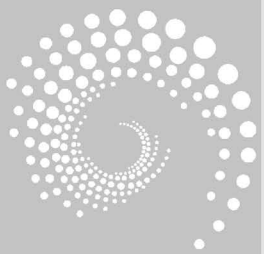
第3章

开始程序设计

CHAPTER 3



【教学目标】

知识目标：

- 理解算法的定义、特性和描述方法。
- 掌握 Python 的语法规则，包括缩进和注释。
- 了解异常处理的方法。

技能目标：

- 能够运用选择结构和循环结构编写程序，能够处理程序中可能出现的异常。
- 掌握 Python 的语法规则、选择结构、循环结构和异常处理。
- 学会使用选择结构和循环结构进行程序设计。

情感与思政目标：

- 增强学生解决问题的信心和耐心。
- 培养诚信、自律等价值观，遵循编程规范。

【引言】

了解了 Python 的数据类型及其运算、操作方法，接下来就要开始设计程序来解决问题了。开发程序与解数学题类似，需要一定的解题思路，即算法；开发程序与写作文也类似，需要一定的语法和结构。本章将介绍程序设计常用的算法、Python 语言的语法和程序的控制结构。

3.1　程序与算法

计算机之所以能够处理复杂的问题，主要依靠于程序的运行。程序设计的一般过程包括 5 个步骤：分析问题、确定数学模型、算法设计、编写程序、运行测试。其中，算法是程序设计的核心，没有算法就没有了程序设计的灵魂。

3.1.1　算法定义与特性

算法就是解决问题的方法和步骤，解决问题的过程就是算法实现的过程。

著名计算机科学家 Donald E. Knuth 曾把算法的特性归纳为以下 5 点。

(1) 有穷性。任意一个算法在执行有穷个计算步骤后必须终止。

(2) 确定性。每一个计算步骤必须精确地定义，无二义性。

(3) 可行性。有限多个步骤应该在一个合理的范围内进行。

(4) 输入。一般有 0 个或多个输入。

(5) 输出。一般有若干输出信息，是反映对输入数据加工后的结果。没有输出结果的算法是毫无意义的。

3.1.2　常用的算法

人们通过长期的研究开发工作总结了一些基本的算法设计方法，例如，枚举法、迭代法、递推法、分治法、回溯法、贪心法和动态规划法等。

1. 枚举

枚举法也称为穷举法或试凑法。它的基本思想是采用搜索的方法，根据题目的部分条件确定答案的大致搜索范围，然后在此范围内对所有可能的情况逐一验证，直到所有情况验证完。若某个情况符合题目的条件，则为本题的一个答案；若全部情况验证完后均不符合题目的条件，则问题无解。枚举法是一种比较耗时的算法，主要利用计算机快速运算的特点。枚举的思想可解决许多问题。

枚举法解决问题的关键主要在于以下三点。

(1) 确定搜索的范围，尽量不遗漏但又避免出现问题求解以外的范围。

(2) 确定满足的条件，把所有可能的条件一一罗列。

(3) 枚举解决问题效率不高，因此，为提高效率，根据解决问题的情况，应尽量减少内循环层数或每层循环次数。

2. 查找

查找在人们日常生活中经常会遇到，利用计算机快速运算的特点，可方便地实现查找。查找的方法有很多，对不同的数结构有对应的方法。例如，对无序数据，用顺序查找；对有序数据，采用二分法查找；对某些复杂的结构的查找，可用树状查找方法。

顺序查找是根据查找的关键值与数组中的元素逐一比较。顺序查找对数组中的数据不

要求有序,查找效率比较低。

二分法查找是在数据量很大时采用的一种高效查找法。采用二分法查找时,数据必须是有序的。假设数组是按递增有序的,实现的方法是已知查找区间的下界 low、上界 high,当 high≥low 时,中间项 mid=(low+high)/2,根据查找的 key 值与中间项 a[mid]比较,有以下三种情况。

(1) key>a[mid],则 low=mid+1,后半部作为继续查找的区域。

(2) key<a[mid],则 high=mid−1,前半部作为继续查找的区域。

(3) key=a[mid],则查找成功,结束查找。

这样每次查找区间缩小一半,直到查找到或者区间内没有要查找的值。

3. 排序

在日常生活和工作中,许多问题的处理过程都要依赖于数据的有序性,因此,需要将数据整理为有序数据,即排序。常用的排序算法有选择排序和冒泡排序。

选择排序是最为简单且易于理解的算法,基本方法是每次在无序数中找到最小数的下标,然后与第一个位置的数交换。

冒泡排序与选择排序相似,选择排序在每一轮中寻找最值小(递增次序)的下标,然后与应放位置的数交换位置。而冒泡排序在每一轮排序时将相邻两个数组元素进行比较,次序不对时立即交换位置,一轮比较结束小数上浮,大数沉底。有 n 个数则进行 $n-1$ 轮上述操作。

4. 迭代

迭代法又称递推法,是利用问题本身所具有的某种递推关系求解问题的一种方法。其基本思想是从初值出发,归纳出新值与旧值间直到最后值为止存在的关系,从而把一个复杂的计算过程转换为简单过程的多次重复,每次重复都从旧值的基础上递推出新值,并由新值代替旧值。

除上述 4 个常用算法之外,还有诸如贪心算法、分治法、回溯法等方法也应用在程序设计中,需要根据不同的情况选择适合的算法来解决问题。

3.1.3 算法描述

算法的表示方法有很多,常用的有自然语言、传统的流程图、伪代码和计算机语言等。目前使用较为广泛的是流程图(便于理清程序的处理过程)和计算机语言(用于在计算机上实现程序功能)。

1. 自然语言

用人们使用的语言,即自然语言描述算法。用自然语言描述算法通俗易懂,但存在以下缺陷。

(1) 易产生歧义性,往往需要根据上下文才能判别其含义,不太严格。

(2) 语句比较烦琐、冗长,并且难以清楚地表达算法的逻辑流程,尤其对描述含有选择、循环结构的算法,不太方便和直观。

2. 流程图

流程图是描述算法的常用工具，采用一些图框、线条以及文字说明来形象、直观地描述算法处理过程。美国国家标准协会(ANSI)规定了一些常用的流程图符号，如表 3-1 所示。

表 3-1　流程图的常用符号

符 号 名 称	图　　形	功　　能
开始与结束框		表示一个过程的开始或结束
输入/输出框		表示数据的输入和输出
处理框		表示算法中的各种处理操作
判断框		表示算法中的条件判断操作
流程线		表示算法的执行方向

如计算两数之和，用流程图表示如图 3-1 所示。

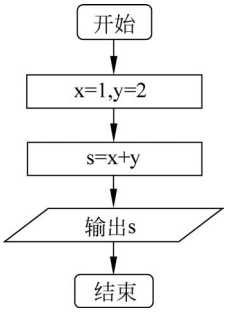


图 3-1　计算两数之和的流程图

3. 伪代码

由于绘制流程图较费时，自然语言易产生歧义性和难以清楚地表达算法的逻辑流程等缺陷，人们开始采用伪代码。伪代码产生于 20 世纪 70 年代，也是一种描述程序设计逻辑的工具。

伪代码是用介于自然语言和计算机语言之间的文字和符号来描述算法，有如下简单约定。

(1) 每个算法以 Begin 开始，以 End 结束。若仅表示部分编写代码可省略。

(2) 每一条指令占一行，指令后不跟任何符号。

(3) “//”标志表示注释的开始，一直到行尾。

(4) 算法的输入/输出以 Input/Print 后加参数表的形式表示。

(5) 用“←”表示赋值。

(6) 用缩进表示代码块结构，包括 While 和 For 循环、If 分支判断等。块中多句语句用一对{}括起来。

(7) 数组形式为数组名[下界…上界]；数组元素为数组名[序号]。

(8) 一些函数调用或者处理简单任务可以用一句自然语言代替。

如计算两数之和的伪代码如图 3-2 所示。

```
BEGIN
    x←1
    y←2
    s=x+y
    Print  t
End
```

图 3-2　计算两数之和的伪代码

4. 计算机语言

计算机无法识别自然语言、流程图、伪代码。这些方法仅为了帮助人们描述、理解算法,要用计算机解题,就要用计算机语言描述算法。只有用计算机语言编写的程序才能被计算机执行。用计算机语言描述算法必须严格遵循所选择的编程语言的语法规则。

例如,计算两数之和的 Python 程序代码如下。

```
x,y = 1,2
s = x + y
print(s)
```

3.2 Python 语法规则

Python 语言除规定了基本的数据类型及其使用方法之外,还有一定的语法要求,如缩进和注释。

3.2.1 缩进

Python 用缩进来标识代码块,它有着严格的缩进规则。缩进是指代码行前面的空白区域,表示代码之间的层次关系,同一层次的代码块必须有相同的缩进。在编写代码时,一般用 Tab 键实现缩进。

例如,在以下代码中,if 和 else 对齐,其中包含的语句常用 Tab 缩进实现对齐。

```
a = int(input("请输入一个数字:"))
if a > 0:
    print(f"{a}是正数")
else:
    print(f"{a}不是正数")
```

3.2.2 注释

注释是程序员在代码中加入的说明性文字,用来对变量、语句、方法等进行功能性说明,以提高代码的可读性。程序运行时编译器或者解释器会忽略注释文字,因此注释不影响程序的运行结果。在 Python 中注释方式有两种写法。单行注释语句用 # 开始,单行注释可以单独出现在一行,也可以与代码放在同一行。多行注释语句使用连续三个双引号或者单引号对表示。注释的写法如下。

```
#这是单行注释
print("Hello,python!") #这是单行注释
'''
这是多行注释的第 1 行
这是多行注释的第 2 行
这是多行注释的第 3 行
'''
```

小提示：三引号与赋值语句等号相连时，可用于定义多行字符串，不与赋值语句相连时，用于多行注释。按 Ctrl＋/组合键，可以将代码转换为注释。

3.3　选择结构

程序的控制结构有顺序结构、选择结构、循环结构，它们是算法的三种基本结构。顺序结构指计算机按照语句出现的先后次序依次执行。选择结构是设定条件进行判断，然后根据可能不同的情况进行不同的处理。循环结构是根据条件判断控制循环体语句的执行次数。本节首先介绍选择结构。

选择结构也叫分支结构，可以根据条件来控制代码的执行分支。Python 中使用 if 语句来实现分支结构。其语法格式如下。

```
if(条件表达式):
    语句/语句块
```

分支结构包含多种形式：单分支、双分支和多分支。

条件表达式可以是关系表达式、逻辑表达式、算术表达式等。语句/语句块可以是单个语句，也可以是多个语句，多个语句的缩进必须对齐一致。

3.3.1　单分支结构

基本语法：

```
if(条件表达式):
    语句/语句块 1
```

流程图如图 3-3 所示，当条件表达式的值为 True 时，表示条件满足，则语句块将被执行，否则该语句块不被执行。

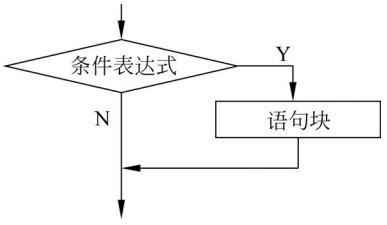


图 3-3　if 单分支语句结构流程图

注意：每个 if 条件后要使用冒号(:)，表示接下来是满足条件后要执行的语句。使用缩进来划分语句块，相同缩进数的语句在一起组成一个语句块。

例 3-1：当用户输入年龄 age 后，判断 age 是否大于或等于 18，若满足条件，输出“你已成年”。

```
age = int(input('请输入你的年龄:'))
if age >= 18:
    print("你已成年")
```

3.3.2 双分支结构

基本语法:

```
if 条件:
    语句 1
else:
    语句 2
```

流程图如图 3-4 所示,当条件表达式的值为 True 时,表示条件满足,则执行语句块 1,否则执行语句块 2。

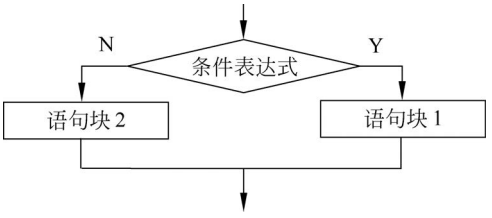


图 3-4 if 双分支语句结构流程图

例 3-2:输入一个整数,如果大于 0,输出“正数”;否则输出“不是正数”。

```
x = float(input("请输入一个数:"))
if x > 0:
    print("是正数")
else:
    print("不是正数")
```

3.3.3 多分支结构

基本语法:

```
if 条件 1:
    语句 1
elif 条件 2:
    语句 2
…
else:
    语句 n
```

流程图如图 3-5 所示,首先计算表达式 A,如果其值为 True,则执行语句块 1,否则计算表达式 B,如果其值为 True,则执行语句块 2,否则计算表达式 C,如果其值为 True,则执行语句块 3,以此类推,如果所有表达式计算的结果都为 False,则执行 else 后的语句块 n。

例 3-3:输入一个学生的整数成绩 n,当成绩大于或等于 90 分时,输出“成绩优秀!”;当成绩大于或等于 80 且小于 90 时,输出“成绩良好!”;当成绩大于或等于 70 且小于 80 时,输出“成绩中等!”;当成绩大于或等于 60 且小于 70 时,输出“成绩合格!”;否则输出“成绩不合格!”。

```
n = float(input("请输入一个成绩:"))
if n >= 90:
```

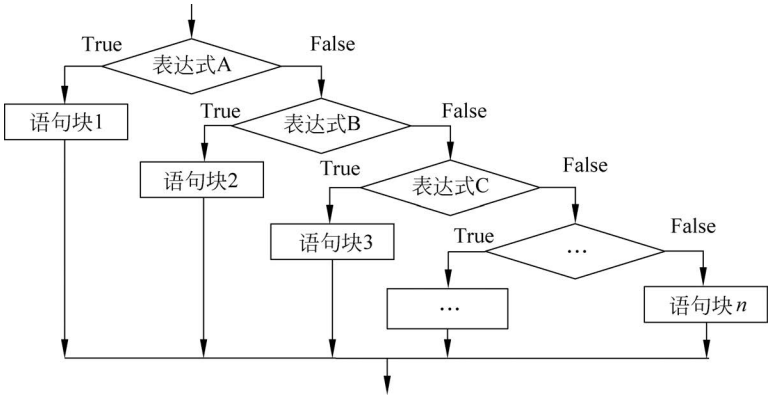


图 3-5　if 多分支语句结构流程图

注：如果省略 else 子句，则表达式都为 False 时，什么语句也不执行。

```
    print("成绩优秀!")
elif n >= 80:
    print("成绩良好!")
elif n >= 70
    print("成绩中等!")
elif n >= 60
    print("成绩合格!")
else:
    print("成绩不合格!")
```

3.3.4　精选案例

1. BMI 计算

BMI 指数即身体质量指数，是目前国际常用的衡量人体胖瘦程度以及是否健康的一个标准。BMI 指数计算公式如下：体质指数(BMI)＝ 体重(kg)÷(身高^2)(m)。BMI 正常值为 20～25，超过 25 为超重，30 以上为肥胖，小于 20 为偏瘦。请编写程序，输入身高和体重，输出对应体型。

思路分析：

第一步，获取用户输入的体重 weight 和身高 height 两个变量。

第二步，计算 BMI 的值。

第三步，用 if 多分支进行判断，并分别输出结果。

编写代码：

```
height = float(input("请输入您的身高(米): "))
weight = float(input("请输入您的体重(千克): "))
bmi = weight/height**2
if bmi < 20:
    print("您的体型属于: 偏瘦")
elif 20 <= bmi < 25:
    print("您的体型属于: 正常")
elif 25 <= bmi < 30:
    print("您的体型属于: 超重")
```

```
else:
    print("您的体型属于: 肥胖")
```

2. 个税计算

个人所得税的税率如表 3-2 所示,超过 5000 元的需要缴税。请输入月收入,计算应缴税额=应缴税所得额×税率-速算扣除数。

表 3-2 个人所得税

全月应缴税所得额	税率	速算扣除数/元
不超过 1500 元	3%	0
1500～4500 元	10%	105
4500～9000 元	20%	555
9000～35 000 元	25%	1005
35 000～55 000 元	30%	2755
55 000～80 000 元	35%	5505
超过 80 000 元	45%	13 505

思路分析:

第一步,获取用户输入的月收入变量。

第二步,用 if 多分支对月收入进行判断,计算应缴税额并输出结果。

编写代码:

```
income = float(input("请输入您的税前月收入: "))
taxable_income = income - 5000
if taxable_income <= 0:
    print("不用缴税")
elif taxable_income <= 1500:
    print("应缴税额为:",taxable_income * 0.03 - 0)
elif taxable_income <= 4500:
    print("应缴税额为:",taxable_income * 0.1 - 105)
elif taxable_income <= 9000:
    print("应缴税额为:",taxable_income * 0.2 - 555)
elif taxable_income <= 35000:
    print("应缴税额为:",taxable_income * 0.25 - 1005)
elif taxable_income <= 55000:
    print("应缴税额为:",taxable_income * 0.3 - 2755)
elif taxable_income <= 80000:
    print("应缴税额为:",taxable_income * 0.35 - 5505)
else:
    print("应缴税额为:",taxable_income * 0.45 - 13505)
```

3.4 循环结构

当有些语句块需要重复执行时,为了简化代码量,可以使用循环结构来实现。循环结构可以用 for 和 while 两种方法来描述。

3.4.1　for 循环结构

for 循环表示当满足条件表达式时，重复执行循环语句，否则跳出循环，其语法格式为

```
for 变量 in range(start, stop[, step]):
    循环语句
```

其中，

start 指计数开始值，默认为 0。例如，range(5)等价于 range(0,5)。

stop 指计数结束值，但不包括 stop。例如，range(0,5)是[0,1,2,3,4]，没有 5。

step 指步长，默认为 1。例如，range(0,5,2)步长是 2，则它的结果为[0,2,4]。

for 循环的流程图如图 3-6 所示。

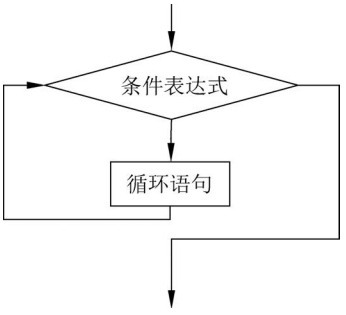


图 3-6　for 循环的流程图

例 3-4：输出 5 遍“Hello,Python!”。

```
for i in range(5):
    print("hello,python!")
```

执行结果：

```
hello,python!
hello,python!
hello,python!
hello,python!
hello,python!
```

例 3-5：编写程序，在操场上跑步，当跑完第 3 圈时停止。

```
for i in range(3):
    print(f"跑完了第{i+1}圈")
```

执行结果：

```
跑完了第 1 圈
跑完了第 2 圈
跑完了第 3 圈
```

例 3-6：计算 $s=n!$，其中，n 由键盘输入。

```
s = 1
n = int(input("请输入一个正整数:"))
for i in range(1,n+1):
    s = s * i
print(f"{n}的阶乘是:{s}")
```

执行结果：

```
请输入一个正整数:5
5 的阶乘是:120
```

例 3-7：计算 $s=1+2+\cdots+100$。

```
s = 0
for i in range(1,101):
    s += i
print(s)
```

执行结果：

```
5050
```

例 3-8：输入 5 个同学的成绩,计算平均成绩。

```
s = 0
for i in range(5):
    n = float(input("请输入成绩:"))
    s += n
print("平均成绩为:",s/5)
```

例 3-9：定义字符串“python”,将每个字母单独输出。

```
s = "python"
for c in s:
    print(c)
```

执行结果：

```
p
y
t
h
o
n
```

例 3-10：请用“*”输出直角三角形。

```
for i in range(5):
    print("*" * i)
```

执行结果：

```
*
**
***
****
```

例 3-11：请用“*”输出等腰三角形。

```
for i in range(5):
    print(" " * (5 - i) + "*" * (i * 2 - 1))
```

执行结果：

```
    *
   ***
  *****
*******
```

例 3-12：判断一个年份是否是闰年。闰年是指能被 4 整除但不能被 100 整除或能被 400 整除的年份。

```
n = int(input("请输入一个年份"))
if n % 4 == 0 and n % 100!= 0 or n % 400 == 0:
    print(f"{n}是闰年")
else:
    print(f"{n}不是闰年")
```

例 3-13：用 for 循环生成一个 1～9 的平方列表。

```
list1 = [i ** 2 for i in range(10)]
print(list1)
```

执行结果：

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

3.4.2　while 循环结构

for 语句适用于明确变量范围的情景，程序更为简洁。当变量范围不确定时，则需要使用 while 语句。其语法格式为

```
while 条件表达式:
    循环语句(块)
```

例 3-14：用 while 语句计算 $s=2+4+\cdots+98+100$。

```
s = 0
i = 2
while i <= 100:
    s += i
    i += 2
print(s)
```

执行结果：

```
2550
```

例 3-15：爸爸今年 40 岁，儿子今年 6 岁，求多少年后爸爸的年龄是儿子的两倍。

```
d_age = 40
s_age = 6
y = 0
while d_age!= s_age * 2:
    y += 1
    s_age += 1
    d_age += 1
print(y)
```

执行结果：

```
28
```

3.4.3　break 和 continue 语句


视频讲解

1. break 语句

在循环的过程中，如果需要中断循环，则需要使用 break 语句，可以极大地节省程序的时间复杂度，且便于获取中断时的变量值。

例 3-16：猜数游戏，给用户无数次机会去猜，如果猜对了输出“猜对了!”，否则提示“没猜对!”。

```
answer = 5
while True:
    n = int(input("请猜数:"))
    if n == answer:
        print("猜对了!")
        break
    else:
        print("没猜对!")
```

例 3-17：录入学生的成绩,以字母“p”为结束符,计算成绩数量、总分和平均值。

```
n,s = 0,0
while True:
    score = input("请输入成绩:")
    if score == "p":
        break
    s += float(score)
    n += 1
print(f"总分为{s},成绩数量为{n},平均成绩为{s/n}")
```

执行结果：

```
请输入成绩:80
请输入成绩:90
请输入成绩:79.5
请输入成绩:77
请输入成绩:p
总分为 326.5,成绩数量为 4,平均成绩为 81.625
```

例 3-18：编写程序判断一个数是否是素数。素数是只能被 1 和它本身整除的数。

```
n = int(input("请输入一个自然数:"))
for i in range(2,n):
    if n % i == 0:
        print(f"{n}不是素数")
        break
else:
    print(f"{n}是素数")
```

执行结果：

```
请输入一个自然数: 5
5 是素数
```

在本例中,for 循环与 else 语句一起使用。当 for 循环正常结束(没有遇到 break 语句)时,else 子句中的代码将被执行。如果 for 循环被 break 语句中断,则 else 子句中的代码将不会被执行。

2. continue 语句

continue 语句的作用是结束本次循环,紧接着执行下一次的循环。

例 3-19：定义字符串“python”,将除“y”以外的每个字母单独输出。

```
for c in "python":
    if c == "y":
        continue
    print(c)
```

执行结果：

```
p
t
h
o
n
```

3.4.4　循环嵌套

在一个循环体内又包含另一个完整的循环结构，称为循环的嵌套。当外层循环执行第一遍时，内层循环需要全部执行完方可执行外层循环第二遍。

循环嵌套的流程图如图 3-7 所示。

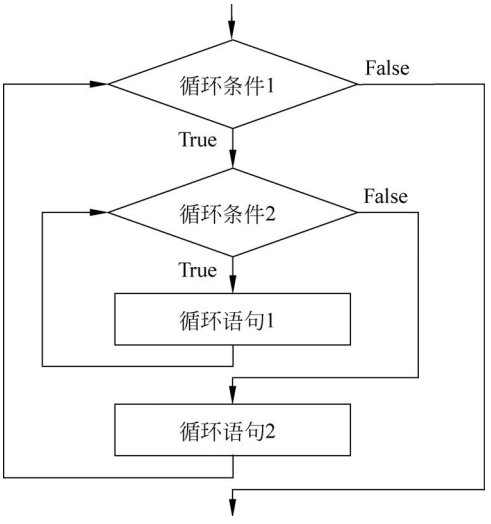


图 3-7　循环嵌套的流程图

例 3-20：打印九九乘法表。


```
for i in range(1,10):
    for j in range(1,i+1):
        print(f"{i}*{j}={i*j:2d}",end=" ")
    print()
```

执行结果如图 3-8 所示。

```
1*1= 1
2*1= 2  2*2= 4
3*1= 3  3*2= 6  3*3= 9
4*1= 4  4*2= 8  4*3=12  4*4=16
5*1= 5  5*2=10  5*3=15  5*4=20  5*5=25
6*1= 6  6*2=12  6*3=18  6*4=24  6*5=30  6*6=36
7*1= 7  7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49
8*1= 8  8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64
9*1= 9  9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```

图 3-8　九九乘法表

例 3-21：打印出 1,2,3 三个数字的所有排列。

```
for i in range(1,4):
    for j in range(1,4):
        for k in range(1,4):
            if i != j and i!= k and j!= k:
                print(i,j,k)
```

执行结果：

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

例 3-22：鸡兔同笼问题。假设共有鸡、兔 30 只，脚 90 只，求鸡、兔各有多少只。

```
for i in range(1,30):
    for j in range(1,30):
        if i + j == 30 and i * 2 + j * 4 == 90:
            print(i,j)
```

执行结果：

```
15 15
```

例 3-23：两个乒乓球队进行比赛，各出三人。甲队为 a,b,c 三人，乙队为 x,y,z 三人。抽签决定比赛名单。有人向队员打听比赛的名单。a 说他不和 x 比，c 说他不和 x、z 比，请编程找出三队赛手的名单。

```
    s = "xyz"
for i in range(3):
    for j in range(3):
        for k in range(3):
            if i!= 0 and k!= 0 and k!= 2 and i!= j and i!= k and j!= k:
                print(f"a - {s[i]},b - {s[j]},c - {s[k]}")
```

执行结果：

```
a - z,b - x,c - y
```


视频讲解

3.4.5 精选案例

1. 猴子吃桃

猴子第一天摘下若干桃子，当即吃了一半，还不过瘾，又多吃了一个。第二天早上又将剩下的桃子吃掉一半，又多吃了一个。以后每天早上都吃了前一天剩下的一半零一个。到第十天早上想再吃时，见只剩下一个桃子了。求第一天共摘了多少个桃子。

分析：倒推回去算，每一天的桃子数 n_1 和前一天的桃子数 n_2 之间可以提炼出公式 $n_2=(n_1+1)\times 2$，往前推算 9 次，即可得到第一天的桃子数量。

编写代码如下。

```
n = 1
for i in range(9):
    n = (n + 1) * 2
print(n)
```

执行结果：

```
1534
```

2. 列表元素计算

a 和 b 是两个列表变量，列表 a 为[3,6,9]已给定，从键盘输入列表 b，计算 a 中元素与 b 中对应元素乘积的累加和。例如，从键盘输入列表 b 为[1,2,3]，则累加和为 $1\times3+2\times6+3\times9=42$，因此，屏幕输出计算结果为 42。

分析：

第 1 步，定义 s 变量为 0，用于累加数据。

第 2 步，定义列表 a 和列表 b，其中，列表 a 已知，列表 b 需要由用户输入。

第 3 步，用 for 循环遍历两个列表对应位置的值，分别相乘再累加，并赋值到变量 s 中。

第 4 步，输出 s 的值。

编写代码如下。

```
a = [3,6,9]
b = eval(input("请输入 3 个数字,以逗号分隔")) # 例如[1,2,3]
s = 0
for i in range(3):
    s += a[i] * b[i]
print(s)
```

3. 斐波那契数列

根据斐波那契数列的定义 $F(0)=0$，$F(1)=1$，$F(n)=F(n-1)+F(n-2)(n\geqslant2)$，输出不大于 100 的序列元素。例如，屏幕输出实例为 0,1,1,2,3,…

分析：

第 1 步，初始化前两个数字，分别用变量 a、b 表示。

第 2 步，根据公式与结果分析，a 的初始值为 0，因此在后续的数列中，将 b 的值赋给 a，将原 $a+b$ 的值赋值给 b。在 Python 中，多变量可以同时赋值。最后输出所有的 a 即可。

第 3 步，不清楚循环次数，因此使用 while 来控制变量范围。

编写代码如下。

```
a, b = 0, 1
while a <= 100:
    print(a, end = ',')
    a, b = b,a + b
```

4. 列表元素计算并生成新列表

a 和 b 是两个长度相同的列表变量，列表 a 为[3,6,9]已给定，从键盘输入列表 b，计

算 a 中元素与 b 中对应元素的和形成的列表 c,在屏幕上输出。

例如,从键盘输入列表 b 为[1,2,3],键盘输出计算结果为[4,8,12]。

分析:获取用户输入列表,然后定义空列表 c,并用 for 循环将列表 a 和列表 b 中对应的元素相加,用 append()方法追加到列表 c 中,最后输出列表 c。

编写代码如下。

```
a = [3,6,9]
b = eval(input("请输入一个包含 3 个数字的列表:")) # 例如[1,2,3]
c = []
for i in range(3):
    c.append(a[i] + b[i])
print(c)
```

5. 多列表元素组合

a 和 b 是两个列表变量,列表 a 为[3,6,9]已给定,从键盘输入列表 b,将 a 列表的三个元素插入列表中对应的前三个元素的后面,并输出在屏幕上。

例如,从键盘输入列表 b 为{1,2,3},因此,屏幕输出计算结果为[1,3,2,6,3,9]。

分析:本例中列表 a 已知,获取用户输入 b 列表包含 3 个元素,要求将 a 列表中元素插入到 b 列表中对应元素的后面,要使用到 insert()函数,插入的索引位置变量初始值为 1,之后以 2 为 step 递增去插入。

编写代码如下。

```
a = [3,6,9]
b = eval(input("请输入一个包含 3 个数字元素的列表:")) # 例如[1,2,3]
j = 1
for i in range(len(a)):
    b.insert(j,a[i])
    j += 2
print(b)
```

6. 生成新列表

获得用户输入的以逗号分隔的三个数字,记为 a、b、c,以 a 为起始数值,b 为差,c 为数值的数量,产生一个递增的等差数列,将这个数列以列表格式输出。

分析:定义 a、b、c 三个变量,分别获取用户输入的三个数字;用 for 循环控制 c 的循环次数,循环体中生成递增的等差数列元素,即用 a 减去 c 个 b,将生成的数列元素追加到新列表中,最后输出新列表。

编写代码如下。

```
a, b, c = eval(input("请输入三个数字,以逗号分隔:"))
ls = []
for i in range(c):
    ls.append(a + b * i)
print(ls)
```

7. 字符串替换

获得用户输入的一个数字,其中,数字字符(0~9)用对应的中文字符“〇 一二三四五六

七八九”替换，输出替换后的结果。

分析：对数字 0～9 用 for 循环依次判断，如果存在，则将其替换为中文字符，中文字符可以用字符串变量 *s* 存储，在替换时用字符串索引位置读取即可。

编写代码如下。

```
n = input()
s = "〇一二三四五六七八九"
for c in "0123456789":
    n = n.replace(c,s[int(c)])
print(n)
```

在本例中，变量 *c* 既是字符串的元素，同时也是字符串的索引位置。

8. 字符串分隔组合输出

程序接收用户输入的 5 个数，以逗号分隔。将这些数字按照输入顺序输出，每个数字占 10 个字符宽度，右对齐，所有数字显示在同一行。例如：

输入：

23,42,543,56,71

输出：

23 42 543 56 71

编写代码如下。

```
num = input().split(',')
for i in num:
    print('{:>10}'.format(i),end = '')
```

9. 正整数输出

程序接收用户输入的一个数字并判断是否为正整数，如果不是正整数，则显示“请输入正整数”并等待用户重新输入，直至输入正整数为止，并显示输出该正整数。例如：

输入：

请输入一个正整数：357

输出：

357

分析：用 while True 控制无限循环，并在循环体中搭配 break 语句，可以实现用户输入正确时退出循环；在循环体中，首先获取用户输入，然后对用户输入进行判断，如果用户输入的值大于 0 并且用户输入数据的类型为 int，则提示输入正确并 break，这里要用到 if…else 结构；如果不正确，则提示用户输入错误。

编写代码如下。

```
while True:
    try:
        a = eval(input('请输入一个正整数：'))
        if a > 0 and type(a) == int:
            print(a)
            break
```

```
        else:
            print("请输入正整数")
    except:
        print("请输入正整数")
```

10. 多行格式化输出

接收用户输入的一个小于 20 的正整数,在屏幕上逐行递增显示从 01 到该正整数,数字实现的宽度为 2,不足位置补 0,后面追加一个空格,然后显示“>”号,“>”号的个数等于行首数字。例如:

输入:

3

输出:

01 >

02 >>

03 >>>

分析:用 for 控制输出数字的个数,考查 format()格式的使用方法,n 个“>”可以使用字符的乘法实现。

执行代码如下。

```
n = input('请输入一个正整数:')
for i in range(int(n)):
    print('{:0>2} {}'.format(i+1, ">"*(i+1)))
```

11. 统计数字和字母数量

让用户输入一串数字和字母混合的数据,然后统计其中数字和字母的个数,显示在屏幕上。例如:

输入:

Fda243fdw3

输出:

数字个数:4,字母个数:6

分析:首先定义字符串变量获取用户输入,然后对字符串用 for 循环进行遍历,在循环体中,用字符串操作方法 isnumeric()判断是否是数字,用 isalpha()判断是否是字母,如果是,则给相应的统计个数的变量加 1,最后输出统计结果。

编写代码如下。

```
ns = input("请输入一串数据:")
dnum,dchr = 0,0                 #双变量赋值方式
for i in ns:
    if i.isnumeric():           #如果是数字字符
        dnum += 1
    elif i.isalpha():
        dchr += 1
    else:
        pass #空语句,为了保持程序结构的完整性,用于占位
```

```
print('数字个数:{},字母个数:{}'.format(dnum,dchr))
```

12. 生成通知书

已有列表 std = [['张三',90,87,95],['李四',83,80,87],['王五',73,57,55]],将 std 列表里的姓名和成绩与已经定义好的模板拼成一段话,显示在屏幕上。例如:

亲爱的张三,你的考试成绩是英语 90,数学 87,Python 语言 95,总成绩 272.特此通知。

分析:本例主要考查 for 循环对列表的遍历方法。

编写代码如下。

```
std = [['张三',90,87,95],['李四',83,80,87],['王五',73,57,55]]
modl = "亲爱的{}, 你的考试成绩是: 英语{}, 数学{}, Python 语言{}, 总成绩{}.特此通知."
for st in std:
    cnt = 0                    #总成绩初始值
    for i in range(3):         #循环三科成绩
        cnt += st[i+1]         #成绩求和
    print(modl.format(st[0],st[1],st[2],st[3],cnt))
```

13. 计算用户输入数字和并格式化输出

接收用户输入的一个大于 10 且小于 10 的 8 次方的十进制正整数,输出这个正整数各字符的和,以 25 位宽度、居中显示,采用等号“=”填充。

例如:

输入:

1357

输出:

==========16==========

分析:首先获取用户输入,然后用 for 循环将用户输入的每一位上的数字相加,最后按要求输出对应的格式。

编写代码如下。

```
s = input("请输入一个正整数: ")
cs = 0                         #求和初始值
for c in s:
    cs += int(c)               #字符转整型
print('{:=^25}'.format(cs))
```

14. 统计中文字符个数

接收用户输入的数据,该数据仅由字母和中文混合构成,无其他类型字符,统计并输出中文字符出现的个数。

分析:判断一个字符是否是中文,可以使用 Unicode 编码来判断,中文字符在'\u4e00'和'\u9fff'之间。

编写代码如下。

```
s = input("请输入中文和字母的组合: ")
count = 0                      #和的初始值
```

```
for c in s:
    if u'\u4e00' <= c <= '\u9fff':
        count += 1
print(count)
```

15. 输出最大值

接收用户输入的以英文逗号分隔的一组数据,其中,每个数据都是整数或浮点数,打印输出这组数据中的最大值。例如:

输入:

1,3,5,7,9,7,5,3,1

输出:

9

分析:首先获取用户输入,并通过 split(",")以逗号为分隔符将用户输入的数字分隔为列表。由于用户输入的数据类型为 str 类型,故需要用 for 循环将列表的每个元素用 eval()转换为数字,再重新追加到新列表中,最后用 max()输出新列表中的最大值。

编写代码如下。

```
s = input("请输入一组数据: ")
ls = s.split(",")
lt = []
for i in ls:
    lt.append(eval(i))
print(max(lt))
```

16. 生成指定字母

按照小写字母 a～z 顺序组成包含 26 个字母的字符表。其中,第一个字符 a 序号为 0,依次递增。程序获得用户输入的起始字母序号及连续输出字母的个数,分别记为变量 h 和 w,以逗号隔开,并根据字符表输出从起始字母序号 h 开始的连续 w 个字母。

示例如下(其中数据仅用于示意)。

输入:

0,3

输出:

abc

分析:首先,获取用户输入起始英文字母的序号和连续输出的个数,保存到变量 h 和 w 中,然后用 for 循环控制输出字符的个数,每次循环时,用 chr()将字母编码转换为对应字符,字母编码由 ord('a')与起始英文字母序号和输出编号之和计算,将每次循环生成的字符连接到新的字符串变量中,最后输出新字符串变量即可。

编写代码如下。

```
h,w = eval(input("请输入起始英文字母的序号和连续输出的个数,以逗号隔开:"))
cstr = ''
for i in range(w):
```

```
    c = chr(ord() + h + i)
    cstr += c
print(cstr)
```

17. 计算数字和并输出

获取一个由英文逗号、字母、汉字、数字字符组成的输入(以英文逗号隔开),计算该输入中所有数字的和,并输出。

示例如下。

输入:

1,海淀区,ab,56,3,中关村

输出:

数字和是:60

分析:本例中需要解决两个难点,一是将用户输入数据以逗号分隔转变为列表,需要使用 split()函数;二是用 for 循环将每个列表元素相加,在相加之前需要将列表元素转变为数字类型,可以使用内置函数 eval()。

编写代码如下。

```
myinput = input("请输入:")
ls = myinput.split(',')
s = 0
for c in ls:
    if c.strip(" ").isdigit():
        s += eval(c)
print("数字和是:" + str(s))
```

18. 列表向量积计算

计算两个列表 ls 和 lt 彼此对应元素乘积的和(即向量积)。

ls =[111,222,333,444,555,666,777,888,999]

lt = [999,777,555,333,111,888,666,444,222]

分析:用 for 循环分别遍历两个列表,将对应列表元素相乘,再加到变量 s 中。

编写代码如下。

```
ls = [111, 222, 333, 444, 555, 666, 777, 888, 999]
lt = [999, 777, 555, 333, 111, 888, 666, 444, 222]
s = 0
for i in range(len(ls)):
    s += ls[i] * lt[i]
print(s)
```

19. 录入成绩并输出最高分、最低分和平均分


视频讲解

从键盘输入小明学习的课程名称及考分等信息,信息之间采用空格分隔,每个课程一行,空行回车结束录入,示例格式如下。

数学 90

语文 95

英语 86

物理 84

生物 87

屏幕输出得分最高的课程及成绩、得分最低的课程及成绩,以及平均分(保留 2 位小数)。注意,其中逗号为英文逗号,格式如下。

最高分课程是语文 95,最低分课程是物理 84,平均分是 88.40

分析:

第 1 步,初始化变量。

首先,定义变量来获取用户输入的课程名和成绩,输入内容之间以空格分隔,然后用 split()将用户输入的数据转换为列表,并将初次录入的课程名和成绩分别赋值到最高分、最低分对应的课程和成绩变量中。

第 2 步,处理用户之后的输入。

使用 while 循环,以空格为终止符,在循环体中,首先获取用户每一次的输入,并对输入的字符串转换为列表处理,然后用 if 语句判断成绩大小,如果当次输入的成绩最小,则更新最小成绩与科目变量,如果最大,则更新最大成绩与科目变量。并对循环次数设定变量进行统计,将每次输入的成绩加到 s 中求和。

第 3 步,输出处理结果。

用 print()语句,使用 format()结构来输出最高分课程与成绩、最低分课程与成绩,以及平均分。

编写代码如下。

```
data = input('请输入课程名称 成绩:') #课程名 考分
ls = data.split()
min_score = int(ls[1])
min_name = ls[0]
max_score = int(ls[1])
max_name = ls[0]
n = 1
sum = int(ls[1])
while data:
    data = input('请再次输入课程名称 成绩:')
    if data == " ":
        break
    n += 1
    lt = data.split()
    if min_score > int(lt[1]):
        min_score = int(lt[1])
        min_name = lt[0]
    if max_score < int(lt[1]):
        max_score = int(lt[1])
        max_name = lt[0]
    sum += int(lt[1])
avg = sum/n
print("最高分课程是{} {}, 最低分课程是{} {}, 平均分是{:.2f}".format(max_name,max_score,
min_name,min_score,avg))
```

20. 统计数量

从键盘输入一组我国高校所对应的学校类型,以空格分隔,共一行,示例格式如下。

综合　理工　综合　综合　综合　师范　理工

统计各类型的数量，以数量从多到少的顺序在屏幕上输出类型及对应数量，以英文冒号分隔，每个类型一行，输出参考格式如下。

综合:4

理工:2

师范:1

分析：本题考查的主要知识点有三个。一是用户录入数据以空格分隔存储到列表中，使用 split()实现；二是用字典来统计数量的经典考法，要用到 d.get()方法；三是用字典统计好数量后，将其转换为列表，对列表进行排序，使用 ls.sort()以及匿名函数。

编写代码如下。

```
txt = input("请输入类型序列: ")
t = txt.split()
d = {}
for i in range(len(t)):
    d[t[i]] = d.get(t[i],0) + 1
ls = list(d.items())
ls.sort(key = lambda x:x[1], reverse = True)          #按照数量排序
for k in ls:
    print("{}:{}".format(k[0], k[1]))
```

21. 计算平均年龄

从键盘输入一组人员的姓名、性别、年龄等信息，信息之间采用空格分隔，每人一行，空行回车结束录入，示例格式如下。

张三 男 23

李四 女 21

王五 男 18

计算并输出这组人员的平均年龄(保留 2 位小数)和其中的男性人数，格式如下。

平均年龄是 20.67 男性人数是 2

分析：本例中需要首先对用户输入数据以空格分隔转换为列表对象，然后用 while 循环，分别对列表中的年龄相加，并统计个数，同时判断如果性别为“男”，另外统计其数量，最后计算并输出平均年龄和男性人数。

编写代码如下。

```
data = input() #姓名 性别 年龄
s,n,i = 0,0,0
while data:
    i = i + 1
    ls = data.split()
    s = s + int(ls[2])
    if ls[1] == '男':
        n = n + 1
    data = input()
    if data == " ":
        break
s = s/i
```

```
print("平均年龄是{:.2f} 男性人数是{}".format(s,n))
```

22. 数字三角阵列

获得用户输入的一个整数 n,输出一个 $n-1$ 行的数字三角形阵列。该阵列每行包含的整数序列为从该行序号开始到 $n-1$。例如,第 1 行包含 $1\sim n-1$ 的整数,第 i 行包含从 1 到 $n-1$ 的整数,数字之间用英文空格分隔。示例如下(其中数据仅用于示意)。

输入:

8

输出:

1 2 3 4 5 6 7

2 3 4 5 6 7

3 4 5 6 7

4 5 6 7

5 6 7

6 7

7

分析:本题要使用嵌套循环,首先看行号 i 的变量范围是 $1\sim n-1$,每一列循环变量 j 的范围为 $i\sim n-1$,输出相应的值即可。

编写代码如下。

```
n = eval(input("请输入一个整数:"))
for i in range(1,n):
    for j in range(i,n):
        print(j,end = ' ')
    print()
```

23. 数列求和

如果 n 为奇数,输出表达式 $1+1/3+1/5+\cdots+1/n$ 的值。如果 n 为偶数,输出表达式 $1/2+1/4+1/6+\cdots+1/n$ 的值。输出结果保留 2 位小数。

输入:

4

输出:

0.75

分析:本例主要考点在于循环的次数与表达式的推导。奇偶数的判断:用 $n\%2==0$ 可判断是偶数,否则为奇数。循环的 range()中,偶数的分母范围为 rang(2,n+1,2),奇数的分母范围为 rang(1,n+1,2),循环体中表达式为 s+=1/i。

编写代码如下。

```
n = eval(input("请输入一个整数:"))
s = 0
if n % 2 == 0:
    for i in range(2,n + 1,2):
```

```
        s += 1/i
else:
    for i in range(1,n + 1,2):
        s += 1/i
print(f'{s:.2f}')
```

24. 字符串拼接

用户按照列表格式输入数据，将用户输入的列表中属于字符串类型的元素连接成一个整字符串，并打印输出。

输入：

[123,"Python",98,"等级考试"]

输出：

Python 等级考试

分析：本例中，主要考点是从用户输入的列表中提取出字符串类型，并将其连接起来输出。可以用 type()来判断列表元素的类型是否是字符串，用字符串的加法来连接字符串。

编写代码如下。

```
ls = eval(input())
s = ""
for item in ls:
    if type(item) == type("香山"):
        s += item
print(s)
```

25. 统计字符串

获取用户输入，要求输入不带数字的文本，并统计字符串的长度。

分析：用 while True 来控制用户的多次输入，搭配 break 来退出循环。在循环体中，首先获取用户输入，然后用 for 循环遍历字符串，如果某字符串在 0～9，那么将 i 的值由初始的 0 加 1，这样可以作为出现数字的标记，在对字符串循环结束后，如果 i 的值仍然为 0，那么表明用户输入内容中没有出现数字，则可以退出 while 循环，否则进入下一次循环，最后输出结果。

编写代码如下。

```
while True:
    s = input("请输入不带数字的文本:")
    i = 0
    for p in s:
        if "0"<=p<="9":
            i = i + 1
    if i == 0:
        break
print(len(s))
```

3.5　异常处理

程序设计要考虑各种可能出现的错误，即使程序语法是正确的，运行时也可能会报错，

影响整个程序的运行。因此 Python 设计了强大的异常处理机制,对程序执行过程中出现的异常进行捕获并处理,使程序能够继续执行下去。

3.5.1 异常类型

Python 提供了强大的异常类 Exception,可以向用户准确反馈出错信息。常见的异常或错误名称如表 3-3 所示。

表 3-3 Python 常见的异常或错误名称

异常或错误名称	功 能 描 述	异常或错误名称	功 能 描 述
BaseException	所有异常的基类	NameError	未声明/初始化对象(没有属性)
SystemExit	解释器请求退出	UnboundLocalError	访问未初始化的本地变量
KeyboardInterrupt	用户中断执行(通常是按 Ctrl+C 组合键)	ReferenceError	弱引用(Weak Reference)试图访问已经被垃圾回收了的对象
Exception	常规错误的基类	RuntimeError	一般的运行时错误
StopIteration	迭代器没有更多的值	NotImplementedError	尚未实现的方法
GeneratorExit	生成器(generator)发生异常来通知退出	SyntaxError	Python 语法错误
StandardError	所有的内建标准异常的基类	IndentationError	缩进错误
ArithmeticError	所有数值计算错误的基类	TabError	Tab 和空格混用
FloatingPointError	浮点计算错误	SystemError	一般的解释器系统错误
OverflowError	数值运算超出最大限制	TypeError	对类型无效的操作
ZeroDivisionError	除(或取模)零(所有数据类型)	ValueError	传入无效的参数
AssertionError	断言语句失败	UnicodeError	Unicode 相关的错误
AttributeError	对象没有这个属性	UnicodeDecodeError	Unicode 解码时的错误
EOFError	没有内建输入,到达 EOF 标记	UnicodeEncodeError	Unicode 编码时错误
EnvironmentError	操作系统错误的基类	UnicodeTranslateError	Unicode 转换时错误
IOError	输入/输出操作失败	Warning	警告的基类
OSError	操作系统错误	DepreciationWarning	关于被弃用的特征的警告
WindowsError	系统调用失败	FutureWarning	关于构造将来语义会有改变的警告
ImportError	导入模块/对象失败	OverflowWarning	旧的关于自动提升为长整型(long)的警告
LookupError	无效数据查询的基类	PendingDeprecation Warning	关于特性将会被废弃的警告
IndexError	序列中没有此索引(index)	RuntimeWarning	可疑的运行时行为(Runtime Behavior)的警告
KeyError	映射中没有这个键	SyntaxWarning	可疑的语法的警告
UserWarning	用户代码生成的警告	MemoryError	内存溢出错误(对于 Python 解释器不是致命的)

3.5.2　异常情况处理

在程序设计中对各种可以预见的异常情况进行处理称为异常处理。合理恰当地使用异常处理可以使程序更加健壮，具有更强的容错性和更好的用户体验，不会因为用户不小心的错误输入或其他原因而造成程序运行终止。Python 中异常处理是通过一些特殊结构的语句实现的，主要有以下几种结构形式。

1. try…except…结构

语法结构：

```
try:
        代码块 1      # 此处放置可能引起异常的语句
except Exception [as reason]:
        代码块 2      # 如果 try 中代码块不能正常执行，则执行此处代码块进行异常处理
```

例 3-24：计算 5 除以 x 的值，其中，x 需要用户输入。如果用户输入 0，会报错导致程序不能继续运行，使用 try…except 异常处理来避免程序错误。


视频讲解

```
a = input("请输入除数:")
try:
    x = int(a)
    print(5/x)
except Exception:
    print("输入数值有误")
```

提示：except 后面可以跟 Exception，也可以跟具体的错误类型，如本题也可写为 except ZeroDivisionError，也可以省略不写，即只有 except。

执行结果：

```
请输入除数:6
0.8333333333333334
请输入除数:0
输入数值有误
请输入除数:a
输入数值有误
```

2. try…except…else…结构

语法结构：

```
try:
        代码块 1      # 此处放置可能引起异常的语句
except Exception [as reason]:
        代码块 2      # 如果 try 中代码块不能正常执行，则执行此处代码块进行异常处理
else:
        代码块 3      # 如果 try 中的代码没有引发异常，则在执行完 try 中的代码后，继续执行此处
#代码块
```

例 3-25：计算 5 除以 x 的值，其中，x 需要用户输入。如果用户输入 0，会报错导致程序不能继续运行，使用 try…except…else…异常处理来避免程序错误。

```
a = input("请输入除数:")
try:
    x = int(a)
    print(5/x)
except ZeroDivisionError:
    print("被零除")
else:
    print("正确")
```

执行结果:

```
请输入除数:6
0.8333333333333334
正确
请输入除数:0
被零除
请输入除数:a
Traceback (most recent call last):
  File "d:\PycharmProjects\pythonProject4\main.py", line 3, in <module>
    x = int(a)
        ^^^^^^
ValueError: invalid literal for int() with base 10: 'a'
```

提示:else 后的代码块只有当 try 语句未出现异常时才执行,一旦执行了 except 语句,else 中的代码块是不被执行的。本例中输入字母"a"时程序报错,原因在于未能设置相应的错误异常处理,这就需要了解多个 except 的结构。

3. 带有 try…多个 except…的结构

如果上例用户在输入时,输入的是非数值,使用 except ZeroDivisionError 排查异常可能还不够,还会出现数值类型的错误,这时需要用多个 except…结构,其语法结构为

```
try:
      代码块 1            #可能引发异常的代码块
except Exception1:        # Exception1 是可能出现的一种异常
      代码块 2            #处理 Exception1 异常的语句
except Exception2:        # Exception2 是可能出现的另一种异常
      代码块 3            #处理 Exception2 异常的语句
[else:]                   #可以没有 else 子句块
代码块 4                  #在以上所有 Exception 异常都未出现时,执行此处代码块
```

例 3-26:计算 5 除以 x 的值,其中,x 需要用户输入。如果用户输入 0,会报错导致程序不能继续运行,使用 try…多个 except…异常处理来避免程序错误。

```
a = input("请输入除数:")
try:
    x = int(a)
    print(5/x)
except ZeroDivisionError:
    print("被零除")
except ValueError:
    print("输入的不是数字")
else:
    print("正确")
```

执行结果如下：

```
请输入除数:6
0.8333333333333334
正确
请输入除数:0
被零除
请输入除数:a
输入的不是数字
请输入除数:python
输入的不是数字
```

4. try…except…finally…结构

语法结构：

```
try:
      代码块 1                    #可能引发异常的代码块
except Exception [as reason]:    #此语句也可直接写成 except:
      代码块 2                    #用来处理异常的代码块
finally
      代码块 3                    #无论异常是否发生此处代码块都会执行
```

例 3-27：计算 5 除以 x 的值，其中，x 需要用户输入。如果用户输入 0，会报错导致程序不能继续运行，使用 try…except…finally…异常处理来避免程序错误。

```
a = input("请输入除数:")
try:
    x = int(a)
    print(5/x)
except:
    print("输入错误")
finally:
    print("下一步")
```

执行结果：

```
请输入除数:6
0.8333333333333334
下一步
请输入除数:0
输入错误
下一步
请输入除数:a
输入错误
下一步
请输入除数:python
输入错误
下一步
```

提示：在本例中，finally 与上文的 else 不同，else 语句仅当 try 语句正常执行时才执行，而 finally 不论执行的是 try 还是 except 语句，均要执行 finally 语句。

在上述 4 种异常处理的方法中，如果不必知道具体的错误或异常类型，最常用的是第 1 种 try…except 语句。

小结

本章内容思维导图如图 3-9 所示。

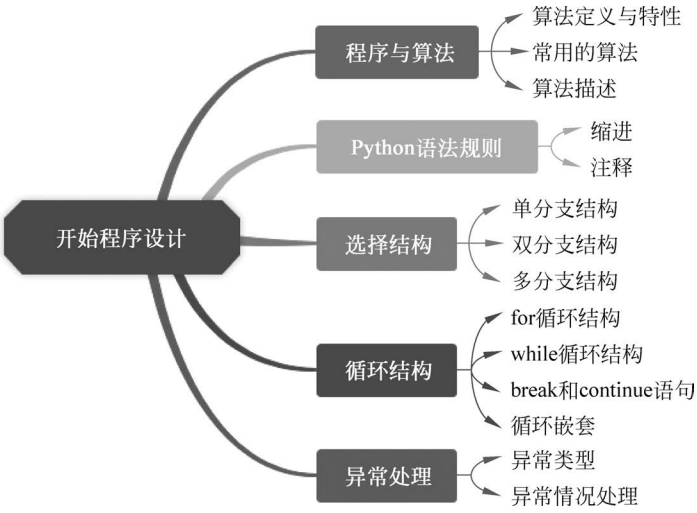


图 3-9 第 3 章内容思维导图

习题

一、选择题

1. 以下代码的执行结果是(　　)。

```
a = eval("12 + 3")
if type(a) == type(123):
    print("整数类型")
elif type(a) == type("123"):
    print("字符串类型")
else:
    print("其他类型")
```

A. 字符串类型　　B. 其他类型　　C. 代码执行错误提示　　D. 整数类型

2. 以下关于 Python 分支的说法中,错误的是(　　)。

A. Python 分支结构使用保留字 if、elif 和 else 来实现,每个 if 后面必须有 elif 或 else

B. if…else 结构是可以嵌套的

C. if 语句会判断 if 后面的逻辑表达式,当表达式为真时,执行 if 后续的语句块

D. 缩进是 Python 分支语句的语法部分,缩进不正确会影响分支功能

3. 以下关于 Python 循环结构的说法中,错误的是(　　)。

A. while 循环使用 break 保留字能够跳出所在层循环体

B. while 循环可以使用保留字 break 和 continue

C. while 循环也叫遍历循环，用来遍历序列类型中的元素，默认提取每个元素并执行一次循环体

D. while 循环使用 pass 语句，则什么事也不做，只是空的占位语句

4. 以下关于程序控制结构的说法，错误的是(　　)。

A. Python 里能用分支结构写出循环的算法

B. 二分支结构可以嵌套组合形成多分支结构

C. 分支结构包括单分支结构、二分支结构和多分支结构

D. 程序由三种基本结构组成

5. 以下代码的输出结果是(　　)。

```
x = 10
while x:
    x -= 1
    if x % 2:
        print(x,end = '')
    else:
        pass
```

A. 86420　　B. 975311　　C. 97531　　D. 864200

6. 以下代码的输出结果是(　　)。

```
for i in range(3):
    for s in "abcd":
        if s == "c":
            break
    print(s,end = "")
```

A. ababab　　B. abcabcabc　　C. aaabbb　　D. aaabbbccc

二、操作题

1. 从键盘输入一个正整数，判断奇偶数，然后输出结果(要求用 input 函数从键盘输入)。

2. 使用三种方法写出求 1＋2＋3＋…＋100 的和。

3. 编写程序，求 1～100 中所有偶数的和。

4. 设计一个用户登录程序。

要求：设定用户账户名为 root，密码是 12345。判断用户名和密码是否正确。为了防止暴力破解，登录仅有三次机会，如果超过三次，程序报错并结束。

5. 获取 100 以内的质数。质数又称为素数，指在一个大于 1 的自然数中，除了 1 和此整数自身外，不能被其他自然数整除的数。