**实验报告**

|  |  |  |
| --- | --- | --- |
| 课程名称：python程序设计实验课 | 班级：软件1912 | 实验日期：2021-5-27 |
| 姓名：李越 | 学号：20192071 | 指导老师：杨帆 |
| 实验名称：面向对象 | 实验序号：12 | 实验成绩： |
| 1. 实验目的： 2. 面向对象思想完成贴瓷砖 3. 设计一个表示动物的类，设计一个鱼类(编程题4) 4. 设计一个名为Triangle的类来扩展GeometricObject类，该类包含：   三个名字为side1，side2，side3的浮点数据域表示三角形的三个边  一个构造方法构造默认一个三角形，执行三边分别为1.0  三个数据域的访问器方法  一个获得面积的方法  一个获得周长的方法  一个对三角形的字符串描述   1. 设计一个名为Location的类来定位一个二维列表中最大值及位置。这个类包含有公有数据域row，coloumn和maxValue来存储最大值和它在二维列表中的下标值，其中row和coloumn是int类型的而maxValue为Float型的。编写一个方法返回二维列表最大值的位置 2. 创建一个Account类来模拟一台ATM机，创建一个有10个账户的列表，其id为0-9，并初始化收支为100美元。系统提示用户输入一个id，如果输入的id不正确，要求输入正确的id，如果正确显示主菜单，1：支出，3：收钱，3：存钱；4：退出。一旦退出，系统需要重新输入id。   二、实验环境：  Python3.9.2  三、实验步骤：  2.  import random as r  legal\_x = [0, 10]  legal\_y = [0, 10]  class Fish:      def \_\_init\_\_(self):          self.x = r.randint(legal\_x[0], legal\_x[1])          self.y = r.randint(legal\_y[0], legal\_y[1])        def move(self):          # 随机计算方向并移动到新的位置（x, y）          new\_x = self.x + r.choice([1, -1])          new\_y = self.y + r.choice([1, -1])          # 检查移动后是否超出场景x轴边界          if new\_x < legal\_x[0]:              self.x = legal\_x[0] - (new\_x - legal\_x[0])          elif new\_x > legal\_x[1]:              self.x = legal\_x[1] - (new\_x - legal\_x[1])          else:              self.x = new\_x          # 检查移动后是否超出场景y轴边界          if new\_y < legal\_y[0]:              self.y = legal\_y[0] - (new\_y - legal\_y[0])          elif new\_y > legal\_y[1]:              self.y = legal\_y[1] - (new\_y - legal\_y[1])          else:              self.y = new\_y          # 返回移动后的新位置          return (self.x, self.y)  class animals:      def \_\_init\_\_(self,name,habbit):          self.name = name          self.habbit = habbit      def getName(self):          return self.name        def setName(self,name):          self.name = name        def getHabbit(self):          return self.habbit        def setHabbit(self,habbit):          self.habbit = habbit        def show(self):          reset = 'name:{0},habbit:{1}'.format(self.name,self.habbit)          print(reset)    fish = Fish()  print(fish.move())  print(fish.move())  animal = animals('邹一泓','吃粑粑')  animal.show()  3.  class Triangle:      \_\_side1 = 0      \_\_side2 = 0      \_\_side3 = 0        def \_\_init\_\_(self):          self.\_\_side1 = 1          self.\_\_side2 = 1          self.\_\_side3 = 1        def getSide(self):          return self.\_\_side1,self.\_\_side2,self.\_\_side3        def setSide(self,side1,side2,side3):          self.\_\_side1 = side1          self.\_\_side2 = side2          self.\_\_side3 = side3        def getArea(self):          area = self.side1\*self.side2/0.5          return area        def getLen(self):          return self.\_\_side1+self.\_\_side2+self.\_\_side3        def show(self):          reset = '三边长分别为：{0},{1},{2}'.format(self.\_\_side1,self.\_\_side2,self.\_\_side3)          print(reset)  triangle = Triangle()  triangle.show()  4.  class location():      \_\_row = 0      \_\_column = 0      \_\_maxValue = 0.0        def \_\_init\_\_(self,list):          self.list = list        def getMax(self):          for i in range(len(self.list[0])):              for j in range(len(self.list)):                  if(self.list[j][i]>self.\_\_maxValue):                      self.\_\_maxValue = self.list[j][i]                      self.\_\_row = j                      self.\_\_column = i          print(self.\_\_row)          print(self.\_\_column)      list = [[1,2,3,4],[5,6,7,8]]  locate = location(list)  locate.getMax()  5.  class Account:      \_\_id = 0      \_\_balance = 0.0      \_\_annualInterestRate = 0.0      def \_\_init\_\_(self,id):          self.\_\_id = id          self.\_\_balance = 100      def getID(self):          return self.\_\_id        def setID(self,id):          self.\_\_id = id        def getBalance(self):          return self.\_\_balance        def setBalance(self,balance):          self.\_\_balance = balance        def getAnnual(self):          return self.\_\_annualInterestRate        def setAnnual(self,annual):          self.\_\_annualInterestRate = annual        def getMonthlyInterestRae(self):          return self.\_\_annualInterestRate        def getMonthlyInterest(self):          Intereset = self.\_\_balance \* self.\_\_annualInterestRate/30\*30          return Intereset        def wirhDraw(self,num):          self.\_\_balance = self.\_\_balance - num  def view():      flag = 0      while(flag != 4):          print('\*\*\*\*\*\*\*\*\*\*\*\*\*')          print('1.支出')          print('2.收钱')          print('3.存钱')          print('4.退出')          print('\*\*\*\*\*\*\*\*\*\*\*\*\*')          flag = eval(input('请输入您的选择：'))  def main():      account=[]      for i in range(1,11):          account.append(Account(i))      flag = True      while(flag == True):          id = eval(input("请输入编号:"))          for ac in account:              if id == ac.getID():                  flag = False      view()  if \_\_name\_\_ == "\_\_main\_\_":      main()  四、实验结果：  2.    3.    4.    5.    五、实验心得：  无 | | |