第 3 章

算 法 探 究

数据和算法是信息时代计算机领域的核心概念，计算机解决问题的过程就是依据一定的步骤和方法对数据进行各种运算，最后得到满意结果及输出的过程。正因如此，算法在计算机领域占据着非常重要的地位。计算机领域的算法既包括传统数学中的算法精华，还包括计算机领域一些特定的数值处理方法。算法在算法竞赛中用得较多，也沉淀下来很多经典的算法。例如枚举算法、递推算法、递归算法、贪心算法、分治算法、探索回溯算法、动态规划算法等。由于本书面向的读者并非专业的信息学竞赛者，因此本章所讲解的算法主要以普及概念和强化前面所学知识为目的。具体内容可以分为三部分：第一部分讲解计算机常规的数值和字符处理方法；第二部分讲解简单算法，包含枚举和排序算法；第三部分通过两道题目简单讲解递推算法和贪心算法，让读者体验复杂算法的魅力，其余的算法读者如果感兴趣可以自行查阅。

3.1 序列求和

已知：$S_n=1+1/2+1/3+\cdots+1/n$（S_n 代表前 n 项和）。显然对于任意一个整数 k，当 n 足够大的时候，S_n 大于 k。现给出一个整数 $k(1\leqslant k\leqslant 15)$，要求计算出一个最小的 n，使 $S_n>k$。

分析：数列中的每一项的形式都为 $1/n$，因此可声明一个变量 S_n 存储每次相加的结果，然后用 S_n 和 k 比较即可，如果 S_n 大于 k，则返回 n 即可，代码如下：

```
//第 3 章/3.1
k = int(input("请输入整数 k:"))
Sn = 0
i = 1
while True:
    Sn = Sn + 1.0/i
    if Sn > k:
        print(i)
    #结束循环
        break
    else:
        #推导下一个数的分母
        i = i + 1
```

程序运行的结果如图 3-1 所示。

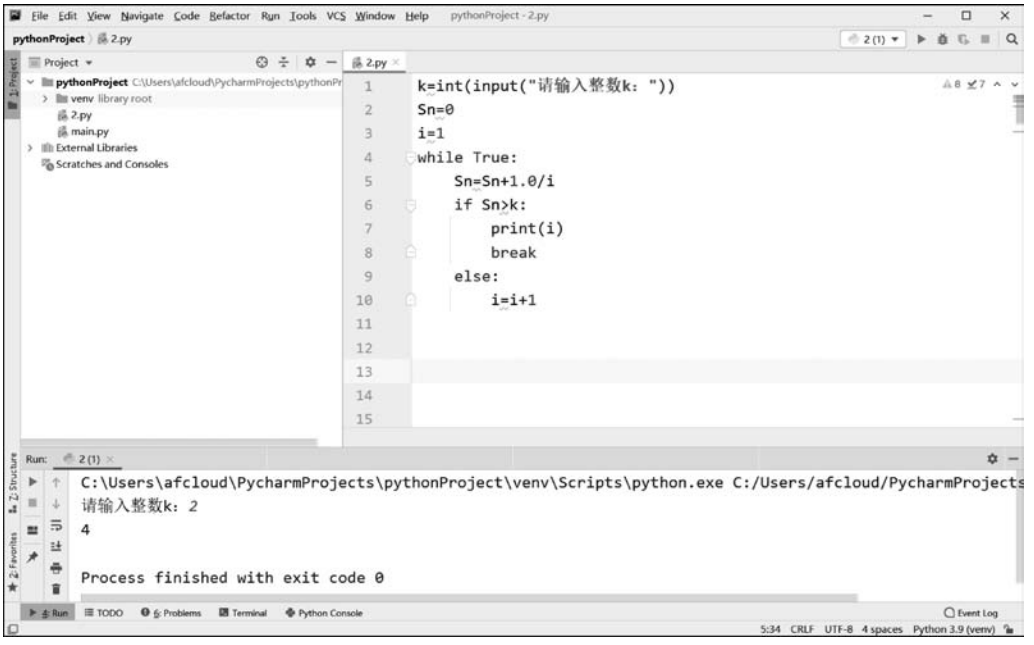

图 3-1　序列求和执行结果

3.2　水仙花数

题目：水仙花数是指一个三位数，它各个数位上数字的立方和等于它本身，例如 $153=1^3+5^3+3^3$，现在请编写程序找出所有的水仙花数。

分析：水仙花数是一道经典的数位分离题目，对数据进行处理的时候，有时需要摘取它各个数位的数进行计算，那么如何分离呢？这就要需要用到第 2 章学过的算术运算符。

Python 常见的算术运算符有“+”“-”“×”“/”“%”“//”，这里要用到的运算符主要为求余(%)、取整(//)这两个运算符。对于任意的一个三位数 a，取百位数字就是用 a 除以 100，得到的整数部分就是百位数字。个位数字的求法就是用 a 除以 10，最后剩下的余数就是个位数字。十位数字的求法比较灵活，可以用 a 除以 100，将所得的余数再除以 10 取整，也可以用 a 除以 10，将所得的整数部分再对 10 求余，所得的余数就是十位数字。程序的代码如下：

```
//第 3 章/3.2
for i in range(100,1000):
    #求百位数字 a
    a = i//100
    #求十位数字 b
    b = i % 100//10
    #求个位数字 c
    c = i % 10
```

```
    #判断是否符合条件
    if a**3+b**3+c**3==i:
        print(i)
```

程序执行的结果如图 3-2 所示，最后找到的水仙花数共 4 个，分别是 153、370、371、407。希望通过此道题目，使读者能够理解和掌握如何分离任意一个数字的各个数位。

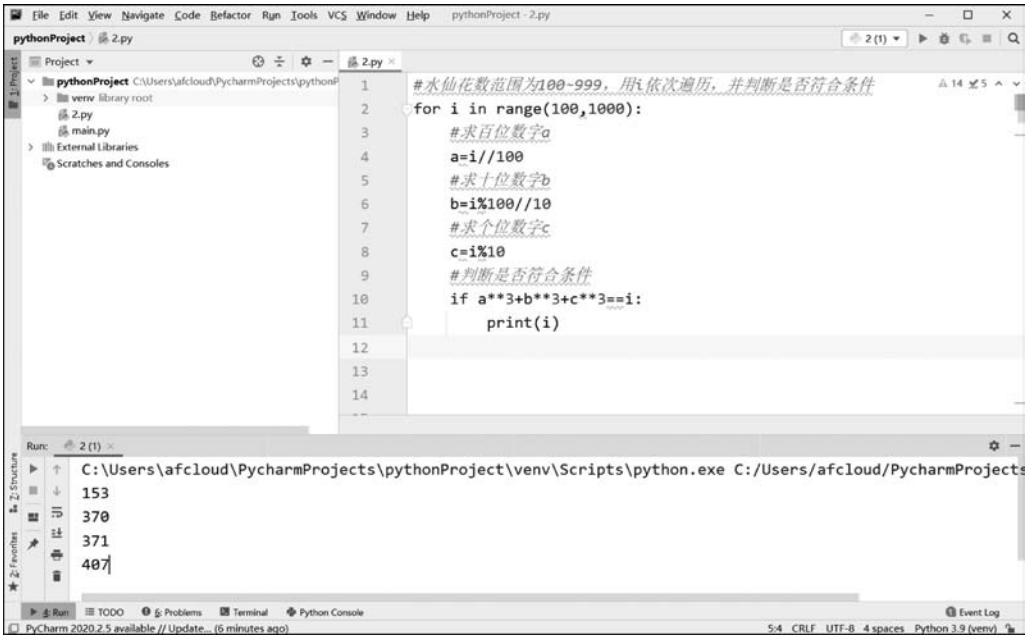

图 3-2　水仙花数执行结果

3.3　字符统计

题目：输入一段英文，分别统计其中数字、英文字母和其他符号的个数。

分析：这道题需要声明一个字符串变量，用来存储输入的内容，而后遍历，依次判断每个字符属于哪一种，然后分别计数。程序代码如下：

```
//第 3 章/3.3
#声明变量 n,用于存储输入的字符串
n = input("请输入一段英文:")
#声明 3 个变量,分别用来存储数字、字母、其他符号出现的次数
a,b,c = 0,0,0
for i in n:
    if i >= "0" and i <= "9":
        a = a + 1
    elif (i >= "A" and i <= "Z") or (i >= "a" and i <= "z"):
        b = b + 1
    else:
        c = c + 1
```

```
print("数字个数:",a)
print("字母个数:",b)
print("其他符号个数:",c)
```

程序运行的结果如图 3-3 所示。

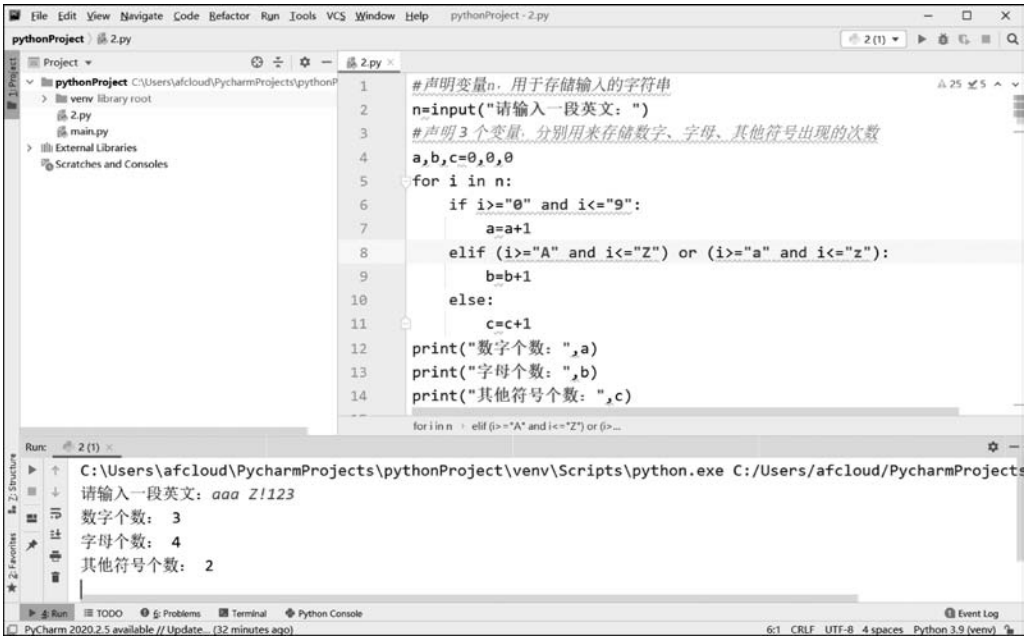

图 3-3　字符统计执行结果

3.4　鸡兔同笼

题目：数学中经典的“鸡兔同笼”问题，已知笼子中同时装有一些鸡和兔子，现在请输入两个数 m、n，分别代表笼子中头和脚的个数，求笼中的鸡和兔子各有多少只？

分析：假设有 a 鸡只，有 b 只兔子。那么 a 和 b 一定为小于或等于 m 的数，同时需要满足 $a+b=m$ 和 $2\times a+4\times b=n$，代码如下：

```
m = int(input("请输入头的数量:"))
n = int(input("请输入脚的数量:"))
for a in range(m + 1):
    for b in range(m + 1):
        if a + b == m and a * 2 + 4 * b == n:
            print("鸡:",a,"兔子:",b)
```

程序的执行结果如图 3-4 所示。

鸡兔同笼题目实际上是一道枚举算法的题目，枚举算法本质上就是列举出所有可能的解，然后逐个判断是否是需要的答案。实际上除鸡兔同笼还有很多问题应用枚举的思想，例如密码暴力破解、方程组求解、人民币组合凑数、钥匙开锁等。

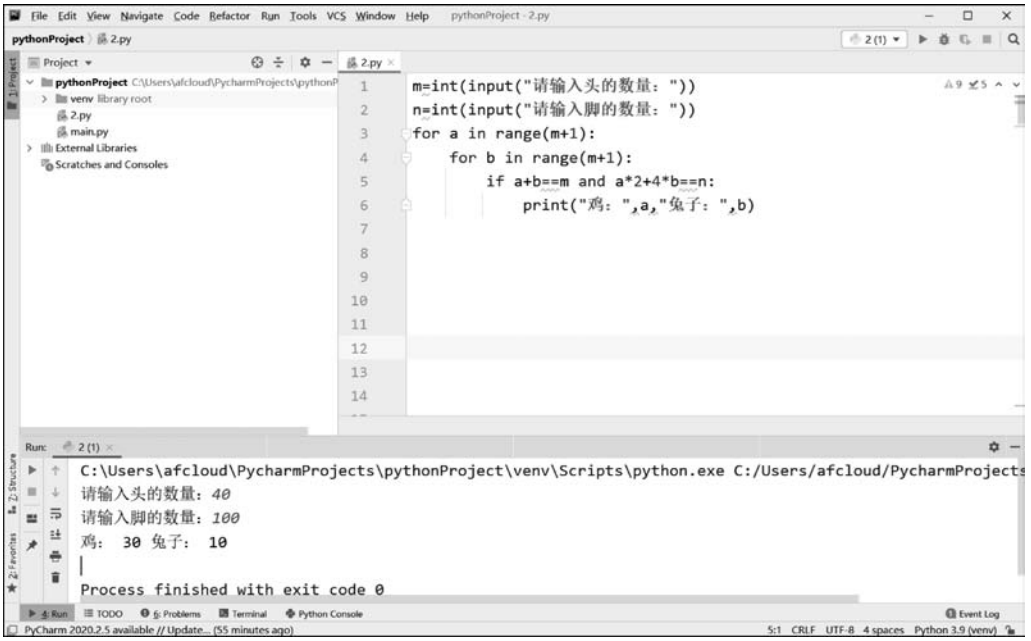

图 3-4　鸡兔同笼执行结果

3.5　最大质因数

题目：已知正整数 n 是两个不同质数的乘积，输入 n，试求出较大的那个质数。

分析：这道题要找到 n 的因子 a，然后判断 a 是不是质数并且是最大的质数，所以可以利用枚举思想，从 $n-1$ 开始反向列举，逐一检验，直到找到有效解，如果找不到，则输出提示信息。同时题目要找最大质因数，这里可以结合前面学过的函数知识，自定义一个函数，用来判断是否为质数（除 1 和它本身，不能被任何数整除）。程序代码如下：

```
//第 3 章/3.5
#自定义判断质数函数
#如果是质数,则返回 1; 如果不是质数,则返回 0
def zhishu(n):
    for i in range(2,n):
        if n % i == 0:
            return 0
    return 1

m = int(input("请输入一个正整数:"))
#反向逐一列举,如果找到答案,则结束

for i in range(m - 1,2, - 1):
    #判断:既是因数又是质数
    if m % i == 0 and zhishu(i) == 1:
        print(i)
        break
```

程序的执行结果如图 3-5 所示。

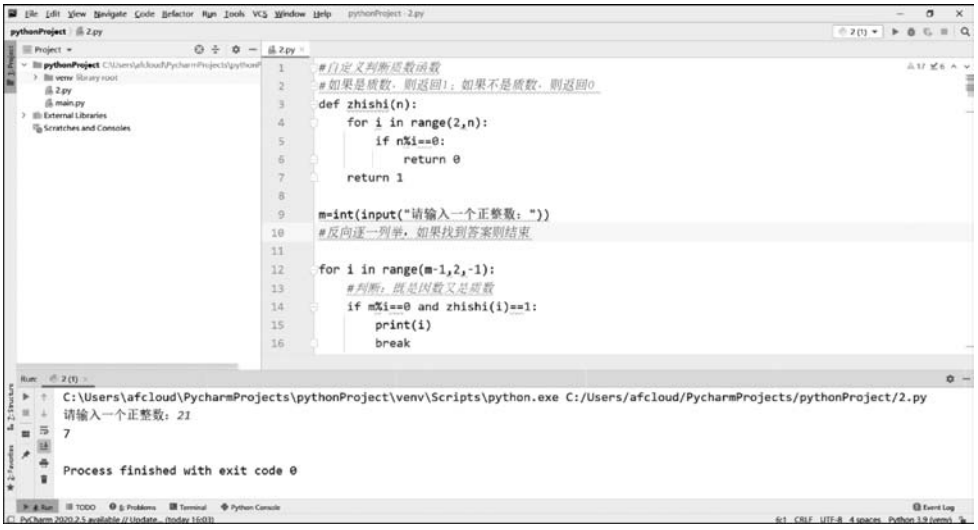

图 3-5　最大质因数执行结果

3.6　排序算法

题目：请输入 10 个数字，数字之间以空格分隔，代表某次考试 10 个同学的分数，现请将分数按照从大到小的顺序输出。

分析：由于输入的数字中间有空格，所以肯定不能直接作为数字来处理，需要用字符串来存储。存储后还需要将输入的字符转化为数字，而后再进行排序。转化的程序代码如下：

```
//第 3 章/3.6/字符串转数字列表
#声明字符串变量 a,用于存储输入的字符,并用 split 方法去掉空格
#去掉空格后将 a 转换为列表类型,里面存储的内容为字符型的数字
a = input().split()
print(a)
#字符型数字需要转换为整型,这样才可以进一步排序
a = list(map(int,a))
print(a)
```

程序的执行结果如图 3-6 所示，经过转化后列表 a 中的数据已经变成 3 个整数，可以进行下一步排序操作。

排序的方法有很多，例如选择排序、插入排序、冒泡排序等。这里介绍两种方法，一种是 Python 自带的排序方法，另一种是选择排序法。

方法一：采用 Python 自带的 sorted()方法排序，代码如下：

```
#将输入的字符串转换为整数列表
a = list(map(int,input("请输入 10 个数,之间以空格分隔:").split()))
#调用内置的 sorted 方法排序,reverse = True 表示从大到小排序
print(sorted(a,reverse = True))
```

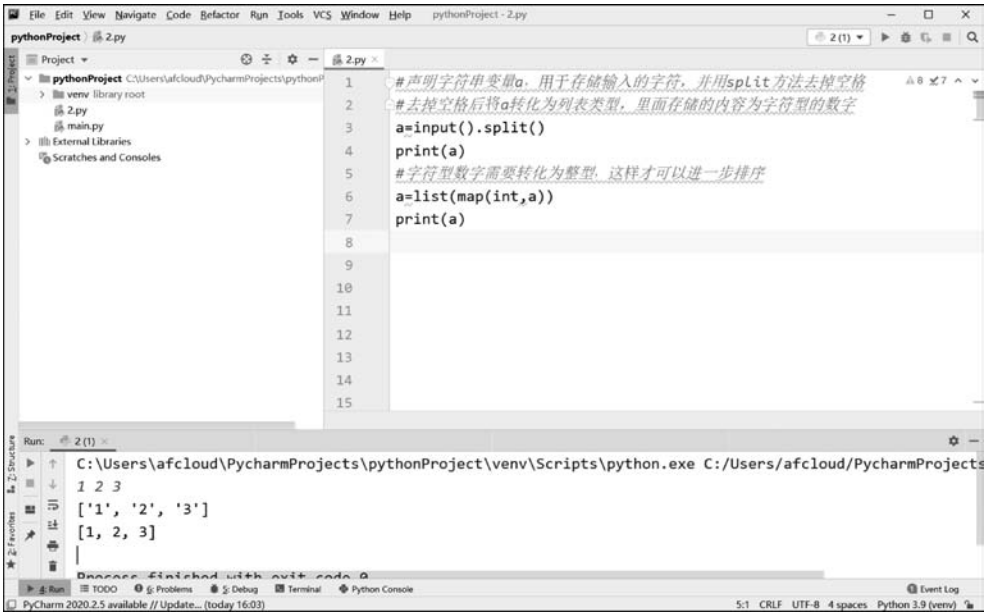

图 3-6　字符转数字执行结果

程序运行的效果如图 3-7 所示。

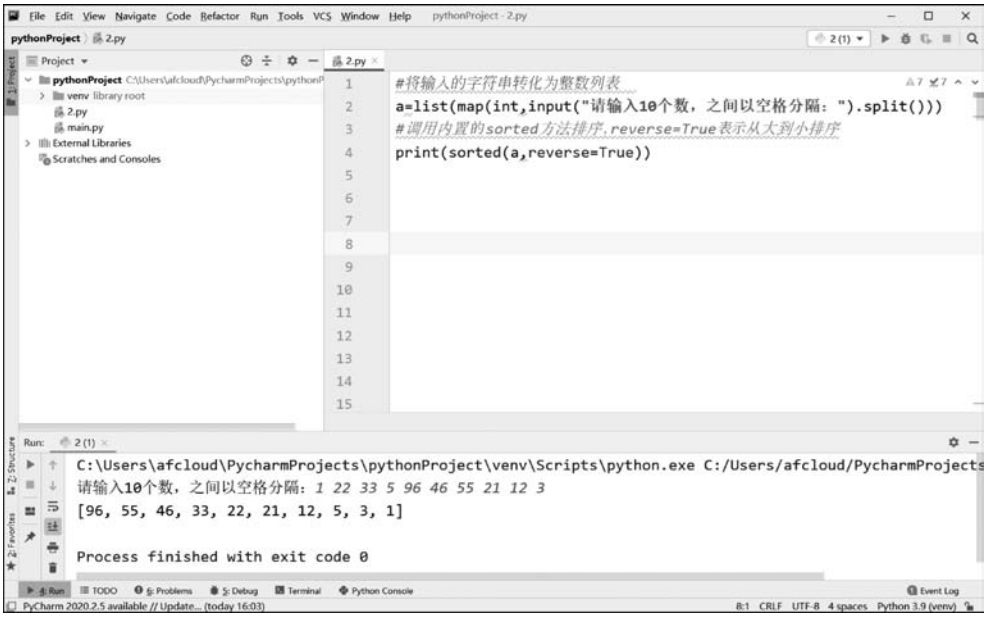

图 3-7　sorted 排序执行结果

方法二：采用选择排序法，代码如下：

```
//第 3 章/3.6/选择排序
#将输入的字符串转换为整数列表
a = list(map(int,input("请输入 10 个数,之间以空格分隔:").split()))
```

```
#查找 a 中的最大数,用 max 方法,找到后将较大数加入新列表,将原列表删除
b = []
#查找 10 次
for i in range(10):
    maxi = max(a)
    b.append(maxi)
    a.remove(maxi)
print(b)
```

程序的执行结果如图 3-8 所示。

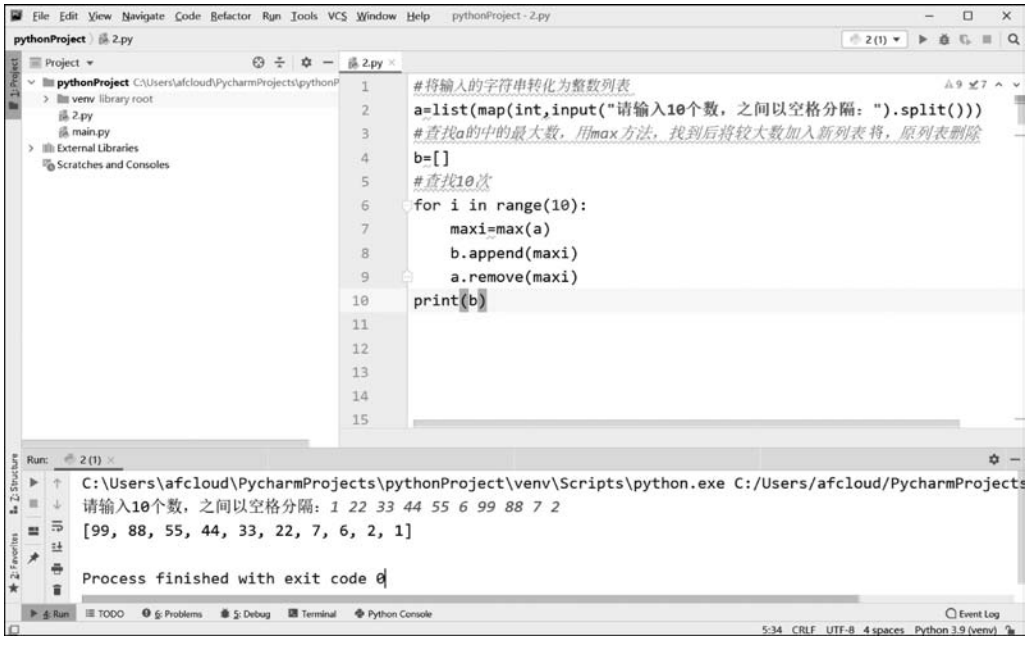

图 3-8　选择排序法的执行结果

3.7　递推算法

题目：假设 A 正在爬楼梯，需要 n 阶才能到达楼顶。A 每次可以爬 1 或 2 个台阶。有多少种不同的方法可以爬到楼顶呢？

分析：爬 1 级台阶有 1 种方法，爬 2 级台阶有两种方法，爬 3 级台阶有 3 种方法，爬 4 级台阶有 5 种方法……实际上对于第 n 级台阶，到达它的途径只有两种，要么从 $n-1$ 级台阶迈一步，要么从 $n-2$ 迈一步，因此到达 n 的方法恰好等于到 $n-1$ 和 $n-2$ 的方法数量之和。得到这个结论，可以先返回去验算，验算无误后说明这是一个递推问题。只要知道前两项，就可以递推出第 n 项，代码如下：

```
//第 3 章/3.7
n = int(input("请输入台阶数:"))
if n <= 2:
```

```
    print(n)
else:
    a = 1
    b = 2
    for i in range(3,n + 1):
        c = a + b
        a = b
        b = c
    print(b)
```

程序执行的效果如图 3-9 所示。

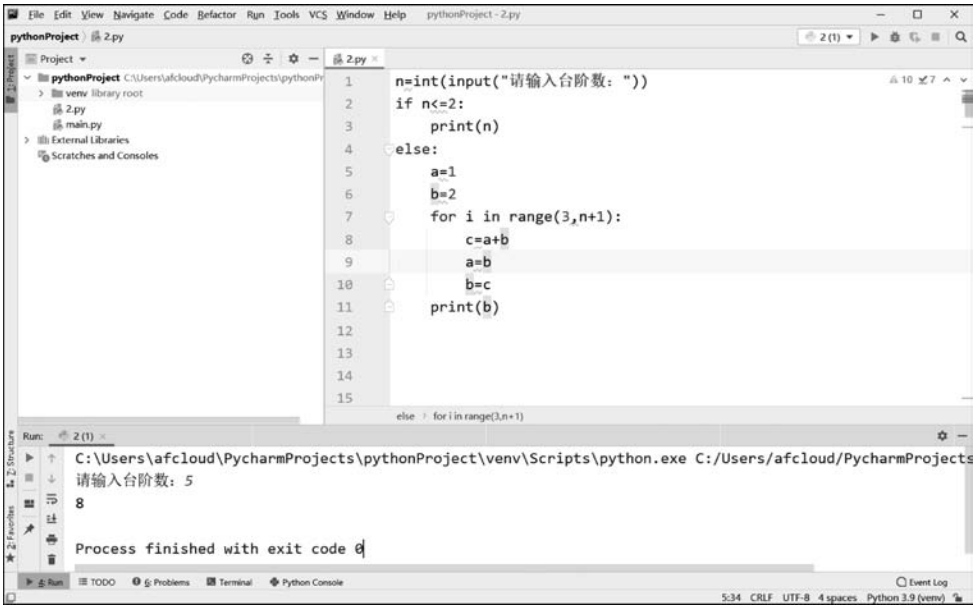

图 3-9　递推算法执行结果

递推算法充分发挥了计算机的计算优势，避开求通项公式的烦恼。竞赛中有很多数学问题可以用递推算法来解决，例如斐波那契数列、汉诺塔问题、骨牌铺法问题、平面分割问题等，有兴趣的读者可以进一步探究。

3.8　贪心算法

题目：有 10 个人在一个水龙头前排队接水，假如每个人接水的时间为 T_i，请编程找出这 10 个人排队的一种顺序，使 10 个人的平均等待时间最小。输入 10 个数字，这 10 个数字分别表示第 1 个人到第 10 个人每人的接水时间 $T_1, T_2, \cdots, T_{10}$，每个数据之间有一个空格。输出文件有两行，第一行为一种排队顺序，即 1～10 的一种排列。第二行为这种排列方案下的平均等待时间(输出结果精确到小数点后两位)。

输入样例

```
56 12 1 99 1000 234 33 55 99 812
```

输出样例

```
3 2 7 8 1 4 9 6 10 5
291.90
```

分析：这是一道典型的贪心算法问题，实际上贪心算法并不是一种具体的算法，只是一种数学思想，即在对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，算法得到的是在某种意义上的局部最优解。根据经验和对样例数据的分析可知，如果想让整体等待时间最短，就要让等待时间少的人先接水，因此这道题的关键就变成了对输入时间进行从小到大排序，然后依次相加，但是要注意在排序或者输出的同时原来数字存储的序号（索引）不能乱。最后程序的代码如下：

```
//第 3 章/3.8
#将输入的 10 个数转化为整数并放在列表中存储
a = list(map(int,input("请输入 10 个数,之间以空格分隔:").split()))
#用 min 方法查找 a 中的最小值,找到后将最小值索引加入新列表 b,同时数值求和
b = []
#查找 10 次
sum = 0
for i in range(10):
    mini = min(a)
    #每个人接水的同时,后面的人要等待,所以接水时间 * 等待人数
    sum = sum + mini * (9 - i)
    b.append(a.index(mini) + 1)
    a[a.index(mini)] = max(a) + 1
for j in b:
    print(j,end = ' ')
print()
print("%.2f" %(sum/10))
```

程序执行的效果如图 3-10 所示。

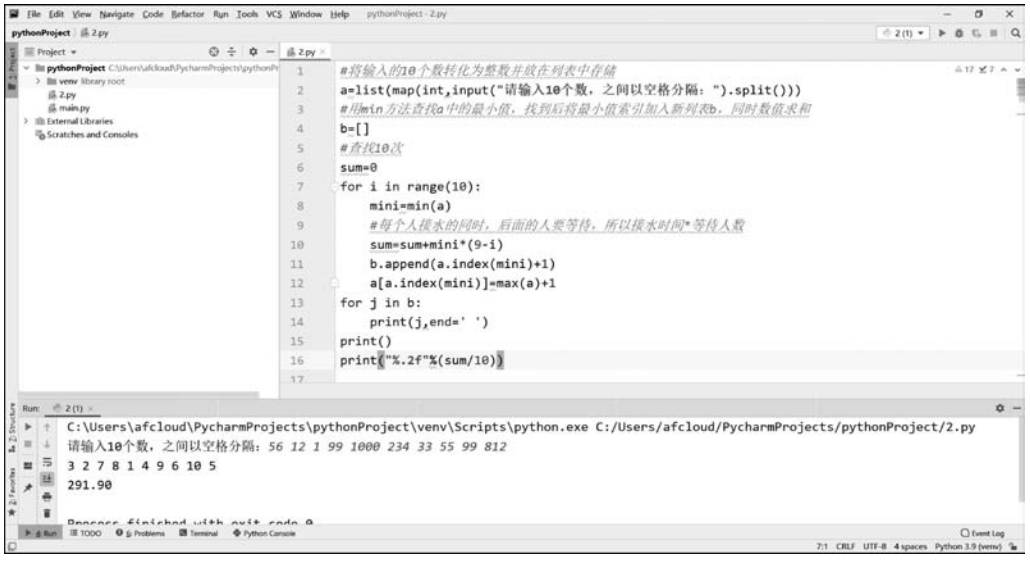

图 3-10　贪心算法执行结果