**中央财经大学**

**实 验 报 告**

**实验项目名称**  第13章上机实验

**所属课程名称**  Python程序设计

**实 验 类 型**  综合、设计性

**实 验 日 期**  2019.11.23

**班 级**  计算机科学18

**学 号**  2018312352

**姓 名**  杨智江

**成 绩**

**实验室**

|  |
| --- |
| **实验概述：** |
| **【实验目的及要求】**  编写程序完成题目要求  **【实验原理】**  Python3.7  **【实验环境】（使用的软件）**  Python3.7 IDLE  Visual Studio Code |
| **实验内容：**  整个程序分为程序入口main.py，前端菜单模块menu.py，后端信用卡模块card.py。用到了time datetime json模块  在信用卡类中，卡片有四个成员变量  class CreditCard():      def \_\_init\_\_(self, balance, credit):          self.balance = balance  # 账户余额          self.credit = credit  # 信用额度          self.record = []  # 交易记录          self.total\_deal = 0  # 交易数量  定义的操作类型有显示账户信息，存款，从账户余额中取款，从信用额度中取现，支付，还款，写入交易记录和写入菜单。  除此以外，经常使用的步骤设置为函数，有提示用户选择，安全地转换浮点数，存取款金额校验三个，都声明为静态方法。      @staticmethod      def user\_select():#用户选择          chosen = False          while not chosen:              try:                  choice = input('>>').strip()[0].lower()              except (EOFError, KeyboardInterrupt):                  print('Error, try again')              if choice not in 'yn':                  print('invalid option, try again')              else:                  chosen = True          if choice == 'y':              return 1          elif choice == 'n':              return 0  @staticmethod      def valid\_money\_atm():          while True:  # 实现自动取款机的操作              money = int(input('请输入金额 (100的整数倍)： '))              if money % 100 != 0 or money < 0:                  print('金额有误，请重新输入')              else:                  return money  @staticmethod      def safe\_float\_num(num):          try:              return float(num)          except(TypeError, ValueError) as e:              print(e)  # 当转换出错时输出错误              return None  显示账户信息  def show(self):  # 显示账户的余额和额度          print('当前账户余额为：%.2f' % self.balance)          print('当前可用信用额度为：%.2f' % self.credit)  存款  def deposit(self):          money = self.valid\_money\_atm()          self.balance += money          self.transaction('deposit', money)          print('存款成功！')  从信用额度取现时，将从账户余额中直接扣除手续费  def draw\_credit(self):          flag = 0          while True:              money = CreditCard.valid\_money\_atm()              if money > self.credit:                  print('您的额度不足，是否重新输入 (Y/N)：')                  choice1 = CreditCard.user\_select()                  if choice1 == 1:                      pass                  else:                      flag = 1                      break              else:                  break          if flag == 0:              print('取现金额：%d 元,手续费：%.2f 元' % (money, 0.05\*money))              print('是否确认取现? (Y/N)')              choice2 = CreditCard.user\_select()              if choice2 == 1:                  if 0.05\*money > self.balance:                      print('手续费扣除失败，余额不足')                  else:                      self.credit -= money                      self.balance -= money\*0.05                      self.transaction('draw\_credit', money)                      self.transaction('fee', money\*0.05)                      print('取现成功！')              else:                  pass  从账户余额中取款  def draw\_balance(self):          flag = 0          while True:              money = CreditCard.valid\_money\_atm()              if money > self.balance:                  print('您的余额不足，是否重新输入(Y/N)：')                  choice1 = CreditCard.user\_select()                  if choice1 == 1:                      pass                  else:                      flag = 1                      break              else:                  break          if flag == 0:              self.balance -= money              self.transaction('draw\_balance', money)              print('提现成功！')  支付时，要将输入安全地转为浮点数  def pay(self):          while True:              money = CreditCard.safe\_float\_num(input('请输入金额：'))              if money < 0:                  print('金额有误，请重新输入')              else:                  break          if money > self.credit:              print('信用额度不足')          else:              self.credit -= money              self.transaction('pay', money)              print('支付成功！')  还款时，显示用户待还的金额，还款额不符合要求也会提示    def repayment(self):          money\_need\_pay = round(15000-self.credit,2)          print('您的待还余额为：%.2f' % money\_need\_pay)          while True:              money = CreditCard.safe\_float\_num(input('请输入还款金额：'))              if money < 0:                  print('金额有误，请重新输入')              elif money > money\_need\_pay:                  print('您太有钱了，不用还这么多')              else:                  self.credit += money                  self.transaction('repay', money)                  print('还款成功！')                  break    每一笔交易都会记录在账户上，包括序号、时间(使用time模块)、交易类型、金额和利息  def transaction(self, type, money):          item = {}          item['code'] = self.total\_deal+1          item['time'] = time.asctime(time.localtime(time.time()))          item['type'] = type          item['amount'] = money          item['interest'] = 0          self.record.append(item)          self.total\_deal += 1  输出账单使用了json模块将字典转换为str，使用了try-else处理异常      def bill(self):          try:              fobj = open('bill.log', 'a+')          except IOError as e:              print(e)          else:              for rec in self.record:                  str\_rec = json.dumps(rec, ensure\_ascii=False)                  fobj.write(str\_rec+'\n')  menu.py主要是生成菜单，实现对应的操作  def menu(mycard):  # 修改了第十章作业的代码      prompt = """                  Welcome:      (S)how account   (B)alance withdraw      (P)ay            (C)redit withdraw      (D)eposit        (R)epayment                  (Q)uit                Have a nice day!      Please select: """      done = False      while not done:          chosen = False          while not chosen:              try:                  choice = input(prompt).strip()[0].lower()              except (EOFError, KeyboardInterrupt):                  choice = 'q'              print('\nYou picked: [%s]' % choice)              print('\n')              if choice not in 'spdbcrq':                  print('invalid option, try again')              else:                  chosen = True          if choice == 's':              mycard.show()          if choice == 'p':              mycard.pay()          if choice == 'd':              mycard.deposit()          if choice == 'b':              mycard.draw\_balance()          if choice == 'c':              mycard.draw\_credit()          if choice == 'r':              mycard.repayment()          if choice == 'q':              done = True  main.py首先加载模块，然后实例化对象，为了方便实现和调试，后面进入循环，每退出一次菜单就相当于过了一天，然后利用系统函数判断今天是否为还款日，如果是就输出账单，提示还款。  from card import \*  from menu import \*  import datetime  mycard=CreditCard(1000,15000)  while True:      date=datetime.datetime.now()      if date.day==30:          mycard.bill()          print('Notice: 还款日到了！账单已出，请及时还款！')          menu(mycard)      else:          menu(mycard)  **程序运行结果**      为了方便调试，将电脑时间设置为30号    输出的账单： |
| **【小结】**  **通过这次实验我使用类完成一个程序，加深了对类的操作的理解，这和学过的c++中的操作有很多不同，调试时也出现了问题，调试时出现了未找到变量名的情况，是因为忘记了类的静态方法要用类名.函数名调用。**  **因为本题是信用卡的操作，会出现许多浮点数，不能用int类型了，在输出时要控制小数位数，同时因为浮点数是不精确的，在比较大小时会出现bug，就要四舍五入进行处理** |
| **指导教师评语及成绩：** |
| **评语：**  **成绩： 指导教师签名：**  **批阅日期：** |

实验报告说明

**1．实验项目名称：**要用最简练的语言反映实验的内容。要求与实验指导书中相一致。

**2．实验类型：**一般需说明是验证型实验还是设计型实验，是创新型实验还是综合型实验。

**3．实验目的与要求**：目的要明确，要抓住重点，符合实验指导书中的要求。

**4．实验原理：**简要说明本实验项目所涉及的理论知识。

**5．实验环境**：实验用的软硬件环境（配置）。

**6．实验方案设计（思路、步骤和方法等）**：这是实验报告极其重要的内容。概括整个实验过程。

对于**操作型实验**，要写明依据何种原理、操作方法进行实验，要写明需要经过哪几个步骤来实现其操作。

对于**设计型和综合型实验**，在上述内容基础上还应该画出流程图、设计思路和设计方法，再配以相应的文字说明。

对于**创新型实验**，还应注明其创新点、特色。

**7．实验过程（实验中涉及的记录、数据、分析）：**写明具体上述实验方案的具体实施，包括实验过程中的记录、数据和相应的分析。

**8．结论（结果）：**即根据实验过程中所见到的现象和测得的数据，做出结论。

**9．小结：**对本次实验的心得体会、思考和建议。

**10．指导教师评语及成绩：**指导教师依据学生的实际报告内容，用简练语言给出本次实验报告的评价和价值。

**注意：**

* 实验报告将记入实验成绩；
* 每次实验开始时，交上一次的实验报告，否则将扣除此次实验成绩。