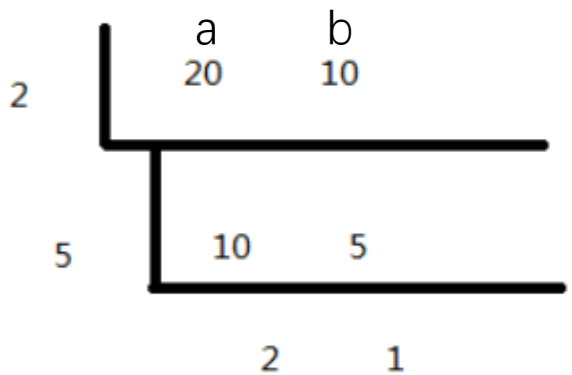
# 第三周上机课

2023-5

# 猜数游戏-在程序中预设一个1-10之间的数字（包括10），让用户通过键盘输入所猜的数，如果大于预设的数，显示“遗憾，太大了”；小于预设的数，显示“遗憾，太小了”，如此循环，直至猜中该数，显示“预测N次，你猜中了”，N为用户输入数字的次数

```
from random import randint
num=randint(1,10)
tim=0
while 1:
    try:
        putnum=eval(input("请输入您猜测的数字："))
        tim+=1
        if putnum > num:
            print("遗憾！太大了")
        elif putnum <num:
            print("遗憾！太小了")
        elif putnum==num:
            print("预测{}次，你猜中了！".format(tim))
            break
    except:
        print("输入有误！")
```

最大公约数和最小公倍数求解，常用的方法是短除法进行因式分解，然后最大公约数是所有公共因子的乘积，
最小公倍数是所有因子的乘积。




最大公约数：2×5=10

最小公倍数：2×5×2×1=20

m 和 n求最大公因数（假设m大于n），先用m除以n，如果余数 r 为 0，则 n 就是最大公因数，否则，将 n 给了 m，将 r 给了 n，再用 m 除以 n，如果余数 r 为 0，则n为最大公因数，否则重复执行上述操作，直至 r 为 0，此时的 n 就是 m 和 n 的最大公因数。

最大公约数计算。从键盘输入两个整数，编写程序求这两个整数的最大公约数和最小公倍数。（求最大公约数使用辗转相除法，求最小公倍数用两数的积除以最大公约数）

- a=int(input(“请输入数字："))
- b=int(input(“请输入数字："))
- # 先给两数排序，保证大数除以小数
- m=max(a,b)
- n=min(a,b)
- t=m%n
- while t!=0:
- 　　m,n=n,t #  每个除式的m、n都是上一个式子的n和余数
- 　　t=m%n  # 更新余数
- print("{}和{}的最大公约数为{},最大公倍数为{}".format(a,b,n,a*b/n))

# 通过函数的调用来实现

```
def Gongyueshu(a,b):
    # 比大小，保证大数除以小数
    if a<b:
        a,b=b,a
    # 判断是否能整除，若能整除，直接返回被除数
    if a%b==0:
        return b
    # 若不能整除，则返回函数gongyueshu，参数做相应变化
    else:
        return gongyueshu (b,a%b)
a=int(input("请输入第一个数字："))
b=int(input("请输入第二个："))
gcd=Gongyueshu (a,b)
print(f"{a}和{b}的最大公约数为{gcd}")
```

## 【例】判断坐标点象限

- 已知坐标点(x,y)，判断其所在的象限

```
x = int(input("请输入x坐标： "))
y = int(input("请输入y坐标： "))
if (x == 0 and y == 0): print("位于原点")
elif (x == 0): print("位于y轴")
elif (y == 0): print("位于x轴")
elif (x > 0 and y > 0): print("位于第一象限")
elif (x < 0 and y > 0): print("位于第二象限")
elif (x < 0 and y < 0): print("位于第三象限")
else: print("位于第四象限")
```

程序运行结果如下：

请输入 x 坐标：1

请输入 y 坐标：2

位于第一象限

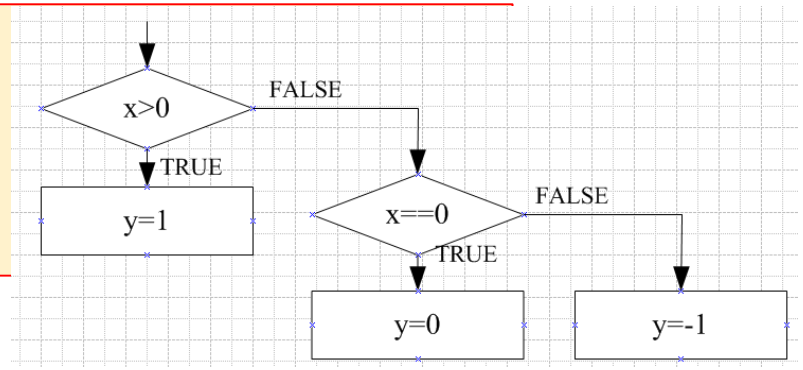
# if语句的嵌套

- 【例】计算分段函数：

$$y=\begin{cases}1 & x>0\\ 0 & x=0\\ -1 & x<0\end{cases}$$

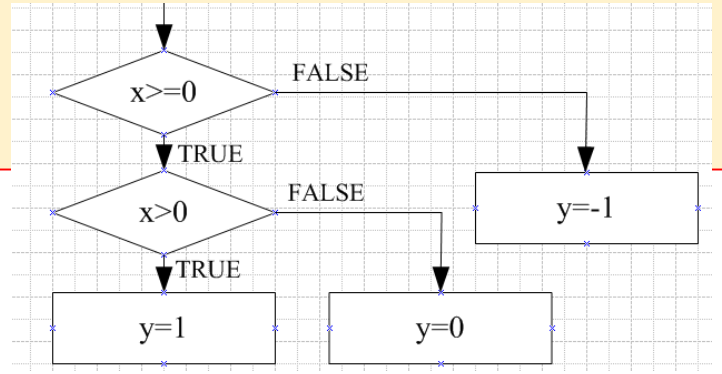
**方法一**（多分支结构）：

```
if (x > 0): y = 1
elif (x == 0): y = 0
else: y = -1
```



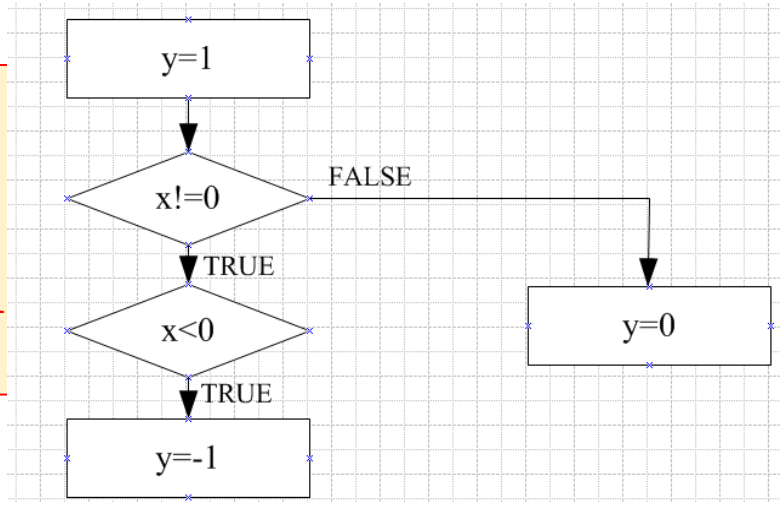

**方法二**（if语句嵌套结构）：

```
if (x >= 0):
    if (x > 0): y = 1
    else: y = 0
else: y = -1
```




**方法三**：

```
y = 1
if (x != 0):
    if (x < 0): y = -1
else: y = 0
```




```
if (条件表达式 1):
    if (条件表达式 11):
        语句 1
    [else:
        语句 2]                    } 内嵌 if
[else:
    if (条件表达式 21):
        语句 3
    [else:
        语句 4]]                   } 内嵌 if
```

# if语句典型示例代码

| 程序功能 | 代码片段 |
| --- | --- |
| 求绝对值 | if a < 0:<br>  a = -a |
| a 和 b 按升序排序 | if a > b:<br>  t = a<br>  a = b<br>  b = t |
| 求 a 和 b 的最大值 | if a > b: maximum = a<br>else:  maximum = b |
| 计算两个数相除的余数，如果除数为 0，则给出报错信息 | if b == 0: print("除数为 0")<br>else: print("余数为： " + a % b) |
| 计算并输出一元二次方程的两个根。如果判别式 $b^2-4ac<0$，则显示“方程无实根”的提示信息 | delta = b*b - 4.0*a*c<br>if delta < 0.0:<br>  print("方程无实根")<br>else:<br>  d = math.sqrt(delta)<br>  print((-b + d)/(2.0*a))<br>  print((-b - d)/(2.0*a)) |

# 选择结构综合举例（1）

- 【例】输入三个数，按从大到小的顺序排序
- 先a和b比较，使得a>b；然后a和c比较，使得a>c，此时a最大；最后b和c比较，使得b>c

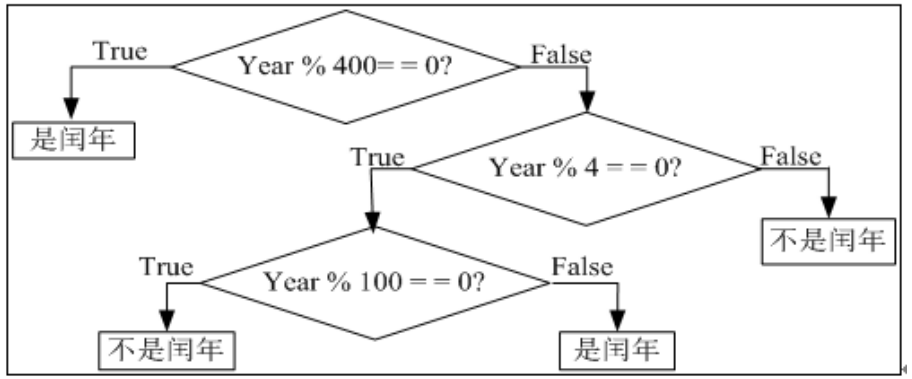
```
a = int(input("请输入整数a： "))
b = int(input("请输入整数b： "))
c = int(input("请输入整数c： "))
if (a < b):
    a, b=b, a    #a和b交换，使得a>b
if (a < c):
    a, c=c, a    #a和c交换，使得a>c
if (b < c):
    b, c=c, b    #b和c交换，使得b>c
print("排序结果（降序）： ", a, b, c)
```

程序运行结果如下：

```
请输入整数 a：3
请输入整数 b：2
请输入整数 c：5
排序结果（降序）： 5 3 2
```

# 选择结构综合举例（2）

- 【例】编程判断某一年是否为闰年。判断闰年的条件是：年份能被4整除但不能被100整除，或者能被400整除




**方法一**。使用一个逻辑表达式包含所有的闰年条件：

```
y=eval(input("请输入年份:"))
if ((y % 4 == 0 and y % 100 != 0) or y % 400 == 0):
    print("是闰年")
else: print("不是闰年")
```

**方法二**。使用嵌套的if语句：

```
y=eval(input("请输入年份:"))
if (y % 400 == 0): print("是闰年")
else:
    if (y % 4 == 0):
        if (y % 100 == 0): print("不是闰年")
        else: print("是闰年")
    else: print("不是闰年")
```

**方法三**。使用if-elif语句：

```
y=eval(input("请输入年份:"))
if (y % 400 == 0): print("是闰年")
elif (y % 4 != 0): print("不是闰年")
elif (y % 100 == 0): print("不是闰年")
else: print("是闰年")
```

**方法四**。使用calendar模块的isleap函数判断闰年

```
y=eval(input("请输入年份:"))
import calendar
if (calendar.isleap(y)): print("是闰年")
else: print("不是闰年")
```

# Python的字符串驻留机制

- a = '1234'
- b = '1234'
- a==b
- ?
- id(a)==id(b)
- ?
- a*=10000
- b*=10000
- a==b
- ?
- id(a)==id(b)
- ?

Python支持字符串驻留机制，即：对于短字符串，将其赋值给多个不同的对象时，内存中只有一个副本，多个对象共享该副本。
这一点不适用于长字符串，即长字符串不遵守驻留机制

判断一个数是不是黑洞数，黑洞数是指这样的整数：由这个数字每位上的数字组成的最大数减去每位数字组成的最小数仍然得到这个数自身。例如 3 位黑洞数是495，因为 954-459=495，4 位数字是6174，因为 7641-1467=6174。

```
a = int(input('请输入一个数字不全相同的数'))
def heidongshu(n):
    max=min=''
    x1=sorted(str(n),reverse=True)
    x2=sorted(str(n))
    for i in x1:
        max+=str(i)
    for j in x2:
        min+=str(j)
    if int(max)-int(min) ==n:
        return True
    else:
        return False
print(heidongshu(a)
```

# 编写程序，生成一个包含50个随机整数的列表，然后删除其中所有奇数。

```
import random
x = [random.randint(0,100) for i in range(50)]
print(x)

for i in range(len(x))[::-1]:
    if x[i]%2 == 1:
        del x[i]
print(x)
```

# 3.4 循环结构（1）

- for语句和while语句来实现循环结构
- 可迭代对象（iterable）
  - 系列（sequence），例如字符串（str）、列表（list）、元组（tuple）
  - 字典（dict）。
  - 文件对象。
  - 迭代器对象（iterator）。
  - 生成器函数（generator）。
- for循环

**for 变量 in 可迭代对象集合:**

**循环体语句/语句块**

```
>>> for i in (1,2,3):
        print(i, i**2, i**3)
```

```
1 1 1
2 4 8
3 9 27
```

# 3.4 循环结构（2）

- range对象 **range(start, stop[, step])**
  - 从start开始，到stop结束（不包含stop）。如果指定了可选的步长step，则序列按步长增长

```
>>> for i in range(1,11): print(i, end=' ')      #输出：1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
>>> for i in range(1,11,3): print(i, end=' ')    #输出：1 4 7 10
1 4 7 10
```

- 【例】利用for循环求1~100中所有奇数的和以及偶数的和（for_sum1_100.py）

```
sum_odd = 0; sum_even = 0      #循环赋初值
for i in range(1, 101):        #i=1~100
    if i % 2 != 0:             #奇数
        sum_odd += i           #奇数和
    else:                      #偶数
        sum_even += i          #偶数和
print("1~100中所有奇数的和：", sum_odd)
print("1~100中所有偶数的和：", sum_even)
```

程序运行结果如下。

```
1~100 中所有奇数的和: 2500
1~100 中所有偶数的和: 2550
```

# 编写程序，至少使用2种不同的方法计算100以内所有奇数的和

- print(sum([i for i in range(1,100) if i%2==1]))

- print(sum(range(1,100)[::2]))

# break语句（1）

- 用于退出for、while循环，即提前结束循环，接着执行循环语句的后继语句
  - 当多个for、while语句彼此嵌套时，break语句只应用于最里层的语句，即break语句只能跳出最近的一层循环
- 【例】使用break语句终止循环（break.py）

```
while True:
    s = input('请输入字符串（按Q或者q结束）：')
    if s.upper() == 'Q':
        break
    print('字符串的长度为：', len(s))
```

程序运行结果如下：

```
请输入字符串（按 Q 或者 q 结束）：Hello, World!
字符串的长度为： 13
请输入字符串（按 Q 或者 q 结束）：您好！
字符串的长度为： 3
请输入字符串（按 Q 或者 q 结束）：q
```

# break语句（2）

- 【例3.19】编程（prime1.py和prime2.py）判断所输入的任意一个正整数是否为素数

**方法一**（利用for循环和break语句）：

```
import math
m = int(input("请输入一个整数(>1)： "))
k = int(math.sqrt(m))
for i in range(2, k + 2):
    if m % i == 0:
        break  #可整除，肯定非素数，结束循环
if i == k+1 : print(m, "是素数！ ")
else: print(m, "是合数！ ")
```

**方法二**（利用while循环和bool变量）：

```
import math
m = int(input("请输入一个整数(>1)： "))
k = int(math.sqrt(m))
flag = True                  #先假设所输整数为素数
i = 2
while (i <= k and flag == True):
    if (m % i == 0): flag = False #可整除，肯定非素数，结束循环
    else: i += 1
if (flag == True): print(m, "是素数！ ")
else: print(m, "是合数！ ")
```

# continue语句（1）

- 结束本次循环，即跳过循环体内自continue下面尚未执行的语句，返回到循环的起始处，并根据循环条件判断是否执行下一次循环
- continue语句与break语句的区别
  - continue语句仅结束本次循环，并返回到循环的起始处，循环条件满足的话就开始执行下一次循环；而break语句则是结束循环，跳转到循环的后继语句执行

# continue语句（3）

- 【例】显示100~200之间不能被3整除的数。要求一行显示10个数

```
100~200之间不能被3整除的数为:
100   101   103   104   106   107   109   110   112   113
115   116   118   119   121   122   124   125   127   128
130   131   133   134   136   137   139   140   142   143
145   146   148   149   151   152   154   155   157   158
160   161   163   164   166   167   169   170   172   173
175   176   178   179   181   182   184   185   187   188
190   191   193   194   196   197   199   200
```

```
j = 0                          #控制一行显示的数值个数
print('100~200之间不能被3整除的数为：')
for i in range(100, 200 + 1):
    if (i % 3 == 0): continue                    #跳过能被3整除的数
   print(str.format("{0:<5}",i), end="")    #每个数占5个位置，不足后面加空格，并且不换行
    j += 1
    if (j % 10 == 0): print()                    #一行显示10个数后换行
```

# 死循环（无限循环）

- 如果while循环结构中循环控制条件一直为真，则循环将无限继续，程序将一直运行下去，从而形成死循环
- 程序死循环时，会造成程序没有任何响应；或者造成不断输出（例如控制台输出，文件写入，打印输出等）
- 在程序的循环体中，插入调试输出语句print，可以判断程序是否为死循环。注意，有的程序算法十分复杂，可能需要运行很长时间，但并不是死循环
- 可以使用快捷键<Ctrl>+<c>终止当前程序的运行
- 【例】死循环示例

```
import math
while True:       #循环条件一直为真
   num = float(input("请输入一个正数:"))
   print(str(num), "的平方根为： ", math.sqrt(num))
print("Good bye!")
```

# 循环语句典型示例代码

| 功能示例 | 实现代码 |
| --- | --- |
| 输出 n 个数(0~n-1)的 2 的乘幂的值列表 | power = 1<br>for i in range(n):<br>····print(str(i) + ": " + str(power))<br>····power *= 2 |
| 输出小于或等于 n 的最大的 2 的乘幂的值 | power = 1<br>while 2*power <= n:<br>····power *= 2<br>print(power) |
| 计算并输出 1 + 2 + . . . . + n 的累积和 | total = 0<br>for i in range(1, n+1):<br>····total += i<br>print(total) |
| 计算并输出 n 的阶乘（n! = 1 × 2 × . . . . × n） | factorial = 1<br>for i in range(1, n+1):<br>····factorial *= i<br>print(factorial) |
| 输出半径为 1~n 的圆的周长列表 | for r in range(1, n+1):<br>····print("r="+str(r), end=" ")<br>····print("p="+str(2.0 * math.pi * r)) |

# 循环结构综合举例（1）

- 【例3.27】使用牛顿迭代法求解平方根
  - 计算一个正实数a的平方根，可以使用牛顿迭代法实现：首先假设t=a，开始循环。如果t=a/t（或小于容差），则t等于a的平方根，循环结束并返回结果；否则，将t和a/t的平均值赋值给t，继续循环

```
EPSILON = 1e-15                          #容差
a = float(input("请输入正实数a： "))  #正实数a
t = a                                    #假设平方根t=a
while abs(t - a/t) > (EPSILON * t):
    t = (a/t + t) / 2.0                  #将t和a/t的平均值赋值给t
print(t)                                 #输出a的平方根
```

```
请输入正实数a： 2
1.414213562373095
```

# 循环结构综合举例（2）

- 【例3.28】显示Fibonacci数列：1、1、2、3、5、8、……的前20项

即 $\begin{cases} F_1 = 1 & n = 1 \\ F_2 = 1 & n = 2 \\ F_n = F_{n-1} + F_{n-2} & n >= 3 \end{cases}$

```
f1 = 1; f2 = 1
for i in range(1, 11):
    print(str.format("{0:6}{1:6}", f1, f2), end=" ") #每次输出2个数，每个数占6位，空格分隔
    if i % 2 == 0: print()                           #显示4项后换行
    f1 += f2; f2 += f1
```

```
     1     1     2     3
     5     8    13    21
    34    55    89   144
   233   377   610   987
  1597  2584  4181  6765
```

# 简单实现汉诺塔的流程（n个按大小排列的圆盘，从a柱子移动到c柱子上去）

```
nt=eval(input("请输入您所需的汉诺塔阶数："))
def move(nt,a,b,c):
    if nt==1:
        print(a,"->",c)
        return
    move(nt-1,a,c,b)
    move(1,a,b,c)
    move(nt-1,b,a,c)
move(nt,"a","b","c")
```

# 编写函数，接收一个正偶数为参数，输出两个素数，并且这两个素数之和等于原来的正偶数。如果存在多组符合条件的素数，则全部输出。

```
import math
def IsPrime(n):
    m=int(math.sqrt(n))+1
    for i in range(2,m):
        if n%i==0:
            return False
    return True

def demo(n):
    if isinstance(n,int) and n>0 and n%2==0:
        for i in range(3,int(n/2)+1):
            if i%2==1 and IsPrime(i) and IsPrime(n-i):
                print(i,'+',n-i,'=',n)

x=eval(input("请输入一个正偶数:"))
demo(x)
```

双色球是一种比较常见的彩票玩法，
每一注彩票由6个介于1到33之间的不重复数字和1个介于1到16之间的数字组成。
下面的代码用来随机生成一注双色球彩票，结果是完全随机的

```
import random


def doubleColor():
    red = random.sample(range(1,34), 6)
    blue = random.choice(range(1, 17))
    return str(red)+'-'+str(blue)


print(doubleColor())
```

# 编写程序，用户从键盘输入小于1000的整数，对其进行因式分解。例如，10=2×5，60=2×2×3×5

```
x = input('请输入一个小于1000的数:')
x = eval(x)
t = x
i = 2
result = []
while True:
    if t==1:
        break
    if t%i == 0:
        result.append(i)
        t = t//i
    else:
        i += 1
print(x,'=', '*'.join(map(str,result)))
```

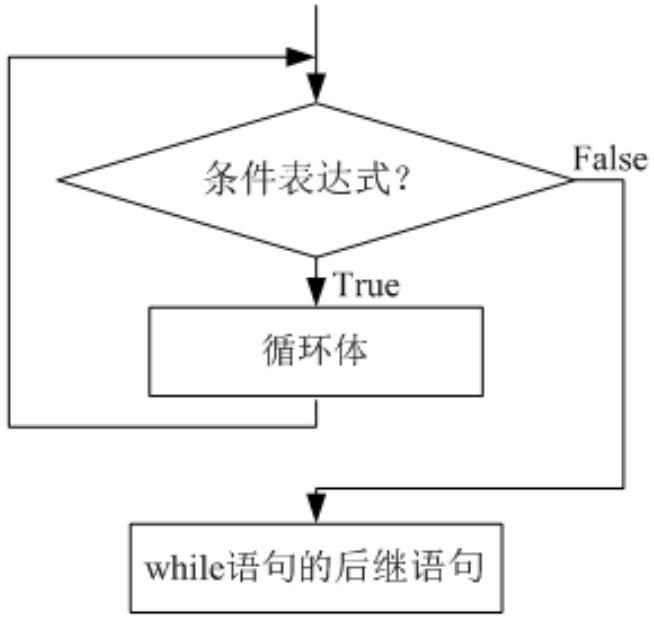
# while循环（1）

- 【例】利用while循环求 $\sum_{i=1}^{100} i$，以及1~100中所有奇数的和、偶数的和（while_sum.py）

```
i = 1; sum_all = 0; sum_odd = 0; sum_even = 0
while (i <= 100):
    sum_all += i          #所有数之和
    if (i % 2 == 0):      #偶数
        sum_even += i     #偶数和
    else:                 #奇数
        sum_odd += i      #奇数和
    i += 1
print("和=%d、奇数和=%d、偶数和=%d"    %
(sum_all, sum_odd, sum_even))
```

**while (条件表达式):**

**    循环体语句/语句块**




程序运行结果如下：

和=5050、奇数和=2500、偶数和=2550

# 第一题

用如下近似公式求自然对数的底数e的值，直到最后一项的绝对值小于$10^{-6}$为止（使用while循环）

$$e \approx 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots\cdots + \frac{1}{n!}$$

- **第二题**

使用break和continue语句跳过循环。要求输入若干学生成绩（按Q或q结束），如果成绩<0，则重新输入。统计学生人数和平均成绩

程序运行结果如下：

```
请输入学生成绩（按 Q 或 q 结束）：65
请输入学生成绩（按 Q 或 q 结束）：87
请输入学生成绩（按 Q 或 q 结束）：-40
请输入学生成绩（按 Q 或 q 结束）：q
学生人数为：2，平均成绩为：76.0
```

- **第三题 请**使用相应语句实现按照下边的界面进行输入，要求是输入三个爱好且没有被中断，则输出-您输入了三个爱好，并最终显示：您的爱好为…　。假如中间用户输入了q或者Q导致程序中断，则只输出您的爱好为　。

程序运行结果如下：

```
>>>
请输入爱好之一（最多三个，按Q或q结束）：旅游
请输入爱好之一（最多三个，按Q或q结束）：音乐
请输入爱好之一（最多三个，按Q或q结束）：运动
您输入了三个爱好。
您的爱好为：    旅游  音乐  运动
>>>
请输入爱好之一（最多三个，按Q或q结束）：音乐
请输入爱好之一（最多三个，按Q或q结束）：q
您的爱好为：    音乐
```

第四题 前边课堂上学习过了斐波那契数列，请编写一个函数，其参数为一个整数t，返回斐波那契数列中大于t的第一个数。（效果如下图）