第5章

函　数

任务一：了解函数

任务描述：

函数是组织好的、可重复使用的，用来实现单一或相关联功能的代码段。函数可以帮助程序员提高编程效率。学习函数前应了解函数的概念、函数的作用以及 Python 中函数的分类。

新知准备：

◇ 函数的概念与作用；

◇ Python 函数的分类。


视频讲解

5.1　函数概述

5.1.1　函数的概念

在计算机编程中，函数是一段可重复使用的代码，它可以接收输入并返回输出。函数可以被调用或执行，并且可以在程序中的任何地方使用。

使用函数是编程中提高代码质量、提高开发效率的重要手段。首先，函数能够提高代码的可读性和可维护性，通过将复杂的逻辑和操作封装起来，使程序结构更加清晰。其次，函数实现了代码的重用，可以在不同的程序部分中多次调用同一函数，减少代码冗余，提高开发效率。此外，函数还可以提高程序的模块化程度，将程序划分为多个独立的模块，各司其职，便于管理和维护。最后，函数有利于降低程序的错误率，通过封装特定功能的代码，可以减少在程序中出现错误的可能性，提高程序的稳定性。

5.1.2　Python 函数的分类

Python 提供了许多内置函数，如前面学习的 print()函数、input()函数、list()函数、tuple()函数等，用户也可以自己定义函数来实现特定的功能。Python 中的函数主要分为内置函数、标准库函数、第三方库函数以及自定义函数，如图 5-1 所示。

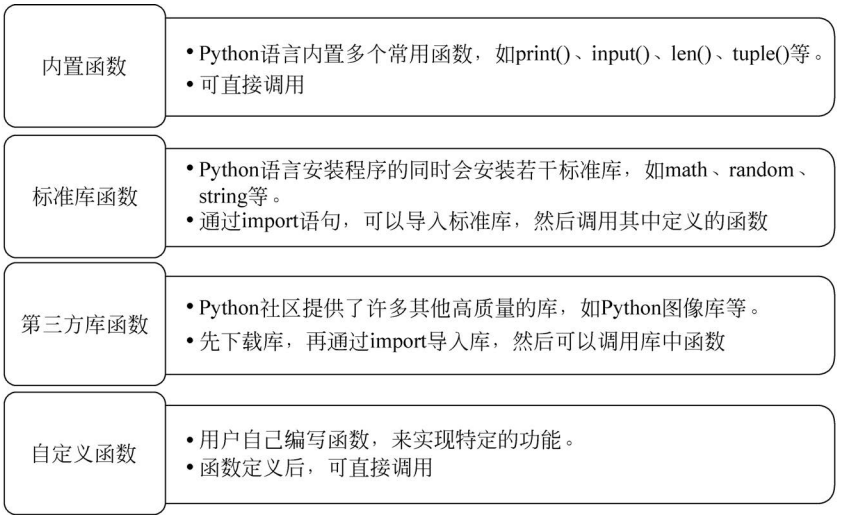

图 5-1　Python 函数的分类

任务二：求解一元二次方程

任务描述：

一元二次方程的一般形式是 $ax^2+bx+c=0(a\neq 0)$。其中，ax^2 是二次项，a 是二次项系数；bx 是一次项，b 是一次项系数；c 是常数项。当 $b^2-4ac>0$ 时，方程有两个不相等的实数根；当 $b^2-4ac=0$ 时，方程有两个相等的实数根；当 $b^2-4ac<0$ 时，方程无实数根。一元二次方程根的求解公式为

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

请编写程序，输入参数 a、b、c 的值输出方程的根。

新知准备：

◇ 函数的定义与调用；

◇ 函数的参数与变量的作用域。

5.2　函数的定义与调用

5.2.1　函数的定义

在 Python 中定义函数的格式如下：

```
def 函数名([形参列表]):
        函数体
```

关于函数定义的说明如下：

(1) 函数代码块以 def 关键词开头，后接函数名称和圆括号“()”。

(2) 函数名是用户自定义的函数名称。

(3) 形参列表可以是零个或多个参数，且任何传入参数必须放在圆括号内。

(4) 最后必须跟一个冒号(:)，函数体从冒号开始，并且缩进。

（5）函数体为实现函数功能的语句块。

（6）函数可以使用 return[表达式]语句返回值，函数中没有 return 相当于返回 None。

5.2.2 函数的调用

定义一个函数只给了函数一个名称，指定了函数里包含的参数和代码块结构。要实现函数所定义的功能，需要去调用函数。函数调用的格式如下：

```
函数名([实参列表])
```

【例 5-1】 定义函数，返回两个数之和。

```
def getSum(x,y):                    #函数定义
    return x + y
result = getSum(56,78)              #函数调用
print(result)                       #134
```

【例 5-2】 定义函数，打印出 n 个星号的无返回值函数。

```
def printStar(n):                    #函数定义
    for i in range(n):
        print("*",end = " ")
printStar(10)                        #函数调用 打印出 10 个 *
```

【例 5-3】 定义函数，返回 n 阶调和数($1+1/2+1/3+\cdots+1/n$)，要求将前 n 项的调和数都输出。

```
def harm(n):
    total = 0.0
    for i in range(1,n + 1):
        total += 1.0/i
    return total
x = 10
for j in range(1,x + 1):
    print("{}的 n 阶调和数为{}".format(j,harm(j)))          #循环调用函数
```

程序的运行结果如下所示。

```
1 的 n 阶调和数为 1.0
2 的 n 阶调和数为 1.5
3 的 n 阶调和数为 1.8333333333333333
4 的 n 阶调和数为 2.083333333333333
5 的 n 阶调和数为 2.283333333333333
6 的 n 阶调和数为 2.4499999999999997
7 的 n 阶调和数为 2.5928571428571425
8 的 n 阶调和数为 2.7178571428571425
9 的 n 阶调和数为 2.8289682539682537
10 的 n 阶调和数为 2.9289682539682538
```

【例 5-4】 定义函数，计算圆的面积，输入圆的半径，返回面积。

```
def getArea(r):
    s = 3.14 * r * r
    return s
print(getArea(15))                  #函数调用
```

【例 5-5】　定义函数，输入商品的数量及单价，输出总价。

```
def get_total(price,num):
    return price * num
print(get_total(5.5,3))            #单价 15.5,数量 3,运行结果 16.5
```

【例 5-6】　定义函数，接收字符串参数，返回一个元组，其中第一个元素为大写字母个数，第二个元素为小写字母个数。

```
def getUppLow(text):
    upp = low = 0
    for i in text:
        if i.isupper():
            upp += 1
        elif i.islower():
            low += 1
    return (upp,low)

str = input("请随机输入一组字符串")
result = getUppLow(str)
print("您输入的字符串为:",str)
print("大小写字母的个数分别为:",result)
```

【例 5-7】　定义函数，接收一个字符串，将字符串中非字母的字符去掉并返回。

```
def getLetter(text):
    new_text = ""
    for t in text:
        if t.isalpha():
            new_text += t
    return new_text

print(getLetter("ACDDD1233$ $ $123cdddf"))          #运行结果'ACDDDcdddf'
```

5.2.3　任务实现

1. 任务分析

要编写函数求解一元二次方程，首先确定函数的参数有 3 个，分别为 a、b、c，然后编写函数体，计算并输出结果。主要编程思路如下所示。

(1) 定义函数，确定形参为 a、b、c。

(2) 编写函数体，首先计算 b^2-4ac 并判断方程根的个数。

(3) 计算并返回方程的根。

(4) 调用函数，进行验证。

2. 编码实现

任务实现代码如下所示。

```
import math
def getResult(a,b,c):#ax**2+bx+c=0
    m = b*b-4*a*c
```

```
    if m >= 0:
        x1 = (-b + math.sqrt(m))/2 * a
        x2 = (-b - math.sqrt(m))/2 * a
        return (x1,x2)
    else:
        return "无解"
getResult(2,4,2)            # 函数调用:求解 2x ** 2 + 4x + 2 = 0 的根
```

任务三：了解函数的参数分类与变量的作用域

任务描述：

函数的形参和实参必须一一对应。Python 中函数的参数类型非常丰富，Python 中函数支持的参数类型包括位置参数、默认参数、关键字参数等，要求掌握并区分 Python 函数中的各种参数类型。

变量作用域指的是变量生效的范围，在 Python 中一共有两种作用域：局部变量与全局变量。要求掌握局部变量与全局变量，并掌握如何在函数体内修改全局变量。

新知准备：

◇ 函数的实参与形参；

◇ 必备参数、关键字参数、默认参数、不定长参数；

◇ 变量的作用域。


视频讲解

5.3 函数的参数与变量的作用域

5.3.1 函数的参数

在调用函数时，经常会用到形式参数和实际参数。其中，定义函数时所声明的参数，即为形式参数，简称形参；在调用函数时，提供函数所需要的参数的值，即为实际参数，简称实参。如图 5-2 所定义的函数，函数定义中的 x、y 为形参，函数调用时的 56、78 为对应的实参。

```
def getSum(x,y):        形参
    return x+y
result=getSum(56,78)    实参
```

图 5-2 实参与形参

1. 必备参数

必备参数须以正确的顺序传递函数。调用时的数量必须和声明时的一样，如 print("hello world",end="\t")。

【例 5-8】 定义函数，接收一个字符串，一个整数 n，实现功能：将用户传入的字符串打印 n 遍。

```
def printText(str,n):
    for i in range(n):
        print(str)
printText("good morning!",3)          # 打印三遍"good morning!"
printText(3 ,"good morning!")         # 因为参数的顺序传错,程序报错
```

2. 关键字参数

关键参数主要指实参，即调用函数时的参数传递方式。

通过关键参数，实参顺序可以和形参顺序不一致，但不影响传递结果，避免了用户需要牢

记位置参数顺序的麻烦。

在例 5-8 中，因为参数传递顺序错误导致程序无法正常运行，为避免实参顺序与形参顺序不一致，将例 5-8 中最后一行代码进行修改。

```
def printText(str,n):
          for i in range(n):
                print(str)
    printText("good morning!",3)              #打印三遍"good morning!"
printText(n = 3 ,str = "good morning!" )       #通过关键字进行参数传递，程序正常运行
```

3. 默认参数

调用函数时，默认参数的值如果没有传入，则被认为是默认值。

例如，在定义 printText()函数时设置 str 的默认值为“hello”。

```
def printText(n,str = "hello"):
    for i in range(n):
        print(str)
printText(3)                                  #未传入 str，则用默认值，输出三遍 hello
printText(2,"Good!")
```

注意：默认值参数只在函数定义时被解释一次，且默认值参数放在后面。

4. 不定长参数

可变长度参数主要有两种形式：

* parameter 用来接收多个实参并将其放在一个元组中；

** parameter 接收关键参数并存放到字典中。

【例 5-9】 定义函数，输出多个参数之和，但是不确定参数的个数。

```
def getSum_X( * p):
    return sum(p)
print(getSum_X(1,2,3,4))                      #运行结果:10
```

【例 5-10】 将字典作为函数参数。

```
def printInfo( ** d):
    for item in d.items():
        print(item)
info = {"name":'Lily',"age":20,"City":"ShangHai"}
printInfo( ** info)
```

程序运行结果为

```
('name', 'Lily')
('age', 20)
('City', 'ShangHai')
```

5.3.2 变量作用域

一个程序的所有变量并不是在哪个位置都可以访问的。访问权限决定于这个变量是在哪里赋值的。

根据变量的作用域变量分为两种：全局变量与局部变量。局部变量是指在函数内部定义的变量，它只在函数内部有效。全局变量是指在函数外部定义的变量。

1. 局部变量

在函数内部定义的普通变量只在函数内部起作用，称为局部变量。当函数执行结束后，局部变量自动删除，不可以再使用。

【例 5-11】 局部变量的访问。

参数 price 在函数体内定义属于局部变量，在函数体内可以正常访问，在函数体外方式时则会报错，错误信息为：NameError：name 'price' is not defined。

```
def get_Total(n):
    price = 18
    total = n * price
    return total
print(price)                    #在函数体外访问局部变量,报错
```

2. 全局变量

能够同时作用于函数内外，称为全局变量，可以通过 global 来定义。具体分为以下两种情况。

（1）一个变量已在函数外定义，如果在函数内需要为这个变量赋值，并要将这个赋值结果反映到函数外，可以在函数内用 global 声明这个变量，将其声明为全局变量。

（2）在函数内部直接将一个变量声明为全局变量，在函数外没有声明，该函数执行后，将增加为新的全局变量。

【例 5-12】 全局变量的访问。

```
price = 18
  def get_Total(n):
      global price          #在函数内部使用全局变量用 global 声明
      return n * price
  t = get_Total(10)
  print(t)                  #180
print(price)                #18 全局变量,访问不再报错
```

任务四：实现斐波那契数列

任务描述：

斐波那契数列（Fibonacci sequence），又称黄金分割数列，因数学家莱昂纳多·斐波那契（Leonardo Fibonacci）以兔子繁殖为例而引入，故又称“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：$F(0)=1, F(1)=1, \cdots, F(n)=F(n-1)+F(n-2)(n\geqslant 2, n\in \mathbf{N}^*)$。

请定义函数实现斐波那契数列。

新知准备：

◇ 递归函数。

5.4 递归函数

视频讲解

5.4.1 递归函数的基本用法

在函数内部，可以调用其他函数。如果一个函数在内部调用函数本身，这个函数就是递归函数。

每个递归函数必须包括终止条件与递归步骤两个主要部分。

(1) 终止条件。表示递归的结束条件，用于返回函数值，不再递归调用。例如，阶乘 $f(n)$ 的结束条件为“n 等于 1”。

(2) 递归步骤。递归步骤把第 n 步的参数值的函数与第 $n-1$ 步的参数值的函数关联。例如，对于阶乘 $f(n)$，其递归步骤为“$n * f(n-1)$”。

【例 5-13】 使用递归函数实现调和数。

$$H(n)=1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n} \tag{5-1}$$

式(5-1)为 n 阶调和数的公式，要通过递归函数实现，分析其终止条件为：$H_1=1$；递归步骤为 $H_n=H_{n-1}+1/n$。

```
def harmonic(n):
    if n == 1:
        return 1.0                    #终止条件
    return harmonic(n-1) + 1.0/n      #递归步骤
#函数调用
for i in range(1,6):                  #输出 1～5 阶的调和数
    print('H',i,' = ', harmonic(i))
```

程序运行结果如下所示。

```
H1 = 1.0
H2 = 1.5
H3 = 1.8333333333333333
H4 = 2.083333333333333
H5 = 2.283333333333333
```

【例 5-14】 使用递归函数实现数字的逆序输出。

```
def reverse(n):
    if n!= 0:
        print(n % 10,end = '')
        reverse(n//10)
print(reverse(123456789))
```

程序运行结果如下所示。

```
请输入一个整数: 123456789
987654321
```

5.4.2 任务实现

1. 任务分析

斐波那契数列的规则是1,1,2,3,5,8,13,…,即$F(0)=1$,$F(1)=1$,后面每项的值等于前面两项之和。要通过递归函数来实现斐波那契数列的第n个值,并输出斐波那契数列的前n项。主要编程思路如下所示。

(1) 定义递归函数,确定函数的名字与形参。

(2) 编写函数体,确定终止条件与递归步骤。终止条件为$F(0)=1$或者$F(1)=1$;递归步骤:$F(n)=F(n-1)+F(n-2)$。

(3) 调用函数,输出斐波那契数列的前n项。

2. 编码实现

```
def fib(n):
    if n == 1 or n == 2:
        return 1
    return fib(n - 1) + fib(n - 2)

for i in range(1,11):
    print(fib(i),end = "\t")
#程序的运行结果为: 1 1 2 3 5 8 13 21 34 55
```

任务五：实现词频排序

任务描述：

在文本分析中,经常需要按词频对文本进行排序,用户一般先通过生成字典的方法统计词的频次,然后给字典排序。

那么如何快速地给字典按照键值进行排序呢?

技术准备：

◇ 匿名函数。

5.5 匿名函数

5.5.1 匿名函数的基本用法

lambda表达式,又称匿名函数,常用来表示内部仅包含1行表达式的函数。如果一个函数的函数体仅有1行表达式,则该函数就可以用lambda表达式来代替。lambda表达式的语法格式如下：

```
name = lambda [list] : 表达式
```

其中,定义lambda表达式,必须使用lambda关键字;[list]作为可选参数,等同于定义函数是指定的参数列表;name为该表达式的名称。

相比函数，lambda 表达式具有以下两个优势：

（1）对于单行函数，使用 lambda 表达式可以省去定义函数的过程，让代码更加简洁；

（2）对于不需要多次复用的函数，使用 lambda 表达式可以在用完之后立即释放，提高程序执行的性能。

【例 5-15】 使用 lambda 表达式，求两数之和。

```
s = lambda x,y:x + y
print(s(4,9))
```

【例 5-16】 使用 lambda 表达式，求两数中的较大值。

```
m = lambda x,y : x if x > y else y
print(m(2,5))
```

5.5.2 与 map()函数结合

Python 中的 map()函数用于将一个函数应用于一个序列的所有元素，并返回一个迭代器。如果该序列的元素不能被函数处理，则抛出 TypeError 异常。

map()函数的语法格式如下：

```
map(function, iterable, …)
```

map()函数的主要参数及其含义如下所述。

（1）function：表示要应用于序列的函数，该函数需要接收一个或多个参数，并返回一个结果。

（2）iterable：表示一个序列，可以是列表、元组、集合等，序列中的每个元素都将被函数处理。

【例 5-17】 返回元素平方的可迭代对象。

```
it = map(lambda x:x * x, range(10))
print(it)
print(list(it))
#结果为:[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

5.5.3 与 filter()函数结合

Python 中的 filter()函数用于将一个函数应用于一个序列的所有元素，并根据该函数返回一个迭代器，该迭代器包含所有使得函数返回 True 的元素。

filter()函数的语法格式如下：

```
filter(function, iterable, …)
```

filter()函数的参数与 map()函数的参数含义与用法相似，在这里不做赘述。

【例 5-18】 过滤列表，返回元素为奇数的可迭代对象。

```
it = filter(lambda x:x % 2 == 1,[i for i in range(1,11)] )
print(it)
print(list(it))          #结果为[1, 3, 5, 7, 9]
```

5.5.4 与 reduce()函数结合

Python 中的 reduce()函数用于将一个二元函数应用于一个序列的所有元素，并逐步将序列中的元素缩减为单一的值。

reduce()函数的语法格式如下：

```
reduce(function, iterable, initial = None)
```

reduce()函数的前两个参数与 map()函数的参数含义与用法相似，在这里不做赘述。其中，第三个参数 initial 表示缩减的初始值，如果提供，则第一次调用函数时使用该值作为第一个参数；如果未提供初始值，则使用序列的第一个元素作为第一个参数，序列的第二个元素作为第二个参数。

【例 5-19】 返回列表元素之积。

```
from functools import reduce
num = [1, 2, 3, 4, 5]
result = reduce(lambda x, y: x * y,num)
print(result)                    #运行结果 120
```

5.5.5 任务实现

1. 任务分析

要实现字典按照键值进行排序，需要用到 Python 的内置 sorted()函数。主要编程思路如下所示。

(1) 确定要进行排序的可迭代对象：调用字典对象的 items()函数以列表返回可遍历的键与值组成的元组数据。

(2) 确定排序规则：按照每个键所对应的值的大小，从大到小进行排序。

(3) 调用 sorted()函数进行排序，并返回排序后的数据。

2. 编码实现

任务实现代码如下所示。

```
dic = {'Java': 3, 'Python': 8, 'Big Data': 5, 'AI': 2}
before = dic.items()
print("排序前数据:",list(before))
after = sorted(before, key = lambda x: x[1],reverse = True)
print("排序后数据:",after)
```

运行结果如下所示。

```
排序前数据: [('Java', 3), ('Python', 8), ('Big Data', 5), ('AI', 2)]
排序后数据: [('Python', 8), ('Big Data', 5), ('Java', 3), ('AI', 2)]
```

5.6 本章实践

实践一：求两个数的最小公倍数

最小公倍数是指两个或多个整数共同拥有的最小公倍数。例如，对于整数 4 和 6，它们的公倍数有 4、6、12、18 等，其中最小公倍数是 12。最小公倍数在解决涉及时间、频率和协调的问题时具有重要意义。

请编写程序，实现输入两个整数，返回它们的最小公倍数。

实现思路如下所示。

(1) 定义一个函数，用于计算最小公倍数。这个函数需要两个参数，分别是两个整数 x 和 y。

(2) 在函数内部，首先找出两个数中较大的一个。然后从较大的数开始，逐个检查直到找到能够同时被 x 和 y 整除的数，这个数就是它们的最小公倍数。

(3) 返回找到的最小公倍数。

(4) 调用函数并输出结果。

任务实现代码如下所示。

```
def get_common_mul(x,y):
    num_max = max(x,y)
    for i in range(num_max,x * y + 1):
        if i % x == 0 and i % y == 0:
            return i
print(get_common_mul(4,6))          #结果为 12
```

实践二：解决猴子吃桃问题

猴子吃桃问题是一个经典的数学问题，这个问题可以帮助学生理解递归概念。问题是这样描述的：猴子第一天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个；第二天早上又将剩下的桃子吃掉一半，又多吃了一个。以后每天早上都吃前一天剩下的一半零一个。到第 10 天早上想再吃时，见只剩下一个桃子了。求第一天共摘了多少个桃子？

请编写程序，利用递归的思想，计算猴子从第 2 天到第 10 天桃子的数量。

实现思路如下所示。

(1) 定义一个函数，用于计算第 n 天结束时猴子剩下的桃子数量。

(2) 在函数中，确定终止条件：如果 n 等于 1，则返回 1；确定递归步骤：如果 n 大于 1，则返回(peach($n-1$)+1) * 2。

(3) 调用函数并输出结果。

任务代码如下：

```
def peach(n):
    if n == 10:
        return 1
    else:
        return (peach(n + 1) + 1) * 2
for i in range(10,0, - 1):
    print("第{}天有{}个桃子".format(i,peach(i)))
```

程序运行结果如下所示。

```
第 10 天有 1 个桃子
第 9 天有 4 个桃子
第 8 天有 10 个桃子
第 7 天有 22 个桃子
第 6 天有 46 个桃子
第 5 天有 94 个桃子
第 4 天有 190 个桃子
第 3 天有 382 个桃子
第 2 天有 766 个桃子
第 1 天有 1534 个桃子
```

实践三：解决自由落体问题

自由落体问题是物理学中的基本问题，它是许多物理学原理的基础，如牛顿运动定律。对自由落体运动的研究，有助于理解物体在重力场中的运动规律，并且在工程、天文学等领域都有广泛的应用。

请编写程序，解决问题：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 10 次落地时反弹的高度和经过的总距离。

实现思路如下所示。

(1) 定义一个全局变量，其数据类型为列表，用于记录每次下落经过的距离。

(2) 定义一个函数，用于计算球在第 n 次落地后反弹的高度。

(3) 在函数中，确定终止条件：如果 n 为 1，则返回 50；确定递归步骤：如果 $n>1$，则反弹高度为 jump(n－1)/2，并且每次将经过的距离添加到列表中。

(4) 调用函数，计算第 10 次落地后的反弹高度，并统计经过的总距离。

任务实现代码如下所示。

```
dis = [100]
def jump(n):
    global dis
    if n == 1:                    #第一次落下 100 米,然后反弹 50 米
        dis.append(100)
        return 50
    else:
        fantan = jump(n-1)/2      #每次反弹高度为上次的一半
        dis.append(fantan*2)
        return fantan
print("第 10 次反弹的高度为",jump(10))
print("第 10 次落地时,共经过的距离",sum(dis[:-1]))   #注意第 10 次落地不包括列表中的最后
                                                     #一个元素
```

程序运行结果如下所示。

```
第 10 次反弹的高度为 0.09765625
第 10 次落地时,共经过的距离 299.609375
```

实践四：验证哥德巴赫猜想

哥德巴赫猜想是数学上的一个著名未解之谜，该猜想的内容为：任意一个大于 2 的偶数，

都可以表示为两个质数之和。例如，6＝3＋3，8＝3＋5，10＝3＋7，12＝5＋7 等。

请编写程序，验证 2000 以内的偶数都可以分解成两个质数之和。

实现思路如下所示。

（1）定义质数判断函数，判断一个数是否为质数，是则返回 True，不是则返回 False。

（2）定义哥德巴赫猜想，首先检查输入的 n 是否是合法的，然后遍历从 3 到 $n/2$ 的所有质数，检查这个偶数是否可以分解成两个质数之和，若能则打印出分解式。

（3）编写循环依次验证。

任务代码如下：

```
def isSushu(x):
    flag = 0
    for i in range(2,x):
        if x % i == 0:
            flag = 1
            return False
    if flag == 0:
        return True

def gold(n):
    if n % 2!= 0 or n < 6:
        print("您输入的数据是不合法的")
        return 0
    for i in range(3,n//2,2):
        j = n - i
        if isSushu(i) and isSushu(j):
            print(n," = ",i," + ",j,end = " ")
            print('对于数字{}猜想成立'.format(n))
            break
for i in range(6,2001,2):
    gold(i)
```

程序运行结果如下所示。

```
8 = 3 + 5 对于数字 8 猜想成立
10 = 3 + 7 对于数字 10 猜想成立
12 = 5 + 7 对于数字 12 猜想成立
14 = 3 + 11 对于数字 14 猜想成立
16 = 3 + 13 对于数字 16 猜想成立
18 = 5 + 13 对于数字 18 猜想成立
20 = 3 + 17 对于数字 20 猜想成立
……
```

5.7 本章习题

一、选择题

1. 使用(　　)关键字来创建 Python 自定义函数。

A. function　　B. func　　C. procedure　　D. def

2. 关于函数参数传递中，形参与实参的描述错误的是(　　)。

A. Python 实行按值传递参数。值传递指调用函数时将常量或变量的值(实参)传递给函数的参数(形参)

B. 实参与形参存储在各自的内存空间中，是两个不相关的独立变量

C. 在参数内部改变形参的值，实参的值一般是不会改变的

D. 实参与形参的名字必须相同

3.（　　）表达式是一种匿名函数。

A. lambda　　B. map　　C. filter　　D. zip

4.（　　）函数用于将指定序列中的所有元素作为参数调用指定函数，并将结果构成一个新的序列返回。

A. lambda　　B. map　　C. filter　　D. zip

5. 关于函数的下列说法不正确的是（　　）。

A. 函数可以没有参数　　B. 函数可以有多个返回值

C. 函数可以没有 return 语句　　D. 函数都有返回值

二、判断题

1. 在函数内部，既可以使用 global 保留字声明使用外部全局变量，也可以使用 global 保留字直接定义全局变量。（　　）

2. lambda 表达式中可以使用任意复杂的表达式，但是只能编写一个表达式。（　　）

3. 定义 Python 函数时，如果函数中没有 return 语句，则默认返回空值 None。（　　）

4. 函数内部定义的局部变量当函数调用结束后被自动删除。（　　）

5. 一个函数如果带有默认值参数，那么所有参数都要设置默认值。（　　）

6. 调用函数时，可以通过关键参数的形式传递值，从而避免必须记住函数形参顺序的麻烦。（　　）

7. 函数中的 return 语句一定能够被执行。（　　）

8. 在定义函数时，某个参数名字前面带有一个 * 符号表示可变长度参数，可以接收任意多个普通实参并存放于一个元组之中。（　　）

9. 在函数内部不能定义全局变量，只能定义局部变量。（　　）

10. 定义函数时，即使该函数不需要接收任何参数，也必须保留一对空的圆括号来表示这是一个函数。（　　）

三、编程题

1. 定义函数，输入一元二次方程的系数 a、b、c，输出方程式的根。

2. 定义函数，输入任意字符串，统计其中元音字母（'a'、'e'、'i'、'o'、'u'，不区分大小写）出现的次数和频率。

3. 定义函数，实现随机产生一个 8 位的随机密码，密码包括字母、数字及下画线，调用函数产生 10 组密码，存放在列表中。