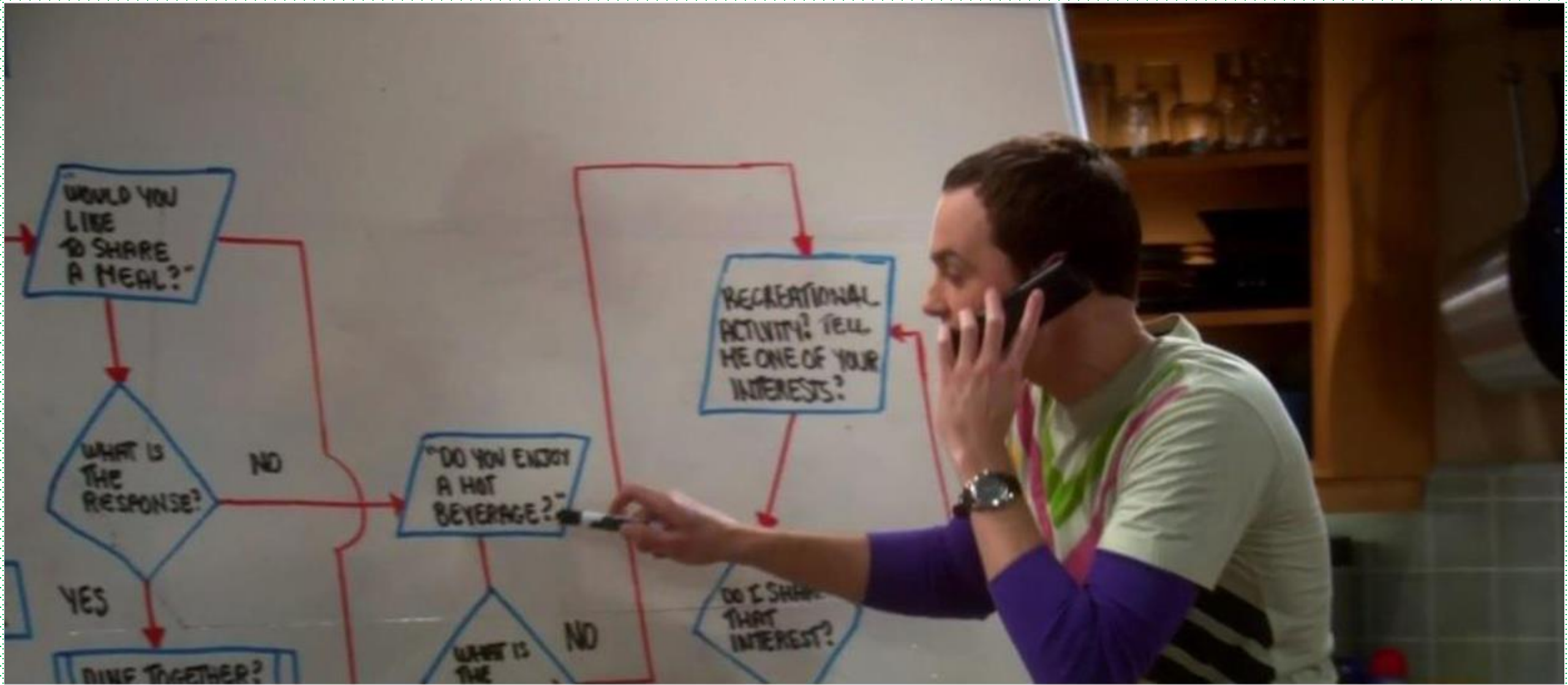
第4章 简单程序-控制流


WOULD YOU LIKE TO SHARE A MEAL?
WHAT IS THE RESPONSE?
NO
YES
DO YOU ENJOY A HOT BEVERAGE?
RECREATIONAL ACTIVITY? TELL ME ONE OF YOUR INTERESTS?
NO

控制结构概述（从字词句和段落）

- 有了合适的数据类型和数据结构之后，还要依赖于选择和循环结构来实现特定的业务逻辑

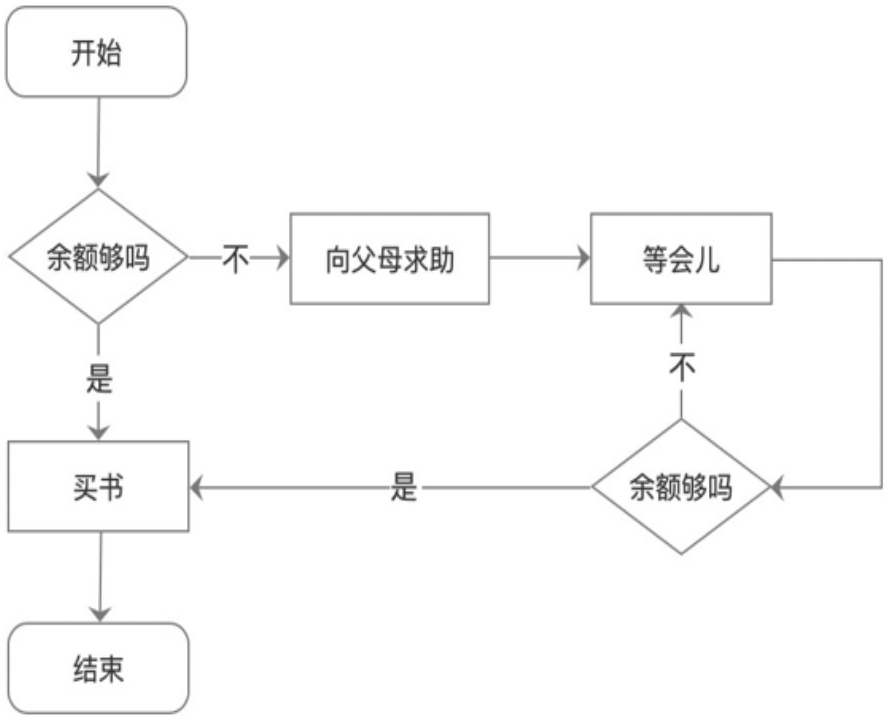
- 如果把数据看作"材料"，语句看作处理这些材料的"事务"。那么， 控制结构就是决定怎么组织这些事务的方式。

- 一个完整控制结构可以看作是一个大的“语句”，从这个角度来讲，程序中的多条“语句”是顺序执行的。

- 从现在开始，大部分情况要在py文件里写段落，而不能在idle里写了

控制结构概述

- 控制结构包括：顺序结构、分支结构和循环结构。



4.1 条件表达式

- 条件表达式的结果为一个布尔值， 即 True 或者 False。
- 在 Python 中， 所有对象都有一个固有的布尔真 / 假值。
- 在选择和循环结构中，条件表达式的值除了表中所列出来的值，Python解释器均认为与True等价。

Python 中的假值对象

对象类型	对象值
布尔值	False
null 类型	None
整型	0
浮点型	0.0
空字符串	"或""
空列表	[]
空元组	()
空字典	{}
空集合	set()

4.1 条件表达式

（1）关系运算符

Python中的关系运算符可以连续使用，这样不仅可以减少代码量，也比较符合人类的思维方式。

```
>>> print(1<2<3)                    #等价于1<2 and 2<3
True
>>> print(1<2>3)
False
>>> print(1<3>2)
True
```

4.1 条件表达式

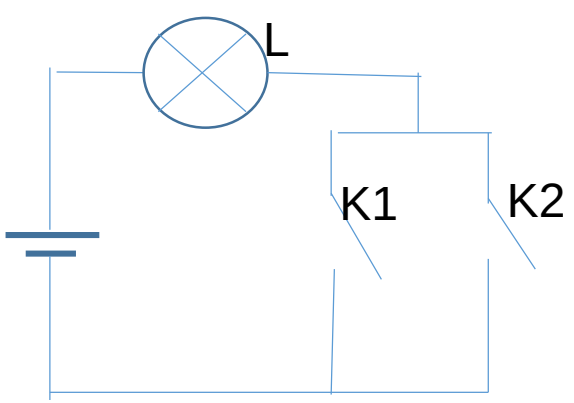
- 在Python语法中，条件表达式中不允许使用赋值运算符“=”，避免了误将关系运算符“==”写作赋值运算符“=”带来的麻烦。在条件表达式中使用赋值运算符“=”将抛出异常，提示语法错误。

```
>>> if a=3:                          #条件表达式中不允许使用赋值运算符
SyntaxError: invalid syntax          #能用 ==, !=
>>> if (a=3) and (b=4):
SyntaxError: invalid syntax
```

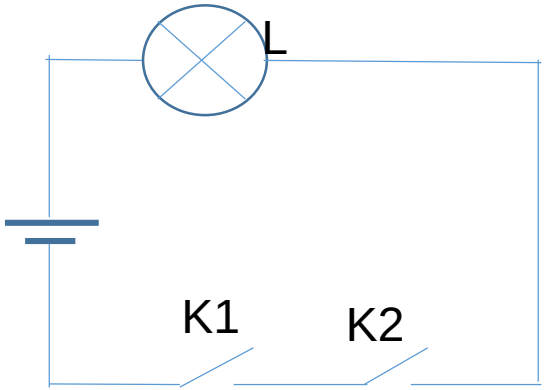
4.1 条件表达式

（2）逻辑运算符

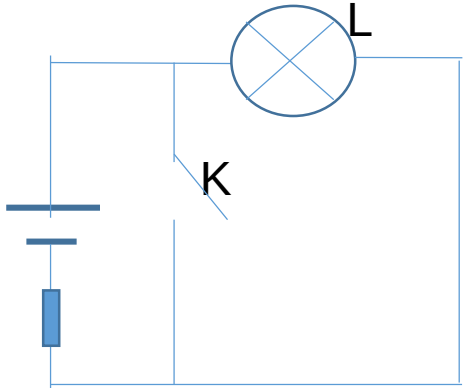
逻辑运算符and和or具有短路求值或惰性求值的特点，可能不会对所有表达式进行求值，而是只计算必须计算的表达式的值。



（1）or，并联电路



（2）and，串联电路



（3）not，短路

4.1 条件表达式

- 以“and”为例，对于表达式“表达式1 and 表达式2”而言，如果“表达式1”的值为“False”或其他等价值时，不论“表达式2”的值是什么，整个表达式的值都是“False”，丝毫不受“表达式2”的影响，因此“表达式2”不会被计算。
- 在设计包含多个条件的条件表达式时，如果能够大概预测不同条件失败的概率，并将多个条件根据“and”和“or”运算符的短路求值特性来组织顺序，可以大幅度提高程序运行效率。

4.1 条件表达式

```
>>> 3 and 5
5
>>> 3 or 5
3
>>> 0 and 5
0
>>> 0 or 5
5
>>> not 3
False
>>> not 0
True
```

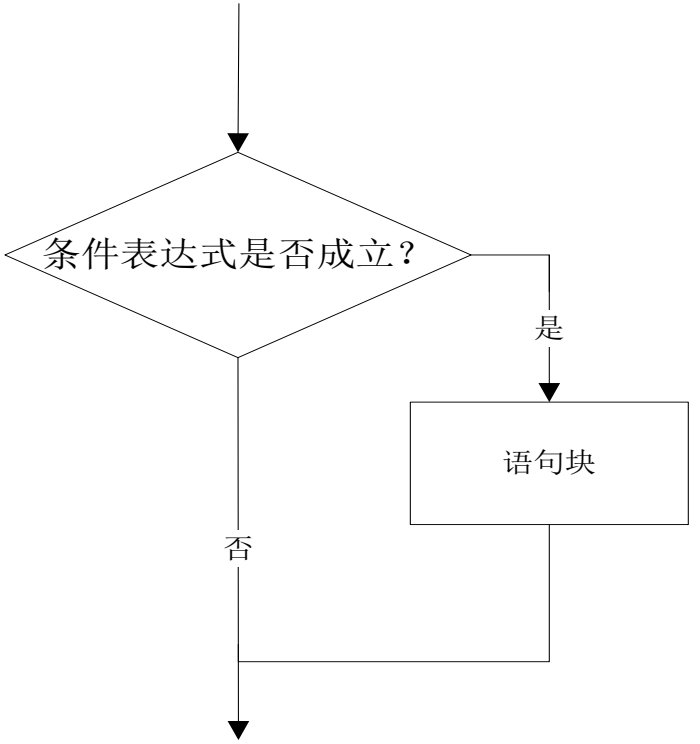
4.2 选择结构

- 常见的选择结构有单分支选择结构、双分支选择结构、多分支选择结构以及嵌套的分支结构，也可以构造跳转表来实现类似的逻辑。

- 循环结构和异常处理结构中也可以带有“else”子句，可以看作是特殊形式的选择结构。

4.2.1 单分支选择结构

```
if 表达式:
    语句块
```



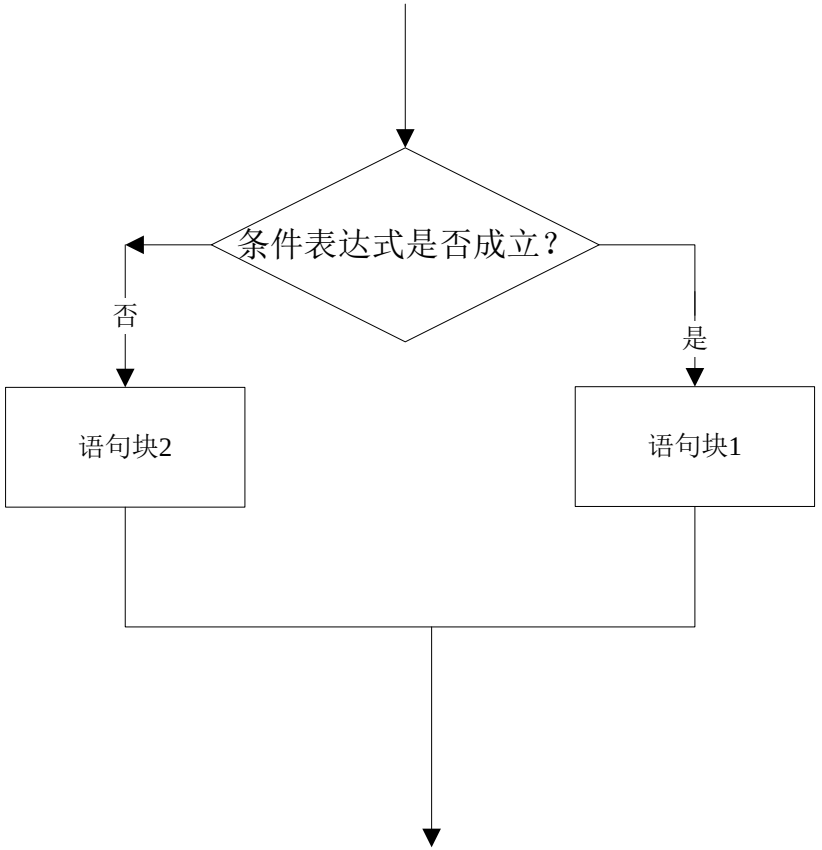

```
x = input('Input two number: a b')
a, b = map(int, x.split())
if a > b:
   a, b = b, a     #序列解包，交换两个变量的值
print(a, b)
```

- #i为list，比如left, right,ship

- if ('父亲' not in i) and ('母亲' in i) and ('儿子' in i) :
- 　　……
- if ('母亲' not in i) and ('父亲' in i) and ('女儿' in i) :
- 　　……
- if ('警察' not in i) and ('犯人' in i) and (len(i)!=1):
- 　　……

4.2.2 双分支选择结构

```
if 表达式:
    语句块1
else:
    语句块2
```




```
>>> chTest = ['1', '2', '3', '4', '5']
>>> if chTest:
    print(chTest)
else:
    print('Empty')

['1', '2', '3', '4', '5']
```

4.2.2 双分支选择结构

- 问题解决：鸡兔同笼。

```
jitu, tui = map(int, input('请输入鸡兔总数和腿总数：').split())
tu = (tui - jitu*2) / 2
if int(tu) == tu:
    print('鸡：{0},兔：{1}'.format(int(jitu-tu), int(tu)))
else:
    print('数据不正确，无解')
```

4.2.2 双分支选择结构

- Python还提供了一个三元运算符，并且在三元运算符构成的表达式中还可以嵌套三元运算符，可以实现与选择结构相似的效果。语法为

```
value1 if condition else value2
```

- 当条件表达式condition的值与True等价时，表达式的值为value1，否则表达式的值为value2。

```
>>> b = 6 if 5>13 else 9          #赋值运算符优先级非常低
>>> b
9
```

4.2.3 多分支选择结构

```
if 表达式1:
    语句块1
elif 表达式2:
    语句块2
elif 表达式3:
    语句块3
else:
    语句块4
```

其中，关键字elif是else if的缩写。

4.2.3 多分支选择结构

- **问题解决：**使用多分支选择结构将成绩从百分制变换到等级制。

```
score = input( ‘请输入score腿总数：')
if score > 100 or score < 0:
        return 'wrong score.must between 0 and 100.'
    elif score >= 90:
        return 'A'
    elif score >= 80:
        return 'B'
    elif score >= 70:
        return 'C'
    elif score >= 60:
        return 'D'
    else:
        return 'E'
```

4.2.4 选择结构的嵌套

```
if 表达式1:
    语句块1
    if 表达式2:
        语句块2
    else:
        语句块3
else:
    if 表达式4:
        语句块4
```

注意：缩进必须要正确并且一致。

4.2.4 选择结构的嵌套

- **问题解决：**使用嵌套选择结构将成绩从百分制变换到等级制。

```
def func(score):
    degree = 'DCBAAE'
    if score > 100 or score < 0:
        return 'wrong score.must between 0 and 100.'
    else:
        index = (score - 60) // 10
        if index >= 0:
            return degree[index]
        else:
            return degree[-1]
```

- https://www.bilibili.com/read/cv17521097/

- 条件选择异常处理
- 变量定义
  - 虽然python不指定变量类型，但是对于大规模生产环境来说，这个便利性最后带来的后果可能是致命的

```
17  local _gcd
18  _gcd = function (a, b)
19      if b == 0 then
20          return a
21      end
22
23      return _gcd(b, a % b)
24  end
25
```

4.3 循环结构

- Python主要有for循环和while循环两种形式的循环结构，多个循环可以嵌套使用，并且还经常和选择结构嵌套使用来实现复杂的业务逻辑。
  - while循环: 一般用于循环次数难以提前确定的情况，当然也可以用于循环次数确定的情况；
  - for循环: 一般用于循环次数可以提前确定的情况，尤其适用于枚举或遍历序列或迭代对象中元素的场合。

4.3.1 for循环与while循环

- 两种循环结构的语法形式分别为：

```
while 条件表达式:
    循环体
```

和

```
for 取值 in 序列或迭代对象:
    循环体
```

4.3.1 for循环与while循环

- **问题解决：**使用循环结构遍历并输出列表中的所有元素。

```
a_list = ['a', 'b', 'mpilgrim', 'z', 'example']
for i, v in enumerate(a_list):
    print('列表的第', i+1, '个元素是：', v)
```

4.3.1 for循环与while循环

- **问题解决：**输出1~100之间能被7整除但不能同时被5整除的所有整数。

```
for i in range(1, 101):
    if i%7==0 and i%5!=0:
        print(i)
```

4.3.1 for循环与while循环

- **问题解决：**使用嵌套的循环结构打印九九乘法表。

```
for i in range(1, 10):
    for j in range(1, i+1):
        print('{0}*{1}={2}'.format(i,j,i*j), end='   ')
    print()                          #打印空行
```

4.3.1 for循环与while循环

- **问题解决：**计算1+2+3+…+99+100的结果。

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

或直接计算：

```
>>> sum(range(1,101))
5050
```

4.3.2 break、continue、else语句

- 对于循环语句， 还有一个更加完整的表达形式， 即可以在循环语句中加入 break、continue和 else语句。其语法格式如下：

```
While 条件表达式1:
      语句块1
      if 条件表达式2:
            break
      if 条件表达式3:
            continue
      语句块2
else:
      语句块3
```

```
for 取值 in 序列或迭代对象:
      语句块1
      if 条件表达式2:
            break
      if 条件表达式3:
            continue
      语句块2
else:
      语句块3
```

- 执行break语句时， break语句所属层次的循环提前结束，不再执行循环体中剩余的语句。
- 执行continue 语句时，提前结束本次循环，忽略continue之后的所有语句，提前进入下一次循环。
- 当循环结束后，如果循环因为条件表达式不成立或序列遍历结束，则执行else子句所引导的代码块。如果循环是因为执行了break语句而导致循环提前结束则不会执行else中的语句。

4.3.2 break、continue、else语句

- **问题解决：**计算小于100的最大素数。

```
for n in range(100, 1, -1):
    if n%2 == 0:
        continue
    for i in range(3, int(n**0.5)+1, 2):
        if n%i == 0:
            #结束内循环
            break
    else:
        print(n)
        #结束外循环
        break
```

8人，又是8人

```
x=[left,right,ship]
    for i in x:
        if ('父亲' not in i) and ('母亲' in i) and ('儿子' in i) :
            ……
        if ('母亲' not in i) and ('父亲' in i) and ('女儿' in i) :
            ……
        if ('警察' not in i) and ('犯人' in i) and (len(i)!=1):
            ……
        if (i==ship):
            if '父亲' not in i and '母亲' not in i and '警察' not in i:
                ……
```

到此，大家觉得还差啥？

```
#定义场景结构
left, right, ship=['警察','犯人','父亲','儿子','儿子','母亲','女儿','女儿'],[],[]

#操作两岸和船的状态
ship.append(), left.pop(), right.append()

#假设i，j是记录每次登船的人
ship.append(left.pop(i))
ship.append(left.pop(j))

#如果满足两岸和划船的条件
 for k in ship:
    right.append(k)
```

```
# 遍历状态，确认符合规则
x=[left,right,ship]
   for i in x:
      if ('父亲' not in i) and ('母亲' in i) and ('儿子' in i) :
         ……
      if ('母亲' not in i) and ('父亲' in i) and ('女儿' in i) :
         ……
      if ('警察' not in i) and ('犯人' in i) and (len(i)!=1):
          ……
      if (i==ship):
         if '父亲' not in i and '母亲' not in i and '警察' not in i:
            ……
```

4.4 精彩案例赏析

- **示例4-1** 输入若干个成绩，求所有成绩的平均分。每输入一个成绩后询问是否继续输入下一个成绩，回答“yes”就继续输入下一个成绩，回答“no”就停止输入成绩。

4.4 精彩案例赏析

```
numbers = []                                    #使用列表存放临时数据
while True:
    x = input('请输入一个成绩：’)
    numbers.appdend(float(x))
while True:
        flag = input('继续输入吗？（yes/no）')
        if flag.lower() not in ('yes', 'no'): #限定用户输入内容必须为yes或no
            print('只能输入yes或no')
        else:
            break
    if flag.lower()=='no':
        break

print(sum(numbers)/len(numbers))
```

4.4 精彩案例赏析

- **示例4-2** 编写程序，判断今天是今年的第几天。

```
import time

date = time.localtime()                                    #获取当前日期时间
year, month, day = date[:3]
day_month = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
if year%400==0 or (year%4==0 and year%100!=0):    #判断是否为闰年
    day_month[1] = 29
if month==1:
    print(day)
else:
    print(sum(day_month[:month-1])+day)
```

4.4 精彩案例赏析

- **示例4-3** 编写代码，输出由星号*组成的菱形图案，并且可以灵活控制图案的大小。

```
n = input("Input an integer:")
n = int(n)
for i in range(n):
    print((' * '*i).center(n*3))
for i in range(n, 0, -1):
    print((' * '*i).center(n*3))
```

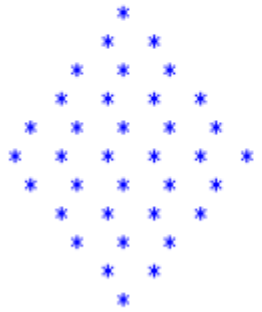


图 4-5　n=6 的运行效果

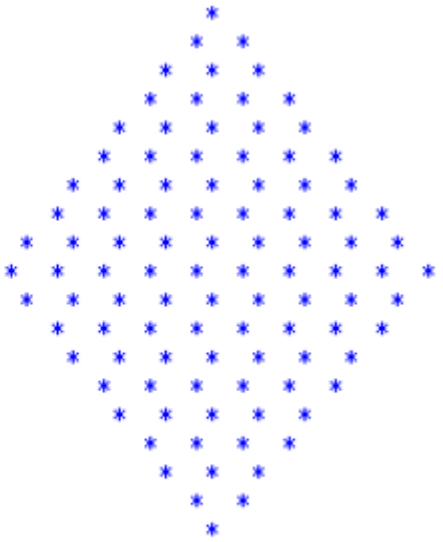


图 4-6　n=10 的运行效果

4.4 精彩案例赏析

- **示例4-4** 快速判断一个数是否为素数。

```
n = input("Input an integer:")
n = int(n)
if n == 2:
    print('Yes')
```

4.4 精彩案例赏析

```
#偶数必然不是素数
elif n%2 == 0:
    print('No')
else:
    #大于5的素数必然出现在6的倍数两侧
    #因为6x+2、6x+3、6x+4肯定不是素数，假设x为大于1的自然数
    m = n % 6
    if m!=1 and m!=5:
        print('No')
    else:
        for i in range(3, int(n**0.5)+1, 2):
            if n%i == 0:
                print('No')
                break
        else:
            print('Yes')
```

4.4 精彩案例赏析

- **示例4-5** 编写程序，计算组合数C(n,i)，即从n个元素中任选i个，有多少种选法。

根据组合数定义，需要计算3个数的阶乘，在很多编程语言中都很难直接使用整型变量表示大数的阶乘结果，虽然Python并不存在这个问题，但是计算大数的阶乘仍需要相当多的时间。本例提供另一种计算方法：以Cni(8,3)为例，按定义式展开如下，对于(5,8]区间的数，分子上出现一次而分母上没出现；(3,5]区间的数在分子、分母上各出现一次；[1,3]区间的数分子上出现一次而分母上出现两次。

$$C_8^3 = \frac{8!}{3!\times(8-3)!} = \frac{8\times7\times6\times\mathit{5}\times\mathit{4}\times\mathit{3}\times\mathit{2}\times\mathit{1}}{3\times2\times1\times\mathit{5}\times\mathit{4}\times\mathit{3}\times\mathit{2}\times\mathit{1}}$$

4.4 精彩案例赏析

```
def Cni2(n, i):
    if not (isinstance(n,int) and isinstance(i,int) and n>=i):
        print('n and i must be integers and n must be larger than or equal to i.')
        return
    result = 1
    Min, Max = sorted((i,n-i))
    for i in range(n,0,-1):
        if i>Max:
            result *= i
        elif i<=Min:
            result /= i
    return result

print(Cni2(6,2))
```

P87 排列组合函数combination

```
#定义场景结构
left, right, ship=['警察','犯人','父亲','儿子','儿子','母亲','女儿','女儿'],[],[]

#操作两岸和船的状态
ship.append(), left.pop(), right.append()

#假设i，j是记录每次登船的人
ship.append(left.pop(i))
ship.append(left.pop(j))

#如果满足两岸和划船的条件
 for k in ship:
    right.append(k)
```

```
# 遍历状态，确认符合规则
x=[left,right,ship]
   for i in x:
      if ('父亲' not in i) and ('母亲' in i) and ('儿子' in i) :
         ……
      if ('母亲' not in i) and ('父亲' in i) and ('女儿' in i) :
         ……
      if ('警察' not in i) and ('犯人' in i) and (len(i)!=1):
          ……
      if (i==ship):
         if '父亲' not in i and '母亲' not in i and '警察' not in i:
            ……

#排列组合
import itertools  # 或者 from itertools import combinations
a=list(itertools.combinations(left,2))
```