第5章 函　　数

5.1 引　　例

例 5.1.1　逆序数的逆序和

输入两个正整数，先将它们分别倒过来，然后相加，最后将结果倒过来输出。注意，前置的 0 将被忽略。例如，输入 305 和 794。倒过来相加得到 1000，输出时只要输出 1 就可以了。测试数据保证结果不超过 $2^{31}-1$。

输入格式：

首先输入一个正整数 T，表示测试数据的组数，然后输入 T 组测试数据。每组测试输入两个正整数 a、b。

输出格式：

对于每组测试，将 a、b 逆序后求和并逆序输出（前导 0 不需要输出）。

输入样例：	输出样例：
2 21 6 123 456	81 579

解析：

本题需求 3 次逆序数，若每次都重复写一个循环实现，则代码将较冗长。因为求逆序数的方法是一样的，可以编写一个求逆序数的函数，调用 3 次即可完成两个输入的整数及一个结果整数的逆序。

思考：当 $x=1234$ 时，如何得到 x 的逆序数？

设 r 为 x 的逆序数，可以这样考虑：$r=((4*10+3)*10+2)*10+1=4321$，即让 r 一开始为 0，再不断把 x 的个位取出来加上 $r*10$ 重新赋值给 r，直到 x 为 0（通过 $x=x//10$ 不断去掉 x 的个位）。具体代码如下。

```
def revNum(x):                          #自定义函数,求参数 n 的逆序数,此行是函数头
    #函数体
    r=0
    while x>0:
        r=r*10+x%10
        x=x//10
```

```
    return r                                    #返回逆序数
T=int(input())
for t in range(T):
    m,n=map(int,input().split())
    res=revNum(revNum(m)+revNum(n))    #3 次调用 revNum 函数
    print(res)
```

运行结果如下。

```
2↵
1234 5708↵
69321
305 794↵
1
```

函数定义以关键字 def 开始,revNum 是函数名,其后的小括号及其后的冒号是必需的,小括号中的 x 是形式参数,用于调用时接收传递过来的实际参数(如 m、n)的值,完成逆序的代码写在冒号之后构成函数体;函数需调用才起作用。程序员在编写程序时,通常会把一个程序中多次使用的代码写成一个函数再调用多次,如本例所示。实际上,一些常用功能即使在一个程序中不被调用多次,也经常写成一个个自定义函数,例如,判断一个数是否是素数、求两个整数的最大公约数或最小公倍数、二分查找及排序等。另外,本例所写的 revNum 函数能够忽略前导 0,原因请读者自行分析。

通过内置函数 str、int、len 及字符串切片操作,可将整数 n 转换为字符串,再取其逆序串转换为整数返回,如此可简化定义求逆序数的 revNum 函数如下。

```
def revNum(n):
    return int(str(n)[-1:-len(str(n))-1:-1])
```

5.2 函数基础知识

5.2.1 函数概述

简言之,函数是一组相关语句组织在一起所构成的整体,并以函数名标注。

从用户的角度而言,函数分为库函数和用户自定义函数。库函数有很多,包括可以直接调用的内置函数及其他标准库或扩展库中的函数,例如,range、print、abs、max、min、sum、sqrt、randint 等。具体用法举例如下。

```
>>>a=range(10); s=sum(a); print(s)   #调用内置函数 range、sum 分别产生数列、求和
45
>>>b=[1,13,52,6,8]
```

```
>>>m=max(b); n=min(b); print(m,n)     #调用内置函数 max、min 分别求最大、最小值
52 1
>>>c=-123.45; print(abs(c))          #调用内置函数 abs 求绝对值
123.45
>>>from math import sqrt             #导入数学库(或称数学模块)math 中的 sqrt 函数
>>>print(sqrt(2))                    #调用 sqrt 函数,求平方根
1.4142135623730951
>>>from random import randint        #导入随机库 random 中的 randint 函数
>>>randint(10,99)                    #调用 randint 函数,产生[10, 99]范围内的整数
52
```

本章主要介绍用户自定义函数。读者可以自行查阅并测试感兴趣的库函数。

一个大的程序一般分为若干程序模块,每个模块用来实现一个特定的功能,每个模块一般定义一个函数来实现。

5.2.2 函数的定义与调用

1. 函数定义

函数定义由函数头和函数体两部分组成。一般形式如下。

```
def 函数名([形参列表]):
    函数体
```

说明:

(1) "def 函数名([形参列表]):"是函数头,函数定义必须以关键字 def 开头,函数名后的小括号不能缺少;冒号之后缩进量相同的若干语句构成函数体。

(2) 函数名必须是合法的标识符。

(3) 函数定义中的参数为形式参数,简称形参。根据是否有形参,函数可分为带参函数和无参函数。形参列表的每个参数指定参数名即可(参数类型根据函数调用时的实参确定),形参列表若有多个参数,则以逗号间隔。Python 支持参数带默认值,而且默认值参数须位于最右边,即任意一个默认值参数之后的所有参数都应该带默认值。

(4) 根据是否有返回值,函数可分为有返回值函数(一般作为表达式调用)和无返回值函数(返回 None,一般作为语句调用)。通过函数中的 return 语句返回函数的返回值。return 语句的一般格式如下。

```
return [返回值表达式]
```

其中[ ]表示返回值表达式可省略,若返回值表达式省略,则仅用 return 语句将控制程序流程返回到调用点;若 return 语句带返回值表达式,则在控制程序流程返回调用点的同时带回一个值。

在 Python 中,函数可以嵌套定义,即在一个函数的函数体中再定义另一个函数。

2. 函数调用

函数定义好之后必须调用才能起作用。函数调用的形式一般如下。

```
[变量=]函数名([实参列表])
```

无返回值的函数一般以语句形式调用,有返回值的函数一般以表达式形式调用,否则其返回值没有意义。

调用时的参数称为实际参数,简称实参。一般情况下,参数的类型、顺序、个数必须与函数定义中的一致,但带默认值参数的函数调用时实参个数可以与形参个数不一致;若调用时指定形参名(关键字参数),则实参的顺序可与函数定义的形参列表中指定的顺序不一致。

函数调用时,先把实参(若有)依序传递给形参,然后执行函数定义体中的语句,执行到 return 语句或函数结束时,程序流程返回到调用点。

3. 函数定义与调用示例

用户自定义函数需先定义再调用。下面给出若干函数定义与调用的例子。

```
def sayHello():                          #无参函数,也是无返回值函数,小括号不能省略
    print("Hello")

def max(a,b):                            #带参函数,也是有返回值函数,有两个形参(用逗号分隔)
    if a>=b:
        return a                         #返回语句,在流程返回的同时带回变量 a 的值
    else:
        return b

def cal(a,b,c='+'):                      #带默认值参数的函数,默认值参数写在最右边
    if c=='+':
        return a+b
    elif c=='-':
        return a-b

def g(n):                                #函数嵌套定义,在 g 函数中定义 f 函数
    def f(n):
        return n**3
    return f(n)

sayHello()                               #无返回值的函数一般作语句调用
print(max(123,321))                      #根据实参确定形参类型
print(max("abcde","abcDE"))              #根据实参确定形参类型
print(cal(1,2))                          #因第 3 个参数未提供,故其使用默认值
print(cal(1,2,'-'))                      #为默认值参数指定实参
print(cal(c='-',b=1,a=2))                #若调用时指定形参名,则实参顺序可与形参不一致
print(g(5))                              #调用 g 函数
```

运行结果如下。

```
Hello
321
abcde
3
-1
1
125
```

上面这些函数的实参都是常量。实际上，函数调用经常使用变量作为实参，若实参变量是不可变类型的引用，则把实参变量的值传递给形参（此类参数简称值参，形参的改变不影响实参），否则形参成为实参变量的引用（此类参数称为引用参数，形参的改变即实参的改变）。实参和形参可以同名，但它们实际上是各自作用域内的不同变量。对于带默认值参数的函数，若在调用时不指定对应的实参，则该参数使用定义时指定的默认值。

形参和函数体中创建的变量是仅在该函数中有效的局部变量，而在函数外创建的变量则是全局变量（或称外部变量），从创建处开始往下都有效。

注意，在 Python 中，若全局变量定义在某个函数 f 的调用之前，则该全局变量可在 f 函数定义中使用。例如：

```
def f():
    print(n**3)                #使用本函数定义之后调用之前创建的全局变量 n,可行
    print(m**2)                #使用本函数定义及调用之后的创建的全局变量 m,出错

n=3                            #在调用 f 函数之前创建全局变量 n,则 f 函数中可以使用 n
f()
m=5                            #在调用 f 数之后创建全局变量 m,则 f 函数中不能使用 m
```

运行结果如下。

```
27
Traceback (most recent call last):
  File "D:/Python/test.py", line 6, in <module>
    f()
  File "D:/Python/test.py", line 3, in f
    print(m**2)                #使用本函数定义及调用之后创建的全局变量 m,出错
NameError: name 'm' is not defined
```

注意，若要在函数定义中修改全局变量，则需用关键字 global 声明该全局变量，格式如下。

```
global 全局变量
```

若在函数定义中给全局变量赋值但之前未用 global 声明该变量，则将在函数中创建一

个与全局变量同名的局部变量。

修改全局变量的简单示例如下。


```
n=123                          #n 是全局变量

def f(t):                      #定义无参函数 f,参数 t 是局限于 f 函数的局部变量
    m=456                      #m 是局部变量,仅在 f 函数中有效
    global n                   #若无此语句,则下一条语句将创建局部变量 n
    n=789                      #因有上一条全局变量声明语句,故此处是在修改全局变量 n
    print(m+n+t)               #输出 1368

f(n)                           #函数需要调用才有效果
print(n)                       #输出修改后的全局变量 n 的值: 789
```

5.2.3 不定长参数

在 Python 中,可以使用不定长参数,即在形参之前加一个星号“*”(该参数接收实参后成为一个元组)或两个星号“**”(该参数接收实参后成为一个字典,且实参应包含参数名和值,其中参数名成为字典中的一个键,参数值成为该键对应的值)。注意,这与可迭代对象作为实参时在其之前加的星号(把可迭代对象中的元素逐个取出成为值参)是不同的。

在形参之前加一个星号的不定长参数的示例如下。

```
def f(a,b,*c):                 #形参 c 之前带 *,表示不定长参数
    print("first:",a)
    print("second:",b)
    print("third:",c)          #参数 c 接收实参后成为一个元组
    print(*c)                  #此处的 * 表示逐个取元组 c 中的元素作为 print 函数的值参

#调用 f 函数,第 1 个参数 1 传递给 a,第 2 个参数 2 传递给 b,剩余的 3 个参数传递给 c
f(1,2,3,4,5)
#调用 f 函数,列表之前的 * 表示逐个取列表中的元素作为 f 函数的值参
f(*[1,2,3,4,5])                #相当于 f(1,2,3,4,5)
```

运行结果如下。

```
first: 1
second: 2
third: (3, 4, 5)
3 4 5
first: 1
second: 2
third: (3, 4, 5)
3 4 5
```

在形参之前加两个星号的不定长参数的示例如下。

```
def f(a,b,**c):                    #形参 c 之前带 **,表示不定长参数
    print("first:",a)
    print("second:",b)
    print("third:",c)              #参数 c 接收实参后成为一个字典
    print(*c)                      #此处的 * 表示逐个取字典 c 中的键作为 print 函数的值参

#调用 f 函数,第 1 个参数 1 传递给 a,第 2 个参数 2 传递给 b,剩余的 3 个参数传递给 c
f(1,2,x=3,y=4,z=5)                 #前两个参数之外的其他参数需有参数名和值
```

运行结果如下。

```
first: 1
second: 2
third: {'x': 3, 'y': 4, 'z': 5}
x y z
```

5.2.4 列表作为函数参数

列表元素作为函数实参时,与普通变量作为函数实参是一致的,即把列表元素的值传递给实参,形参的变化不会影响实参。例如:

```
def f(a):
    a=123

b=[2,5,6]
f(b[1])
print(b)
```

运行结果如下。

```
[2, 5, 6]
```

列表名作为函数的参数,指的是形参和实参都使用列表名。此时形参列表是实参列表的引用,即在函数调用期间形参列表与实参列表是同一个列表,因此对形参列表的改变就是对实参列表的改变。例如:

```
def g(a):
    for i in range(len(a)):
        a[i]=a[i]**3
```

```
b=[2,5,6]
g(b)
print(b)
```

运行结果如下。

```
[8, 125, 216]
```

例 5.2.1　*m* 趟选择排序

先在第 1 行输入整数 n 和 m,再在第 2 行输入 n 个整数构成的数列,要求利用选择排序(每趟排序最多交换一次)进行排序,并输出第 m 趟排序后的数列状况。把选择排序定义为一个函数。

解析:

选择排序的思想和方法在前面的章节中已经讨论过,这里以函数的形式表达,且排序趟数以参数 m 控制。具体代码如下。

```
def selectSort(a,n,m):                  #对包含 n 个元素的整型列表 a 进行 m 趟排序
    for i in range(m):                  #n 个数进行 m 趟排序
        k=i                             #k 是假设最小数的下标,第 i 趟假设下标为 i 的元素最小
        for j in range(i+1,n):          #控制假设最小数与其后数据比较
            if a[k]>a[j]:               #若后面的数据小于假设最小数,则将其下标保存到 k 中
                k=j
        if k!=i:                        #若当前最小数不在当前最前面位置,则交换
            a[k],a[i]=a[i],a[k]

n,m=map(int,input().split())
b=list(map(int,input().split()))
selectSort(b,n,m)                       #调用 m 趟选择排序的函数
print(*b)                               #输出结果
```

运行结果如下。

```
6 3↵
3 5 1 2 8 6↵
1 2 3 5 8 6
```

从运行结果可见,实参列表 b 在调用 selectSort 函数之后发生了改变,即形参列表 a 的改变影响到了实参列表 b。因列表名作为函数参数时,传递的是引用,在函数调用期间形参列表 a 与实参列表 b 是同一个列表,故对列表元素 $a[i](0\leqslant i<n)$的改变就是对 $b[i]$的改变。

5.2.5　匿名函数

Python 语言使用关键字 lambda 创建匿名函数,形式如下。

```
[函数名=]lambda [参数 1[,参数 2,…,参数 n]]: 表达式
```

关键字 lambda 创建包含简单逻辑的函数，其参数位于 lambda 和冒号“:”之间，可以有 0 个、1 个或多个参数，若有多个参数则以逗号“,”分隔，其主体部分（冒号之后）是一个表达式。可以通过赋值语句给匿名函数取名。例如：

```
f1=lambda a, b : a if a>=b else b   #创建包含两个参数的匿名函数,并取名为 f1
print(f1(12,34))

c=1
f2=lambda a, b : c if a>b else 0    #匿名函数的主体中只能使用参数和全局变量
print(f2(123,78))

f3=lambda a: a**3                   #一个参数的匿名函数,并取名为 f3
print(f3(3))

f4=lambda : "Hello"                 #无参匿名函数,并取名为 f4
print(f4())
```

运行结果如下。

```
34
1
27
Hello
```

lambda 匿名函数可返回整数、实数、字符串、元组和列表等不同类型的值。例如，如下 f 函数和 g 函数分别返回一个元组和一个列表。

```
f=lambda a : (a**2, a**3)           #返回元组
g=lambda n : [n%2==0, n]            #返回列表
f(3)                                #调用返回元组(9, 27)
g(3)                                #调用返回列表[False, 3]
```

lambda 匿名函数可理解为简化的函数定义，例如，上述 g 函数相当于如下函数定义。

```
def g(n):
    return [n%2==0, n]
```

5.3 函数举例

例 5.3.1　素数判断

输入一个正整数 n，判断 n 是否是素数，若是则输出 yes，否则输出 no。要求写一个判断一个正整数是否是素数的函数。

解析：

关于 n 是否是素数,可从 2 至$\sqrt{n}$判断是否有 n 的因子,若有则 n 不是素数。素数判断在 3.3.5 节的例 3.3.14 中已有介绍,这里可把相关代码作为一个整体写成一个函数;因为要判断一个数是否是素数,所以该函数需有一个参数。具体代码如下。

```
from math import sqrt                    #导入函数 sqrt
def isPrime(n):                          #判断 n 是否是素数的函数
    flag=True                            #假设 n 是素数,标记设为 True
    k=int(sqrt(n))                       #求得 sqrt(n),整数部分存放在 k 中
    for i in range(2,k+1):               #从 2 到 k 判断是否存在 n 的因子
        if n%i==0:                       #若 i 是 n 的因子,则 n 不是素数,结束循环
            flag=False
            break
    if n==1: flag=False                  #对 1 进行特判
    return flag

n=int(input())
if isPrime(n)==True:                     #若 n 是素数
    print("yes")
else:
    print("no")
```

运行结果如下。

```
2147483647↵
yes
```

例 5.3.2　最小回文数

输入整数 n,输出比该数大的最小回文数。其中回文数指的是正读、反读都一样的数,如 131、1221 等。要求写一个判断一个整数是否是回文数的函数。

解析:

判断是否是回文数可以调用例 5.1.1 中的求逆序数的函数 revNum,判断该数与逆序数是否相等。因为要找比 n 大的最小回文数,可以从 $n+1$ 开始逐个检查是否满足回文数的条件,第一个满足条件的数即为结果。

```
def revNum(n):                           #求逆序数的函数
    s=0
    while n>0:
        s=s*10+n%10
        n=n//10
    return s

def isSymmetric(n):                      #判断是否是回文数的函数
    return n==revNum(n)                  #若 n 等于其逆序数,则返回 True,否则返回 False
```