构，并输出结果。

2. 源代码清单

程序代码如表 3-5 所示。

表 3-5　任务 3-5 程序代码

#程序名称 task3_5.py

序号	程 序 代 码
1	print("请输入三角形三条边的边长：")
2	a = int(input("请输入第一条边的边长"))
3	b = int(input("请输入第二条边的边长"))
4	c = int(input("请输入第三条边的边长"))
5	#第一个 if 判断是否是三角形
6	if(a + b > c and a + c > b and c + b > a):
7	#满足三角形的条件，继续判断三角形的类型
8	#下面的 if 是直角三角形的判断
9	if(a ^2 + b ^2 == c ^2 or a ^2 + c ^2 == b ^2 or b ^2 + c ^2 == a ^2):
10	print("这是个直角三角形")
11	#下面的 if 是锐角三角形的判断
12	elif(a ^2 + b ^2 > c ^2 or a ^2 + c ^2 > b ^2 or b ^2 + c ^2 > a ^2):
13	print("这是个锐角三角形")
14	#上述两个条件都不满足，则是钝角三角形
15	else:
16	print("这是个钝角三角形")
17	else:
18	#不满足三角形的条件
19	print("这不是三角形")

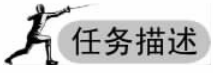
任务 3-5 运行结果举例.txt

3.2.4 switch 语句的替代方案

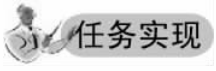
C 语言或 Java 语言都支持 switch 多分支结构，但是 Python 没有提供 switch 关键字。然而在有些情况下，类似于 switch 结构的代码的清晰性和可读性要强于 if 多分支结构。

在 Python 中可以通过字典方式模拟类似的结果，其实现方法分为两步：首先，定义一个字典。字典是由键值对组成的集合。字典的使用参见单元 4。其次，调用字典的 get()函数获取相应的表达式。

任务 3-6 简单的计算器



编写一个 Python 支持的算术运算符的简单计算器。用户输入格式为 data1 op data2。其中，data1 和 data2 是参加运算的两个数；op 为运算符，可以是＋、－、＊、/、％、//和 ^。



1. 设计思路

根据题目要求，用一个字符串接收一个运算式，需要使用正则表达式分解出两个操作数和一个运算符，然后使用字典方式构造多分支结构，计算表达式的结果并输出。

2. 源代码清单

程序代码如表 3-6 所示。

表 3-6 任务 3-6 程序代码

＃程序名称 task3_6.py	
序号	程 序 代 码
1	import re ＃导入正则表达式模块
2	print("简单计算器")
3	str = input("请输入只有一个运算符的式子(如 5 + 3):")
4	p = re.compile(r'\\d + ') ＃生成正则表达式对象，用于提取数字
5	op1 = int(p.findall(str)[0]) ＃获得提取到的第一个运算数
6	op2 = int(p.findall(str)[1]) ＃获得提取到的第二个运算数
7	q = re.compile(r'\W + ') ＃定义提取非字母字符
8	opt = q.findall(str)[0] ＃获得提取到的运算符
9	＃在做除法运算之前检测是否会被零除
10	if((opt == '/'or opt == '％'or opt == '//') and op2 == 0):
11	print("除数为零，非法!") ＃检测到被零除，提示并退出
12	exit(0)
13	＃定义运算字典.po 是字典的名字，字典中用逗号分隔的是键值对
14	po = {
15	'＋':op1 + op2,'－':op1 － op2,
16	'＊':op1 ＊ op2,'/':op1/op2,

续表

序号	程 序 代 码
17	'^':op1 ^op2,'%':op1 %op2,
18	'//':op1//op2
19	}
20	#通过字典对象调用 get()函数,获得参数 opt 对应的键值
21	result = po.get(opt)
22	#输出运算结果
23	print('{0}{1}{2} = {3}'.format(op1,opt,op2,result))

任务 3-6 运行结果举例.txt

3.3 循环结构

循环结构是结构化程序设计常用的结构,可以简化程序,或解决顺序结构和选择结构无法解决的问题。循环是指在满足一定条件的情况下,重复执行一组语句的结构。重复执行的语句称作循环体。

3.3.1 while 循环

while 循环语法格式如下:

```
[初始化语句]
while (循环条件):
    语句块
    [迭代语句]
```

循环结构的执行流程如下所述:执行到 while 循环的时候,先判断"循环条件",如果为 True,则执行下面缩进的循环体;执行完毕,再次判断"循环条件"。若为 True,继续执行循环体;若为 False,不再执行循环体,循环结束。循环结束后,继续执行循环结构之后的语句。

(1) while 条件之后的冒号不能丢掉。

(2) 如果循环条件不成立,则循环体一次也不执行。

(3) 如果循环控制变量的改变不是向着循环结束条件的方向变化,或者循环条件是一个结果为 True 的表达式,则该循环结构是死循环或无限循环,即循环结构在没有特殊语句的控制下,循环会一直运行,无法结束。无限循环经常用于某些特定的场合,如菜单设计中。

(4) 初始化语句和迭代语句可以没有,用于特殊的场合。

(5) 循环体的所有语句必须对齐,且与 while 的位置具有相同的缩进。

(6) 若 while 循环结构的循环体只有一条语句,这条语句可以直接跟在 while 行尾的冒号之后,即写在同一行上。

循环结构的设计其实就是循环次数的确定。对于 while 循环结构设计,只要掌握了如下“三要素原则”,就可以设计出循环结构。

(1) 初始化语句:循环控制变量赋初值,或其他循环中用到的变量的初始化。

(2) 循环条件:循环结构继续执行的条件,是一个结果为 True 或 False 的表达式。

(3) 迭代语句:通常是循环控制变量的改变,且朝着循环结束条件的方向变化,使得循环可以正常结束。

例如,设计一个执行 100 次的循环,假定 i 为循环控制变量,则循环的 3 个要素设计如下:

(1) i=1

(2) i<=100

(3) i=i+1

再比如,判断一个整数 n 是否是素数,需要判断 2～n－1 之间的所有数与 n 是否有整除关系。假定 i 为循环控制变量,则循环的 3 个要素设计如下:

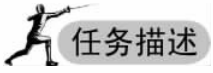
(1) i=2

(2) i<=n－1

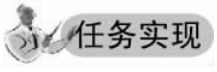
(3) i=i+1

循环的三个要素确定以后,只要把这 3 个要素放在循环语句中合适的位置,再添加要重复执行的语句(循环体),便构成一个可执行的循环结构。

任务 3-7　自然数求和



编写一个 Python 程序,计算 1000 以内所有偶数的和。



1. 设计思路

根据题目,可确定至少需要 2 个变量,其中一个用作循环控制变量,另一个是用于存放和的累加器。循环控制变量用 i 表示,则循环的 3 个要素是:①i=2;②i<1000;③i=i+2,生成 1000 内的所有偶数。每执行一次,生成一个新的偶数。累加器 s 必须赋初值 0。循环体是一条累加语句 s=s+i。

2. 源代码清单

程序代码如表 3-7 所示。