|  |  |  |  |
| --- | --- | --- | --- |
| 实验地点 | 专业班级 | 学号 | 姓名 |
| 宿舍 | 软件工程三班 | 2220202929 | 蔡博宇 |

3-1实现数学表达式

一、实验环境

编程语言： Java/Python

IDE环境： IDEA/Pycharm

二、课设题目

实现数学公式

三、分析问题、问题的模型

分析所求公式，分解可得，可重复利用每次计算结果，而不需要重复计算

四、解题思路、方案

4.1方案实现

4.1.1 选用的算法策略、使用该策略的理由

数学分析

4.1.2 算法的时间复杂度、空间复杂度分析

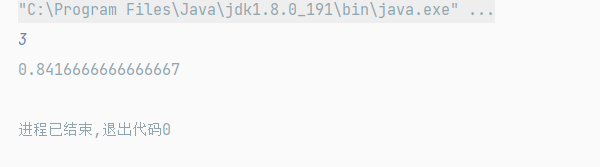
时间复杂度：O（n）

空间复杂度：O（0）

4.2 Java 实现程序源代码

|  |
| --- |
| public class FactorialLoop {  public static void main(String[] args) {  int n = 3;  System.out.println(factorialLoop(n));  }  static int factorialLoop(int n){  int S = 1;  int sign = 1;  int t = 1;  for (int i = 2; i <= n; i++) {  sign = -sign;  t = t \* ( (2 \* i-2) \* (2 \* i - 1));  S += sign / t;  }  return S;  }  } |

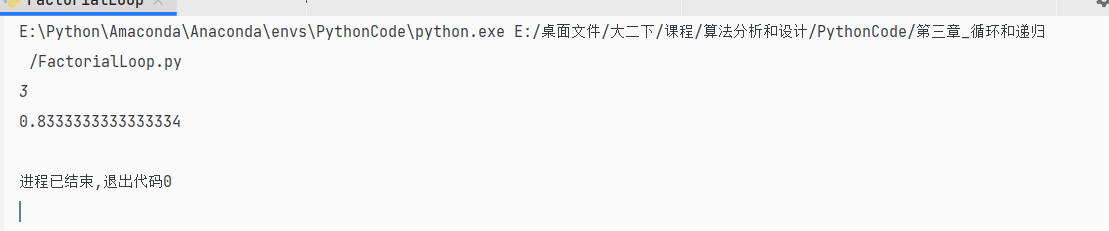
4.2.1程序运行结果：



4.3 python 实现

|  |
| --- |
| def factorialLoop(nn):  S = 1.0000  sign = 1  t = 1  for i in range(2, nn):  sign = -sign  t = t \* ((2 \* i - 2) \* (2 \* i - 1))  S += sign / t  return S  if \_\_name\_\_ == '\_\_main\_\_':  n = int(input())  print(factorialLoop(n)) |

4.3.1 程序运行结果



五、结论

畅所欲言，可从问题本身、程序设计语言、数据结构、算法策略优劣比较、实验结果、问题、解决方案的意义等多方面进行分析讨论。

3-2找出1000以内的完数

一、实验环境

编程语言： Java/Python

IDE环境： IDEA/Pycharm

二、课设题目

完数定义：所有因子累加得自身，包含1，且不重复

三、分析问题、问题的模型

1）质因数不为完数

2）依次遍历1000以内的数，依据条件找数

四、解题思路、方案

4.1方案实现

4.1.1 选用的算法策略、使用该策略的理由

循环

4.1.2 算法的时间复杂度、空间复杂度分析

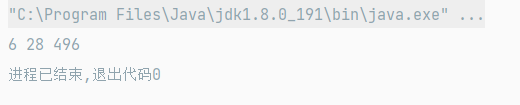
时间复杂度：O（n）

空间复杂度：O（0）

4.2 Java 实现程序源代码

|  |
| --- |
| public class PerfectNumber {  public static void main(String[] args) {  perfectNumber();  }  static void perfectNumber(){  int s ; // 统计和  int k ; // 因子数  for (int i = 1; i <= 1000; i++ ) {  s = i;  k = 0;  for (int j = 1; j < i; j++ ) {  if(i % j == 0) {  s-=j;  }  }  if(s == 0){  System.out.print(i + " ");  }  }  }  } |

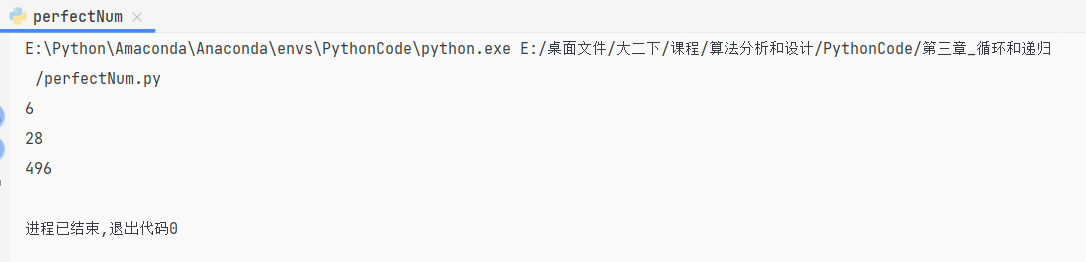
4.2.1程序运行结果：



4.3 python 实现

|  |
| --- |
| def perfectNumber():  for i in range(1000):  s = i  for j in range(1,i):  if i % j == 0:  s -= j  if s == 0 and i != 0:  print(i)  if \_\_name\_\_ == '\_\_main\_\_':  perfectNumber() |

4.3.1 程序运行结果



五、结论

畅所欲言，可从问题本身、程序设计语言、数据结构、算法策略优劣比较、实验结果、问题、解决方案的意义等多方面进行分析讨论。

3-3整数划分问题

一、实验环境

编程语言： Java/Python

IDE环境： IDEA/Pycharm

二、课设题目

将一个整数划分为比它小的数的累加

三、分析问题、问题的模型

划分为三种情况，设函数（n,m）n为输入数，m为不大于m的累加数

1）n=1 and m =1 只有一种全为1 例如 6 =1+1+1+1+1+1

2）n = m 时 一种情况 自身累加 例如 6 = 6

3）n > m 时 可分为包含m和不包含m的情况

四、解题思路、方案

4.1方案实现

4.1.1 选用的算法策略、使用该策略的理由

递归+分治

4.1.2 算法的时间复杂度、空间复杂度分析

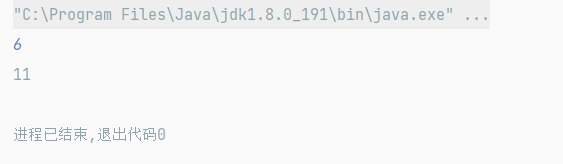
时间复杂度：O（n）

空间复杂度：O（0）

4.2 Java 实现程序源代码

|  |
| --- |
| public class DivInteger {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  System.out.println(Divider(n,n));  }  /\*\*  \* 整数划分  \* @param n 输入数  \* @param m 不超过m的被加数  \* @return 组合方式  \*/  static int Divider(int n,int m){  if(m==1 || n==1){  return 1;  }  else if (n < m) {  return Divider(n,n);  }else if (n == m) {  return Divider(n,n-1) + 1;  // 注意 n和m相等时只有一种划分方式  }else {  // 划分方式为 包含m的划分方式 + 不包含m的划分方式  return Divider(n-m,m)+Divider(n,m-1);  }  }  } |

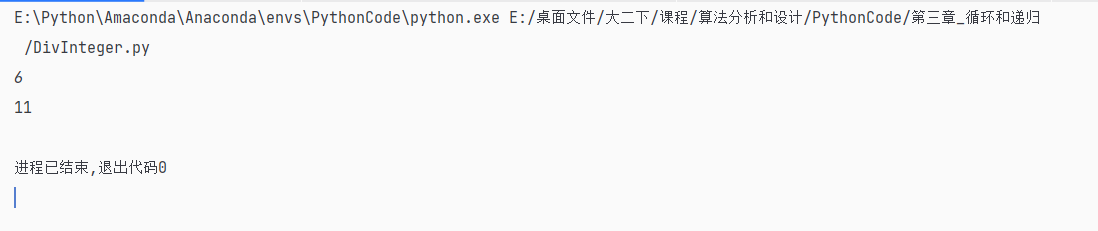
4.2.1程序运行结果：



4.3 python 实现

|  |
| --- |
| def divider(n, m):  if n == 1 or m == 1:  return 1  elif n < m:  return divider(n, n)  elif n == m:  return divider(n, n - 1) + 1  else:  return divider(n - m, m) + divider(n, m - 1)  if \_\_name\_\_ == '\_\_main\_\_':  nn = input()  nn = int(nn)  print(divider(nn, nn)) |

4.3.1 程序运行结果



五、结论

畅所欲言，可从问题本身、程序设计语言、数据结构、算法策略优劣比较、实验结果、问题、解决方案的意义等多方面进行分析讨论。

3-4 n数r组合问题

一、实验环境

编程语言： Java/Python

IDE环境： IDEA/Pycharm

二、课设题目

有从1到n个数，求只包含r个数的不重复组合

三、分析问题、问题的模型

可类比排列组合问题

1. 选定1个数，此时还需要选择r-1个数，有n-1个选择
2. 持续选择，直到满足条件

四、解题思路、方案

4.1方案实现

4.1.1 选用的算法策略、使用该策略的理由

递归

4.1.2 算法的时间复杂度、空间复杂度分析

时间复杂度：O（n !）

空间复杂度：O（n）

4.2 Java 实现程序源代码

|  |
| --- |
| public class Constitute {  static int total = 0;  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  int r = sc.nextInt();  constitute(n,r);  }  static void constitute(int n,int r){  if (n < r){  return;  }  int[] a = new int[n];  a[0] = r;  comb(n,r,a);  System.out.println(total);  }  static void comb(int n,int r,int[] a){  for (int i = n; i >= r; i--) {  a[r] = i;  if(r > 1){  comb(i-1,r-1,a);  }else {  for (int j = a[0]; j > 0 ; j--) {  System.out.print(a[j] + " ");  }  total = total + 1;  System.out.println();  }  }  }  } |

4.2.1程序运行结果：



4.3 python 实现

|  |
| --- |
| def constitute():  global a  a = []  global total  total = 0  for r in range(0, 100):  a.append(0)  n = int(input())  r = int(input())  a[0] = r  comb(n, r)  if n < r:  return  print(total)  def comb(n, r):  global total  for i in range(n, r-1, -1):  a[r] = i  if r > 1:  comb(i - 1, r - 1)  else:  for j in range(a[0], 0, -1):  print(a[j], end=" ")  total += 1  print(end="\n")  if \_\_name\_\_ == '\_\_main\_\_':  constitute() |

4.3.1 程序运行结果



五、结论

畅所欲言，可从问题本身、程序设计语言、数据结构、算法策略优劣比较、实验结果、问题、解决方案的意义等多方面进行分析讨论。