

###例 6.1 设 $f(0)=0, f(1)=1$, 当 $n \geq 2$ 时, $f(n)=f(n-1)+f(n-2)$, 求 $f(n)$ 的前 8 项值

#斐波那契数列

```
def fibs(i):
    f=[0,1]
    for n in range(2,i):
        f.append(f[n-1]+f[n-2])
    return f
print(fibs(8))
```

例 6.7 求输入多个数的和

```
def sum(*x):
    result = 0 #定义一个变量用来保存结果
    for i in x :# 遍历元组, 并将元组中的数进行累加
        result += i
    print(result)
sum(1,2,5,6) #调用函数
sum(1,2) #调用函数
sum( ) #调用函数
```

例 6.8 定义一个带星号*和带两个星号**的函数。

```
def my_fun(a,b,*c,d,**e):
    print("a=",a)
    print("b=",b)
    print("c=",c)
    print("d=",d)
    print("e=",e)
my_fun(3,4,1,2,5,6,d=9,x1=1,x2=2,y1=3,y2=4)
```

例 6.9 定义一个带星号*和带两个星号**的函数

```
def my_fun(a,b,*c,d,**e):
    print('a=',a)
    print('b=',b)
    print('c=',c)
    print('d=',d)
    print('e=',e)
kwargs={'x1':1, 'x2':2, 'y1':3, 'y2':4}
my_fun(3,4,1,2,5,6,d=9,**kwargs)
s=(1,2,5,6)
kwargs={'x1':1, 'x2':2, 'y1':3, 'y2':4}
my_fun(3,4,*s,d=9,**kwargs)
```

例 6.10a 默认值参数在*参数前面

```
def print_student(id,sex='male',*score):
    print(f'学号={id},性别={sex}')
    if score:
```

```
print(f'成绩={score}, 平均分={sum(score)/len(score)}')
```

```
fenshu=[95,100,80,84]
```

#语法正确，参数传递错误，相当于把 95 按位置传递给 sex 参数

```
print_student('12103080121',95,100,80,84)
```

```
print('-----')
```

#下句语法正确，参数传递错误，*号相当于解包操作，跟上一句一样的效果。

```
print_student('12103080121',*fenshu)
```

```
print('-----')
```

```
print_student('12103080121','female',*fenshu) #★★★正确传递
```

例 6.10b 默认值参数在*参数后面

```
def print_student(id,*score,sex='male'):
```

```
    print(f'学号={id},性别={sex}')
```

```
    if score:
```

```
        print(f'成绩={score}, 平均分={sum(score)/len(score)}')
```

```
fenshu=[95,100,80,84]
```

```
print_student('12103080121',*fenshu) #正确传递，sex 参数使用默认值
```

```
print('-----')
```

```
print_student('12103080121',95,100,80,84) #跟上一句效果一样
```

```
print('-----')
```

```
# print_student('12103080121',*fenshu, 'female')
```

#上面这条 print 语句语法错误。*参数后的参数必须给定参数名称

#★★★下面这句正确，对 sex 参数按关键字参数方式给定值

```
print_student('12103080121',*fenshu,sex='female')
```

```
print('-----')
```

例 6.11 实现九九乘法表的函数

```
def multiplicationTable():
```

```
    for i in range(1,10):
```

```
        for j in range(1,i + 1):
```

```
            print(f'{i} * {j} = {i * j}', end="\t")
```

```
        print()
```

```
multiplicationTable() #该函数返回 None，可以打印试试看
```

例 6.12 定义一个函数，计算给定整数 M 和 N 区间内所有整数的积

```
def mul_fun(m,n):
```

```
    result=1
```

```
    for i in range(m,n+1):
```

```
        result=result*i
```

```
    print(f'{m}—{n}的所有整数的乘积是:{result}')
```

```
    return result
```

```
print("*****")#该语句放在 return 后面，执行不到
print(mul_fun(2,6))
```

例 6.13 定义一个函数，计算给定整数 **M** 和 **N** 区间内所有整数的和、积

```
def sum_and_mul(n1,n2):
    s=0
    m=1
    for i in range(n1,n2+1):
        s+=i
        m*=i
    return s,m          #返回的是元组形式
a,b=sum_and_mul(5,7)  #对元组解包操作
c=sum_and_mul(5,7)
print(a,b)
print(c)
```

例 6.14 一球从 100 米高度自由落下，每次落地后反跳回原高度的一半再落下，求它在第 **n** 次落地时，共经过多少米？第 **n** 次反弹多高？

```
def ball(height,n):
    i=1
    s=height
    f=height/2
    while i<n:
        s+=f*2
        f=f/2
        n-=1
    return s,f
height,n=100,3
d_n,h_n=ball(100,3)
print(f"初始高{height}米,第{n}次落地经过{d_n}米，第{n}次反弹高度为{h_n}米")
```

例 6.15 实现九九乘法表的函数，函数中不能输出打印。要求：

- (1) 每行的所有乘法式子构成一个列表
- (2) 所有行构成一个列表。
- (3) 根据上述 (1) (2) 得到的嵌套列表作为函数结果返回
- (4) 在主函数中调用函数，输出结果。

```
def multiplicationTableNew():
    table = [[f"{i + 1} * {j + 1} = {(i + 1) * (j + 1)}"
              for j in range(i + 1)] for i in range(9)]    #这是续行上一行
    return table
table = multiplicationTableNew()
for i in table:
    for j in i:
```

```
    print(j, end="\t")
print()
```

例 6.16 定义一个函数，实现阶乘

```
def fact(n):
    if n==0: #初始条件，终止递归
        return 1;
    else:
        return n*fact(n-1) #递归调用
print(fact(5))
```

例 6.17 实现斐波那契数列

```
def f(n):
    if n==1 or n==2: #终止递归条件（初始条件）
        return 1
    else:
        return f(n-1)+f(n-2) #递归调用
print([f(i) for i in range(1,10)])
```

例 6.18 使用递归实现归并排序算法

```
def mergeSort(list1):
    if len(list1) <= 1:
        return list1
    mid = int (len(list1)/2)
    left = mergeSort(list1[0:mid])
    right = mergeSort(list1[mid:len(list1)])
    return merge(left, right)

def merge(l, r): #将 l 和 r 两部分合并
    c = []
    i,j = 0,0
    while j < len(l) and i < len(r):
        if l[j] >= r[i]:
            c.append(r[i])
            i = i + 1
        else:
            c.append(l[j])
            j = j + 1
    for i in (l[j:] if i == len(r) else r[i:]):
        c.append(i)
    return c

list2 = [8, 4, 5, 7, 1, 3, 6, 2]
print (mergeSort(list2))
```

例 6.19 使用递归实现快速排序算法

```
def quicksort(array):
    less = [ ]
    more = [ ]
    if len(array) <= 1:
        return array
    p = array.pop( )
    for x in array:
        if x > p:
            more.append(x)
        else:
            less.append(x)
    return quicksort(less) + [p] + quicksort(more)
array1 = [21,32,42,53,34,25,57,54,78,89]
print (quicksort(array1))
```