**陕 西 国 际 商 贸 学 院**

**实 验 报 告**

**（ 理 工 类 ）**

|  |  |
| --- | --- |
| **二级学院：** | **信息工程学院** |
| **专 业：** | **信息管理与信息系统** |
| **课程名称：** | **Python程序设计** |
| **班 级：** | **B2201** |
| **学 号：** | **22104040116** |
| **姓 名：** | **李霄钰** |
| **实验日期：** | **2024 年 5 月 11 日** |

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称：** | **项目三 函数和模块及面向对象编程** | | |
| **实验目的** | 掌握函数和模块的定义和使用；掌握类和对象的定义和使用。 | **实验室名称** | 4-311 |
| **主要仪器名称** | 计算机 | | |
| **实验类型** | 设计性 | | |
| **实验要求：**   1. 掌握函数和模块的定义和使用； 2. 了解模块在程序中的作用； 3. 掌握类的定义和使用； 4. 掌握对象的定义和使用。 | | | |
| **实验内容：**   1. 函数的调用、函数参数、函数的嵌套、lambda函数、递归函数； 2. 定义和使用类、对象的属性和方法； 3. 类的继承、运算符重载、模块中的类。 | | | |
| **实验步骤和实验记录（截图）：**   1. 打印杨辉三角。（当n=10时，打印结果如下图所示）     源代码：  import math  def yanghui(n):  triangle=[[1]\*i for i in range(1,n+1)]  for i in range(2,n):  for j in range(1,i):  triangle[i][j]=triangle[i-1][j-1]+triangle[i-1][j]  for i in range(1,n+1):  print(' '\*(n-i),end="")  for j in range(1,i+1):  print(triangle[i-1][j-1],end=" ")  print() if \_\_name\_\_ == '\_\_main\_\_':  n=int(input("请输入n的值："))  yanghui(n)  运行结果：    2、春节期间小明在微信群里抢到很多红包,有时是手气王,有时却只抢到1分。请利用函数编写一个程序,实现微信红包的随机分配法。用户输入红包金额和份数,系统按照顺序输出每个红包的金额。  【输人输出样例1】(其中红色字体表示输入数据)  请输入人数:2  请输入金额:10  第1个红包金额为:5.11  第2个红包金额为:4.89  【输人输出样例2】  请输入人数:3  请输入金额:9  第1个红包金额为:6.3  第2个红包金额为:0.21  第3个红包金额为:2.49  【输入输出样例3】  请输入人数:4  请输入金额:10  第1个红包金额为:4.07  第2个红包金额为:2.14  第3个红包金额为:3.1  第4个红包金额为:0.69  源代码：  import random  def hongbao(money,shuliang):  if shuliang <= 0 or money <= 0:  return "输入不正确！"  min\_money=0.01  hongbao01=[]  for i in range(shuliang-1):  max\_money=money-sum(hongbao01)-(shuliang-1)\*min\_money  nowmoney=random.uniform(min\_money,max\_money)  hongbao01.append(round(nowmoney,2))  last\_hongb=round(money-sum(hongbao01),2)  hongbao01.append(last\_hongb)  for i,nowmoney in enumerate(hongbao01,start=1):  print(f"第{i}个红包的金额是{nowmoney}") rs=int(input("请输入人数：")) je=float(input("请输入金额：")) hongbao(je,rs)  运行结果：    3、使用递归实现快速排序。  源代码：  def quick\_sort(list):  if len(list)<=1:  return list  pivot=list[len(list)//2]  left=[x for x in list if x<pivot]  middle=[x for x in list if x==pivot]  right=[x for x in list if x>pivot]  return quick\_sort(left)+middle+quick\_sort(right)  if \_\_name\_\_ == '\_\_main\_\_':  list01=[1,6,3,4,1,6,5,8,5]  print(quick\_sort(list01))  运行结果：    4、一般形式为的方程为一元二次方程，请设计一个一元二次方程类Equation，这个类中有：   * 代表方程的3个系数：a、b、c。 * 构造方法\_\_init\_\_。 * 一个名为computeDelta的实例方法，用来返回判别式的值。 * 一个名为getRoots的方法，用来返回方程的根。   【提示】若令，，则  当时，方程有两个相同的实根：r1=r2=p。  当时，方程有两个不相同的实根：r1=p+q，r2=p-q。  当时，方程有一对共轭复根：r1=p+qi，r2=p-qi。  【输入输出样例1】（红色字体表示输入数据）  请输入方程的系数a，b，c（用逗号隔开）：0，1，2  以上系数无法构成一元二次方程！  【输入输出样例2】（红色字体表示输入数据）  请输入方程的系数a，b，c（用逗号隔开）：1，2，1  r1=r2=－1.00  源代码：  import math class Equation:  def \_\_init\_\_(self,a,b,c):  self.a=a  self.b=b  self.c=c  def compuetrDelta(self):  return pow(self.b,2)-(4\*self.a\*self.c)  def getRoots(self):  if self.a==0:  return "此方程不是二元一次方程组！"  delta=self.compuetrDelta()  p=-self.b/(2\*self.a)  q=math.sqrt(abs(delta))/(2\*self.a)  if delta==0:  return f"r1=r2={p:.2f}"  if delta>0:  return f"r1={p+q:.2f},r2={p-q:.2f}"  if delta<0:  return f"r1={p:.2f}+{abs(q):.2f}i,r2={p:.2f}-{abs(1):.2f}i"  if \_\_name\_\_ == '\_\_main\_\_':  num\_list=input("请输入方程的系数").split(',')  e1=Equation(int(num\_list[0]),int(num\_list[1]),int(num\_list[2]))  print(e1.getRoots())  运行结果：    5、自定义一个分数类Rational。用重载运算符完成分数的加、减、乘、除四则运算。该类还能处理分母为0或分数不是最简形式等情况。  【输入输出样例1】（红色字体表示输入数据）  请输入第一个分数：1/2  请输入第二个分数：3/4  1/2+3/4=5/4  1/2-3/4=-1/4  1/2\*3/4=3/8  1/2/3/4=2/3  【输入输出样例2】  请输入第一个分数：3/4  请输入第二个分数：6/8  3/4+3/4=3/2  3/4-3/4=0  3/4\*3/4=9/16  3/4/3/4=1  源代码：  from math import gcd   class Rational:  def \_\_init\_\_(self, number, numbermother=1):  if numbermother == 0:  raise ValueError("Denominator cannot be zero.")  self.number = number  self.numbermother = numbermother  self.simplify()   def simplify(self):  s = gcd(abs(self.number), abs(self.numbermother))  self.number //= s  self.numbermother //= s   def \_\_str\_\_(self):  if self.numbermother == 1:  return str(self.number)  else:  return f"{self.number}/{self.numbermother}"   def \_\_add\_\_(self, other):  if isinstance(other, Rational):  new\_number = self.number \* other.numbermother + other.number \* self.numbermother  new\_numbermother = (self.numbermother \* other.numbermother)  return Rational(new\_number, new\_numbermother)  else:  raise TypeError("Unsupported operand types for +: 'Rational' and '{}'".format(type(other).\_\_name\_\_))   def \_\_sub\_\_(self, other):  if isinstance(other, Rational):  new\_number = self.number \* other.numbermother - other.number \* self.numbermother  new\_numbermother= self.numbermother \* other.numbermother  return Rational(new\_number, new\_numbermother)  else:  raise TypeError("Unsupported operand types for -: 'Rational' and '{}'".format(type(other).\_\_name\_\_))   def \_\_mul\_\_(self, other):  if isinstance(other, Rational):  new\_number = self.number \* other.number  new\_numbermother = self.numbermother \* other.numbermother  return Rational(new\_number, new\_numbermother)  elif isinstance(other, int) or isinstance(other, float):  return Rational(self.number \* other, self.numbermother)  else:  raise TypeError("Unsupported operand types for \*: 'Rational' and '{}'".format(type(other).\_\_name\_\_))   def \_\_truediv\_\_(self, other):  if isinstance(other, Rational):  new\_number = self.number \* other.numbermother  new\_numbermother = self.numbermother \* other.number  return Rational(new\_number, new\_numbermother)  elif isinstance(other, int) or isinstance(other, float):  return Rational(self.number, self.numbermother \* other)  else:  raise TypeError("Unsupported operand types for /: 'Rational' and '{}'".format(type(other).\_\_name\_\_))   if \_\_name\_\_ == "\_\_main\_\_":  fraction1 = Rational(\*map(int, input("请输入第一个分数（分子/分母）: ").split('/')))  fraction2 = Rational(\*map(int, input("请输入第二个分数（分子/分母）: ").split('/')))  print(f"{fraction1}+{fraction2}={fraction1 + fraction2}")  print(f"{fraction1}-{fraction2}={fraction1 - fraction2}")  print(f"{fraction1}\*{fraction2}={fraction1 \* fraction2}")  print(f"{fraction1}/{fraction2}={fraction1 / fraction2}")  运行结果： | | | |
| **实验总结：**  **熟练掌握了类和对象的使用方法。** | | | |
| **成绩和评语：**  **指导教师：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **审阅日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | | |