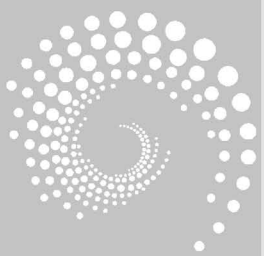
第3章

Python控制语句

CHAPTER 3



对于Python程序中的执行语句，默认时是按照书写顺序依次执行的，这时我们说这样的语句是顺序结构的。但是，仅有顺序结构还是不够的，因为有时候需要根据特定的情况，有选择地执行某些语句，这时就需要一种选择结构的语句。另外，有时候还可以在给定条件下往复执行某些语句，这时称这些语句是循环结构的。有了这三种基本的结构，就能够构建任意复杂的程序了。

3.1　选择结构


视频讲解

三种基本程序结构中的选择结构，可用 if 语句、if…else 语句和 if…elif…else 语句实现。

3.1.1　if 语句

Python 的 if 语句的功能与其他语言的非常相似，都是用来判定给出的条件是否满足，然后根据判断的结果（即真或假）决定是否执行给出的操作。if 语句是一种单选结构，它选择的是做与不做。它由三部分组成：关键字 if 本身、测试条件真假的表达式（简称为条件表达式）和表达式结果为真（即表达式的值为非零）时要执行的代码。if 语句的语法形式如下。

```
if 表达式:
    语句 1
```

if 语句的流程示意图如图 3-1 所示。

if 语句的表达式用于判断条件，可以用>（大于）、<（小于）、==（等于）、>=（大于或等于）、<=（小于或等于）来表示其关系。

现在用一个示例程序来演示一下 if 语句的用法。程序很简单，只要用户输入一个整数，如果这个数字大于 6，那么就输出一行字符串；否则，直接退出程序。代码如下。

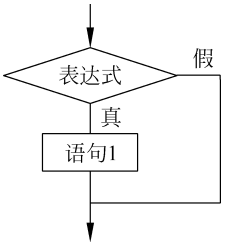


图 3-1　if 语句的流程示意图

```
#比较输入的整数是否大于 6
a = input("请输入一个整数: ")                    #取得一个字符串
a = int(a)                                       #将字符串转换为整数
if a > 6:
    print(a, "大于 6")
```

通常，一个程序都会有输入/输出，这样可以与用户进行交互。用户输入一些信息，计算机会对输入的内容进行一些适当的操作，然后再输出给用户想要的结果。Python 的输入/输出，可以用 input 进行输入，print 进行输出，这些都是简单的控制台输入/输出，复杂的有处理文件等。

3.1.2　if…else 语句

上面的 if 语句是一种单选结构，也就是说，如果条件为真（即表达式的值为非零），那么执行指定的操作；否则就会跳过该操作。而 if…else 语句是一种双选结构，在两种备选行动中选择一个。if…else 语句由 5 部分组成：关键字 if、测试条件真假的表达式、表达式结果为真（即表达式的值为非零）时要执行的代码，以及关键字 else 和表达式结果为假（即表达式的值为零）时要执行的代码。if…else 语句的语法形式如下。

```
if 表达式:
    语句 1
```

```
else:
    语句 2
```

if…else 语句的流程示意图如图 3-2 所示。

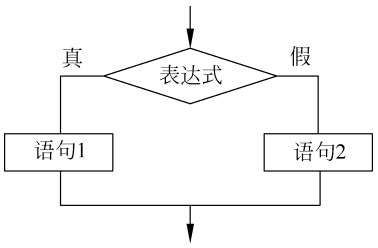


图 3-2 if…else 语句的流程示意图

下面对上面的示例程序进行修改,以演示 if…else 语句的使用方法。程序很简单,只要用户输入一个整数,如果这个数字大于 6,那么就输出一行信息,指出输入的数字大于 6;否则,输出另一行字符串,指出输入的数字小于或等于 6。代码如下。

```
a = input("请输入一个整数: ")                    #取得一个字符串
a = int(a)                                       #将字符串转换为整数
if a > 6:
    print(a, "大于 6")
else:
    print(a, "小于或等于 6")
```

再例如,输入一个年份,判断是否为闰年。闰年的年份必须满足以下两个条件之一。

(1) 能被 4 整除,但不能被 100 整除的年份都是闰年。

(2) 能被 400 整除的年份都是闰年。

设变量 year 表示年份,判断 year 是否满足条件。

条件(1)的逻辑表达式是:year%4==0 and year%100!=0。

条件(2)的逻辑表达式是:year%400==0。

两者取“或”,即得到判断闰年的逻辑表达式为

```
(year % 4 == 0 and year % 100!= 0) or year % 400 == 0
```

程序如下。

```
year = int(input('输入年份:'))                    #input()获取的是字符串,所以需要转换成整型
if year % 4 == 0 and year % 100!= 0 or year % 400 == 0:   #注意运算符的优先级
    print(year, "是闰年")
else:
    print( year, "不是闰年")
```

【例 3-1】 任意输入三个数字,按从小到大的顺序输出。

分析:①先将 x 与 y 比较,把较小者放入 x 中,较大者放 y 中;②再将 x 与 z 比较,把较小者放入 x 中,较大者放 z 中,此时 x 为三者中的最小者;③最后将 y 与 z 比较,把较小者放入 y 中,较大者放 z 中,此时 x、y、z 已按由小到大的顺序排列。

```
x = int( input('x = ') )                    #输入 x
y = int( input('y = ') )                    #输入 y
z = int( input('z = ') )                    #输入 z
if x > y:
    x, y = y, x                             #x, y 互换
if x > z:
    x, z = z, x                             #x, z 互换
if y > z:
    y, z = z, y                             #y, z 互换
print(x, y, z)
```

假如 x,y,z 分别输入 1,4,3,以上代码执行输出结果如下。

```
x = 1 ↙   (输入 x 的值,↙表示回车)
y = 4 ↙   (输入 y 的值)
z = 3 ↙   (输入 z 的值)
1 3 4
```

其中,x,y=y,x 这种语句是同时赋值,将赋值号右侧的表达式依次赋给左侧的变量。例如,x,y=1,4 相当于 x=1; y=4 效果,从而可见 Python 语法多么简洁。

3.1.3　if…elif…else 语句

有时候需要在多组动作中选择一组执行,这时就会用到多选结构,对于 Python 语言来说就是 if…elif…else 语句。该语句可以利用一系列条件表达式进行检查,并在某个表达式为真的情况下执行相应的代码。需要注意的是,虽然 if…elif…else 语句的备选动作较多,但是有且只有一组动作被执行,该语句的语法形式如下。

```
if 表达式 1:
    语句 1
elif 表达式 2:
    语句 2
    …
elif 表达式 n:
    语句 n
else:
    语句 n + 1
```

注意,最后一个 elif 子句之后的 else 子句没有进行条件判断,它实际上处理跟前面所有条件都不匹配的情况,所以 else 子句必须放在最后。if…elif…else 语句的流程示意图如图 3-3 所示。

下面继续对上面的示例程序进行修改,以演示 if…elif…else 语句的使用方法。还是要用户输入一个整数,如果这个数字大于 6,那么就输出一行信息,指出输入的数字大于 6; 如果这个数字等于 6,则输出另一行字符串,指出输入的数字等于 6; 否则,指出输入的数字小于 6。具体的代码如下。

```
a = input("请输入一个整数: ")              #取得一个字符串
a = int(a)                                  #将字符串转换为整数
if a > 6:
    print(a, "大于 6")
```

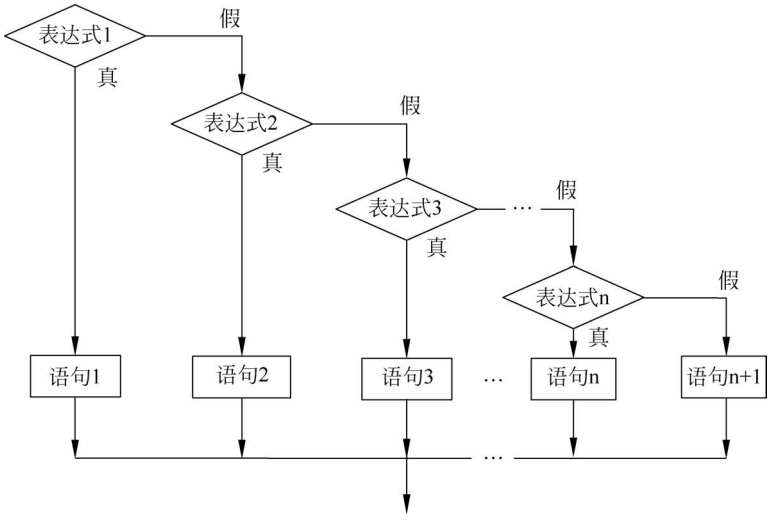


图 3-3 if…elif…else 语句的流程示意图

```
elif a == 6:
    print(a, "等于 6")
else:
    print(a, "小于 6")
```

【例 3-2】 输入学生的成绩 score,按分数输出其等级:score≥90 为优,90>score≥80 为良,80>score≥70 为中等,70>score≥60 为及格,score<60 为不及格。

```
score = int(input("请输入成绩"))          #int()转换字符串为整型
if score >= 90:
    print("优")
elif score >= 80:
    print("良")
elif score >= 70:
    print("中")
elif score >= 60:
    print("及格")
else:
    print("不及格")
```

说明:三种选择语句中,条件表达式都是必不可少的组成部分。当条件表达式的值为零时,表示条件为假;当条件表达式的值为非零时,表示条件为真。那么哪些表达式可以作为条件表达式呢?基本上,最常用的是关系表达式和逻辑表达式,例如:

```
if  a == x  and  b == y:
    print("a = x, b = y")
```

除此之外,条件表达式可以是任何数值类型表达式,甚至字符串也可以。

```
if  'a':              #'abc':也可以
    print("a = x, b = y")
```

另外,C 语言是用花括号{}来区分语句体的,但是 Python 的语句体是用缩进形式来表

示的，如果缩进不正确，会导致逻辑错误。

3.1.4　pass 语句

Python 提供了一个关键字 pass，类似于空语句，可以用在类和函数的定义中或者选择结构中。当暂时没有确定如何实现功能，或者为以后的软件升级预留空间，或者其他类型功能时，可以使用该关键字来“占位”。例如，下面的代码是合法的。

```
if a < b:
    pass                                    #什么操作也不做
else:
    z = a
class A:                                    #类的定义
    pass
def demo():                                 #函数的定义
    pass
```

3.2　循环结构

程序在一般情况下是按顺序执行的。编程语言提供了各种控制结构，允许更复杂的执行路径。循环语句允许执行一个语句或语句组多次，Python 提供了 for 循环和 while 循环（在 Python 中没有 do…while 循环）。

3.2.1　while 语句


视频讲解

Python 中 while 语句用于循环执行程序，即在某条件下，循环执行某段程序，以处理需要重复处理的相同任务。其基本形式为

```
while 判断条件:
    执行语句
```

执行语句可以是单个语句或语句块。判断条件可以是任何表达式，任何非零或非空（null）的值均为 True。当判断条件为 false 时，循环结束。while 语句的流程示意图如图 3-4 所示。

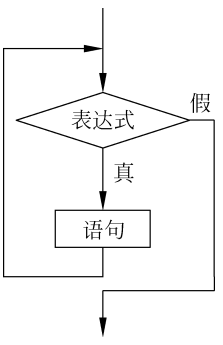


图 3-4　while 语句的流程示意图

同样需要注意冒号和缩进。另外，在 Python 中没有 do…while 循环。例如：

```
count = 0
while count < 5:
   print('The count is:', count)
   count = count + 1
print("Good bye!")
```

以上代码执行输出结果：

```
The count is: 0
The count is: 1
```

```
The count is: 2
The count is: 3
The count is: 4
Good bye!
```

此外,while 语句的“判断条件”还可以是个常值,表示循环必定成立。例如:

```
count = 0
while 1:                                    #判断条件是常值 1
   print('The count is:', count)
   count = count + 1
print("Good bye!" )
```

这样就形成无限循环,可以借助后面学习的 break 语句结束循环。

在 Python 中,可以在循环语句中使用 else 子句,else 中的语句会在循环正常执行完的情况下执行(不管是否执行循环体)。例如:

```
count = int(input())
while count < 5:
    print(count,"is less han 5")
    count = count + 1
else:
    print(count,"is not less than 5")
```

程序的一次运行结果如下。

```
8↙
8 is not less than 5
```

【例 3-3】 输入两个正整数,求它们的最大公约数。

分析:求最大公约数可以用“辗转相除法”,方法如下。

(1) 比较两个数,并使 m 大于 n。

(2) 将 m 作被除数,n 作除数,相除后余数为 r。

(3) 循环判断 r,若 r=0,则 n 为最大公约数,结束循环。若 r≠0,执行步骤 m←n,n←r;将 m 作被除数,n 作除数,相除后余数为 r。

```
num1 = int(input("输入第一个数字: "))    #用户输入两个数字
num2 = int(input("输入第二个数字: "))
m = num1
n = num2
if m < n:                                  #m,n 交换值
    t = m
    m = n
    n = t
r = m % n;
while  r!= 0:
    m = n;
    n = r
    r = m % n
print( num1,"和", num2,"的最大公约数为", n)
```

以上代码执行输出结果:

```
输入第一个数字: 36
输入第二个数字: 48
36 和 48 的最大公约数为 12
```


视频讲解

3.2.2　for 语句

for 语句可以遍历任何序列的项目，如一个列表、元组或者一个字符串。

1. for 循环的语法

for 循环的语法格式如下。

```
for 迭代变量 in 序列:
    循环体
```

for 语句的执行过程是：每次循环从序列中依次取出一个元素，存放于迭代变量，该元素值提供给循环体内的语句使用；直到所有元素取完为止，则结束循环。

例如，使用 for 循环把字符串中字符遍历出来：

```
for letter in 'Python':                    #第一个示例
    print( '当前字母 :', letter )
```

以上示例输出结果：

```
当前字母 : P
当前字母 : y
当前字母 : t
当前字母 : h
当前字母 : o
当前字母 : n
```

再如，使用 for 循环把列表中元素遍历出来。

```
fruits = ['banana', 'apple',  'mango']
for fruit in fruits:                       #第二个示例
   print( '元素 :', fruit)
print( "Good bye!" )
```

依次打印 fruits 的每一个元素，以上示例输出结果：

```
元素 : banana
元素 : apple
元素 : mango
Good bye!
```

【例 3-4】 计算 1～10 的整数之和，可以用一个 sum 变量做累加。

```
sum = 0
for x in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
    sum = sum + x
print(sum)
```

如果要计算 1～100 的整数之和,从 1 写到 100 有点困难,幸好 Python 提供了一个 range()内置函数,可以生成一个整数序列,再通过 list()函数可以转换为 list。

例如,range(0, 5)或 range(5)生成的序列是从 0 开始小于 5 的整数,不包括 5。

```
>>> list(range(5))
```

```
[0, 1, 2, 3, 4]
```

range(1, 101)就可以生成 1～100 的整数序列,计算 1～100 的整数之和如下。

```
sum = 0
for x in range(1,101):
    sum = sum + x
print(sum)
```

请自行运行上述代码,看看结果是不是当年高斯心算出的 5050。

2. 通过索引循环

对于列表,另外一种执行循环的遍历方式是通过索引(元素下标)。例如:

```
fruits = ['banana', 'apple',  'mango']
for i in range(len(fruits)):
   print( '当前水果 :', fruits[i] )
print("Good bye!")
```

以上示例输出结果:

```
当前水果 : banana
当前水果 : apple
当前水果 : mango
Good bye!
```

以上示例使用了内置函数 len()和 range(),函数 len()返回列表的长度,即元素的个数。通过索引 i 访问每个元素 fruits[i]。

3.2.3 continue 和 break 语句

break 语句在 while 循环和 for 循环中都可以使用,一般放在 if 选择结构中,一旦 break 语句被执行,将使得整个循环提前结束。

continue 语句的作用是终止当前循环,并忽略 continue 之后的语句,然后回到循环的顶端,提前进入下一次循环。

除非 break 语句让代码更简单或更清晰,否则不要轻易使用。

【例 3-5】 continue 和 break 用法示例。

```
#continue 和 break 用法示例
i = 1
while i < 10:
    i += 1
    if i % 2 > 0:                          #非偶数时跳过输出
        continue
```

```
    print(i)                                    #输出偶数 2、4、6、8、10

i = 1
while 1:                                        #循环条件为 1 必定成立
    print(i)                                    #输出 1～10
    i += 1
    if i > 10:                                  #当 i 大于 10 时跳出循环
        break
```

3.2.4　循环嵌套

Python 语言允许在一个循环体里面嵌入另一个循环。可以在循环体内嵌入其他的循环体，如在 while 循环中可以嵌入 for 循环；也可以在 for 循环中嵌入 while 循环。嵌套层次一般不超过三层，以保证可读性。

注意：

(1) 循环嵌套时，外层循环和内层循环之间是包含关系，即内层循环必须被完全包含在外层循环中。

(2) 当程序中出现循环嵌套时，程序每执行一次外层循环，其内层循环必须循环所有的次数(即内层循环结束)后，才能进入外层循环的下一次循环。

【例 3-6】 打印九九乘法表。

```
for i in range(1,10):
    for j in range(1,i+1):
        print(i,'*',j,'=',i*j,'\t',end = "")        #end = ""的作用是不换行
    print("")                                       #仅换行
```

以上代码执行输出结果如图 3-5 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 3-5　九九乘法表

【例 3-7】 使用嵌套循环输出 2～100 中的素数。

素数是除 1 和本身，不能被其他任何整数整除的整数。判断一个数 m 是否为素数，只要依次用 2，3，4，…，m－1 作除数去除 m，只要有一个能被整除，m 就不是素数。

```
m = int(input("请输入一个整数"))
j = 2
while j <= m - 1 :
   if m % j == 0: break                         #退出循环
   j = j + 1
if(j > m - 1) :
   print(m, "是素数")
else:
   print(m, "不是素数")
```

应用上述代码,对于一个非素数而言,判断过程往往很快可以结束。例如,判断 30009 时,因为该数能被 3 整除,所以只需判断 j=2,3 两种情况。而判断一个素数尤其是当该数较大时,例如判断 30011,就要从 j=2,3,4,…,一直判断到 30010 都不能被整除,才能得出其为素数的结论。实际上,只要从 2 判断到$\sqrt{m}$,若 m 不能被其中任何一个数整除,则 m 即为素数。

```
#找出 100 以内的所有素数
import math                                  #导入 math 数学模块
m = 2
while m < 100 :                              #外层循环
   j = 2
   while j <= math.sqrt(m) :                 #内层循环, math.sqrt()是求平方根
      if m % j == 0: break                   #退出内层循环
      j = j + 1
   if(j > math.sqrt(m)) :
      print(m, "是素数")
   m = m + 1
print("Good bye!")
```

【例 3-8】 使用嵌套循环输出如图 3-6 所示的金字塔图案。

分析:观察图形包含 8 行,因此外层循环执行 8 次;每行内容由两部分组成:空格和星号。假设第 1 行星号在第 10 列,则第 i 行空格的数量为 10−i,星号数量为 2i−1。

```
for i in range(1,9):                 #外层循环
    for j in range(0,10 - i):        #循环输出每行空格
        print(" ", end = "")
    for j in range(0,2 * i - 1):     #循环输出每行星号
        print("*", end = "")
    print("")                       #仅换行
```

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

图 3-6 金字塔图案

也可以通过以下代码实现。

```
for i in range(1,9):
    print(" " * (10 - i), "*" * (2 * i - 1))        #使用重复运算符输出每行空格、星号
```

3.2.5 列表生成式

列表生成式是 Python 内置的一类极其强大的生成 list 列表的表达式。如果要生成一个 list[1,2,3,4,5,6,7,8,9],可以用 range(1,10)。

```
>>> L = list(range(1, 10))                  #L 是[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

可是,如果要生成[1*1,2*2,3*3,…,10*10],可以使用循环。

```
>>> L = []
>>> for x in range(1 , 10):
        L.append(x * x)
>>> L
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

而列表生成式，可以只用一句代替以上的烦琐循环来完成上面的操作。

```
>>> [x * x for x in range(1 , 11)]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

列表生成式的书写格式：把要生成的元素 x * x 放到前面，后面跟上 for 循环。这样就可以把 list 创建出来。for 循环后面还可以加上 if 判断，例如，筛选出偶数的平方：

```
>>> [x * x for x in range(1 , 11) if x % 2 == 0]
[4, 16, 36, 64, 100]
```

再如，把一个 list 列表中所有的字符串变成小写形式。

```
>>> L = ['Hello', 'World', 'IBM', 'Apple']
>>> [s.lower() for s in L]
['hello', 'world', 'ibm', 'apple']
```

当然，列表生成式也可以使用两层循环，例如，生成'ABC'和'XYZ'中字母的全部组合。

```
>>> print( [m + n for m in 'ABC' for n in 'XYZ'] )
['AX', 'AY', 'AZ', 'BX', 'BY', 'BZ', 'CX', 'CY', 'CZ']
```

再例如，生成所有的扑克牌的列表。

```
>>> color = ["草花","方块","红桃","黑桃"]
>>> rank = ["A","1","2","3","4","5","6","7","8","9","10","J","Q","K"]
>>> print( [m + n for m in color for n in rank])
['草花 A', '草花 1', '草花 2', '草花 3', '草花 4', '草花 5', '草花 6', '草花 7', '草花 8', '草花 9', '草
花 10', '草花 J', '草花 Q', '草花 K', '方块 A', '方块 1', '方块 2', '方块 3', '方块 4', '方块 5', '方
块 6', '方块 7', '方块 8', '方块 9', '方块 10', '方块 J', '方块 Q', '方块 K', '红桃 A', '红桃 1', '红
桃 2', '红桃 3', '红桃 4', '红桃 5', '红桃 6', '红桃 7', '红桃 8', '红桃 9', '红桃 10', '红桃 J', '红
桃 Q', '红桃 K', '黑桃 A', '黑桃 1', '黑桃 2', '黑桃 3', '黑桃 4', '黑桃 5', '黑桃 6', '黑桃 7', '黑
桃 8', '黑桃 9', '黑桃 10', '黑桃 J', '黑桃 Q', '黑桃 K']
```

for 循环其实可以同时使用两个甚至多个变量，如字典(dict)的 items()可以同时迭代 key 和 value。

```
>>> d = {'x': 'A', 'y': 'B', 'z': 'C' }        #字典(dict)
>>> for  k, v  in d.items():
        print(k, '键 = ', v, endl = ';')
```

输出结果：

```
y 键 = B; x 键 = A; z 键 = C;
```

因此，列表生成式也可以使用两个变量来生成列表。

```
>>> d = {'x': 'A', 'y': 'B', 'z': 'C' }
>>> [ k + ' = ' + v  for k, v in d.items()]
['y = B', 'x = A', 'z = C']
```


视频讲解

3.3 应用案例——猜单词游戏

【案例】 应用案例——猜单词游戏。计算机随机产生一个单词,打乱字母顺序,供玩家去猜。

分析:游戏中需要随机产生单词以及随机数字,所以引入 random 模块随机数函数,其中,random.choice()可以从序列中随机选取元素。例如:

```
#创建单词序列
WORDS = ("python", "jumble", "easy", "difficult", "answer", "continue"
        , "phone", "position", "position", "game")
#从序列中随机挑出一个单词
word = random.choice(WORDS)
```

word 就是从单词序列中随机挑出一个单词。

游戏中随机挑出一个单词 word 后,如何把单词 word 的字母顺序打乱? 方法是随机从单词字符串中选择一个位置 position,把 position 位置那个字母加入乱序后单词,同时将原单词中 position 位置那个字母删去(通过连接 position 位置前字符串和其后字符串实现)。通过多次循环就可以产生新的乱序后单词。

```
while word:#word 不是空串循环
    #根据 word 长度,产生 word 的随机位置
    position = random.randrange(len(word))
    #将 position 位置字母组合到乱序后单词
    jumble += word[position]
    #通过切片,将 position 位置字母从原单词中删除
    word = word[:position] + word[(position + 1):]
print("乱序后单词:", jumble)
```

猜单词游戏程序代码如下。

```
#Word Jumble 猜单词游戏
import random
#创建单词序列
WORDS = ("python", "jumble", "easy", "difficult", "answer", "continue"
        , "phone", "position", "position", "game")
#开始游戏
print(
"""
     欢迎参加猜单词游戏
   把字母组合成一个正确的单词.
"""
)
iscontinue = "y"
while iscontinue == "y" or iscontinue == "Y":
    #从序列中随机挑出一个单词
    word = random.choice(WORDS)
    #一个用于判断玩家是否猜对的变量
    correct = word
    #创建乱序后单词
```

```
    jumble = ""
    while word:                                  #word 不是空串时循环
        #根据 word 长度,产生 word 的随机位置
        position = random.randrange(len(word))
        #将 position 位置字母组合到乱序后单词
        jumble += word[position]
        #通过切片,将 position 位置字母从原单词中删除
        word = word[:position] + word[(position + 1):]
    print("乱序后单词:", jumble)

    guess = input("\n 请你猜: ")
    while guess != correct and guess != "":
        print("对不起不正确.")
        guess = input("继续猜: ")

    if guess == correct:
        print("真棒,你猜对了!\n")
    iscontinue = input("\n\n 是否继续(Y/N): ")
```

运行结果：

```
     欢迎参加猜单词游戏
   把字母组合成一个正确的单词.
乱序后单词: yaes
请你猜: easy
真棒,你猜对了!
是否继续(Y/N): y
乱序后单词: diufctlfi
请你猜: difficutl
对不起不正确.
继续猜: difficult
真棒,你猜对了!
是否继续(Y/N): n
>>>
```

实验三　控制语句的使用

一、实验目的

通过本实验，了解程序控制的基本概念，学会使用 Python 的流程控制语句，掌握循环结构及循环嵌套结构的程序设计。

二、实验要求

(1) 掌握循环结构及循环嵌套结构的程序设计。
(2) 掌握 while 语句和 for 语句的使用方法。

三、实验内容与步骤

(1) 编写程序，求级数 1×2+2×3+3×4+4×5+…+n×(n+1)+…前 n 项的和。

分析：这是一个计数循环，每一个加数可以由数列的通项公式求得。

```
n = int(input("请输入一个正整数"))
sum = 0
for i in range(1,n + 1):
   a = i * (1 + i)
   sum = sum + a
print(sum)
```

(2) 编写程序。一个小球从 n 米高度落下，每次落地后反弹为原来高度的一半，再落下，球在 10 次落地后，未再弹起时，共经过了多少米？(要求 n 为偶数。)

分析：这是一个 9 次的计数循环求和，其中的加数(小球弹起高度)又应用了迭代算法(提示：需要将和初始化为 n)。

```
n = int(input("请输入一个正整数"))
s = n
for i in range(1,10):
   n = n/2
   s += 2 * n
print(s)
```

(3) 斐波那契数列因著名意大利数学家 Fibonacci 以兔子繁殖为例子而引入，故又称为“兔子数列”，其数值为 1、1、2、3、5、8、13、21、34、…。在数学上，这一数列以如下递推的方法定义：F(0)＝1，F(1)＝1，F(n)＝F(n－1)＋F(n－2)(n≥2)。

设计输出斐波那契数列的 Python 程序。

分析：首先调用 input 输入要打印的斐波那契数列的长度，然后把斐波那契数列存储于一个序列中，并逐个打印序列的元素。

```
#输入斐波那契数列的长度
FibonacciUptoNumer = int(input('Please input a Number : '))
n = FibonacciUptoNumer
fibs = [1, 1]
for number in range(n - 2):
    fibs.append(fibs[ - 2] + fibs[ - 1])
print(fibs)
```

(4) 设计删除一个列表里面的重复元素程序。

分析：首先调用 List.sort()对序列进行排序，然后调用 last＝List[－1]语句从后向前找出重复的元素，最后逐个打印非重复的元素。

此实验实现代码如下。

```
#List = eval(input("请输入列表"))  #键盘输入列表形式如[34,67,67,45,67,33,44,34,68,23]
List = [34,67,67,45,67,33,44,34,68,23]
List.sort()
last = List[ - 1]                          #最后一个元素
print(List)
print(range(len(List) - 2, - 1, - 1))
for i in range(len(List) - 2, - 1, - 1):   #从倒数第二个元素开始
    if last == List[i]:
```

```
            del List[i]
        else:
            last = List[i]
    print(List)
```

运行结果如下。

```
[23, 33, 34, 34, 44, 45, 67, 67, 67, 68]
range(8, -1, -1)
[23, 33, 34, 44, 45, 67, 68]
```

四、编程并上机调试

(1) 输入一个数 n,判断 1～n 范围里的所有完数。完数是指一个数恰好等于它的因子之和,如 6＝1＋2＋3,6 就是完数。

(2) 键盘上输入一个两位整数,求该数以内所有能被 3 整除的奇数个数。

(3) 输入一个成绩序列,输出各个成绩等级的人数。

数据：score＝[45,98,65,87,43,83,68,74,20,75,85,67,79,99]

等级：A(90～100),B(80～89),C(70～79),D(60～69),E(60 以下)

(4) 数字重复统计。

① 随机生成 1000 个整数,数字的范围为[20,100]。

② 升序输出所有不同的数字及其每个数字重复的次数。

习题

1. 输入一个整数 n,判断其能否同时被 5 和 7 整除,如能则输出“xx 能同时被 5 和 7 整除”,否则输出“xx 不能同时被 5 和 7 整除”。其中,“xx”为输入的具体数据。

2. 输入一个百分制的成绩,经判断后输出该成绩的对应等级。其中,90 分以上为“A”,80～89 分为“B”,70～79 分为“C”,60～69 分为“D”,60 分以下为“E”。

3. 某百货公司为了促销,采用购物打折的办法。消费 1000 元以上者,按九五折优惠;消费 2000 元以上者,按九折优惠;消费 3000 元以上者,按八五折优惠;消费 5000 元以上者,按八折优惠。编写程序,输入购物款数,计算并输出优惠价。

4. 编写一个求整数 n 阶乘(n!)的程序。

5. 编写程序,求 1!＋3!＋5!＋7!＋9!。

6. 编写程序,计算下列公式中 s 的值(n 是运行程序时输入的一个正整数)。

s＝1＋(1＋2)＋(1＋2＋3)＋…＋(1＋2＋3＋…＋n)

s＝12＋22＋32＋…＋(10×n＋2)

$s=1\times2-2\times3+3\times4-4\times5+\cdots+(-1)^{(n-1)}\times n\times(n+1)$

7. 分析“百马百瓦问题”。有 100 匹马驮 100 块瓦,大马驮 3 块,小马驮 2 块,两个马驹驮 1 块。在该问题中,大马、小马、马驹各有多少匹?

8. 有一个数列,其前 3 项分别为 1、2、3,从第 4 项开始,每项均为其相邻的前 3 项之和

的 1/2。在该数列中,从第几项开始会超过 1200?

9. 编写程序,找出 1 与 100 之间的全部“同构数”。“同构数”是指出现在其平方数右端的数。例如,5 的平方是 25,5 是 25 右端的数,则 5 就是同构数;25 也是一个同构数,因其平方是 625。

10. 分析猴子吃桃问题。猴子第一天摘下若干桃子,吃了一半后又多吃了一个,第二天早上将剩下的桃子吃了一半后又多吃了一个,以此类推,每天早上都吃前一天剩下的一半再加一个。猴子到第 10 天早上再吃时,发现只剩下一个桃子了。求猴子第一天共摘了多少个桃子?

11. 开发猜数字小游戏。计算机随机生成 100 以内的数字,如果玩家所猜数字过大或过小都会给出提示,直到猜中该数则显示“恭喜! 你猜对了”,同时统计玩家猜的次数。

12. 编写程序,求解数字重复统计问题,要求如下。

(1) 随机生成 1000 个整数,数字的范围为[20,100]。

(2) 升序输出所有的不同数字及其每个数字重复的次数。

13. 求每个学生的平均成绩,结果保留两位小数。

学生成绩:`s = {"Teddy":[100,90,90],"Sandy":[100,90,80],"Elmo":[90,90,80]}`

输出结果:`{'Teddy': 93.3, 'Sandy': 90, 'Elmo': 86.7}`

14. 现有一个字典存放学生学号和三门课程成绩:

```
dictScore = {"101":[67,88,45],"102":[97,68,85],"103":[98,97,95],"104":[67,48,45],"105":
[82,58,75],"106":[96,49,65]}
```

编写程序,要求返回每个学生的学号及其最高分。