山西工程技术学院

**结课报告**

（2024-2025学年第二学期）

课程名称： python程序设计

专业班级： 22计算机科学与技术一班

学 号： 2210708130

学生姓名： 郝泓毅

任课教师： 谢瑞洁

2025年6月

结课报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 综合应用 | | | 指导教师 | 谢瑞洁 |
| 类型 | 综合型 | 学时 | 32 | 时间 | 2025-6 |
| 1. **目的与要求**   (1)熟练使用python语言编写简单的应用程序，利用Python语言解决实际问题。  (2)掌握python中经典算法的应用，提高算法设计能力，培养编程的一般性思维。 | | | | | |
| **二、环境**  操作系统：win-10 | | | | | |
| **三、内容和步骤**  1.用迭代法求最大公约数。  代码如下：  def gcd(x, y):      while y:          x, y = y, x % y      return x  num1 = input("Please enter the first number: ")  num2 = input("Please enter the second number: ")  num1, num2 = int(num1), int(num2)  result = gcd(num1, num2)  print(f"The greatest common divisor of {num1} and {num2} is: {result}"  运行结果如下：    2.假设公司有三类员工，将员工定义为基类，三类员工分别继承基类中的属性，并定义自己的特殊属性，利用派生类实现不同的薪资计算方法。  代码如下：  class Staff:      def \_\_init\_\_(self, name, wage):          self.name = name          self.wage = wage      def get\_name(self):          return self.name      def get\_wage(self):          return self.wage      def set\_wage(self, new\_wage):          self.wage = new\_wage  class Agroup(Staff):      def \_\_init\_\_(self, name, wage, special\_skill):          super().\_\_init\_\_(name, wage)          self.special\_skill = special\_skill      def get\_name(self):          return super().get\_name()      def get\_wage(self):          return 1.5 \* super().get\_wage()      def set\_wage(self, new\_wage):          super().set\_wage(new\_wage)      def show\_special\_skill(self):          return self.special\_skill  class Bgroup(Staff):      def \_\_init\_\_(self, name, wage, project\_experience):          super().\_\_init\_\_(name, wage)          self.project\_experience = project\_experience      def get\_name(self):          return super().get\_name()      def get\_wage(self):          return 1.2 \* super().get\_wage()      def set\_wage(self, new\_wage):          super().set\_wage(new\_wage)      def show\_project\_experience(self):          return self.project\_experience  class Cgroup(Staff):      def \_\_init\_\_(self, name, wage, seniority):          super().\_\_init\_\_(name, wage)          self.seniority = seniority      def get\_name(self):          return super().get\_name()      def get\_wage(self):          return 1.0 \* super().get\_wage()      def set\_wage(self, new\_wage):          super().set\_wage(new\_wage)      def show\_seniority(self):          return self.seniority  if \_\_name\_\_ == "\_\_main\_\_":        staff\_member = Staff("John", 5000)      print(f"Name: {staff\_member.get\_name()}")      print(f"Wage: {staff\_member.get\_wage()}")      staff\_member.set\_wage(6000)      print(f"Modified Wage: {staff\_member.get\_wage()}")      a\_staff = Agroup("Alice", 5000, "Data Analysis")      print(f"\nA Group Staff - Name: {a\_staff.get\_name()}")      print(f"A Group Staff - Wage: {a\_staff.get\_wage()}")      print(f"A Group Staff - Special Skill: {a\_staff.show\_special\_skill()}")      b\_staff = Bgroup("Bob", 5000, "5 years of project experience")      print(f"\nB Group Staff - Name: {b\_staff.get\_name()}")      print(f"B Group Staff - Wage: {b\_staff.get\_wage()}")      print(f"B Group Staff - Project Experience: {b\_staff.show\_project\_experience()}")      c\_staff = Cgroup("Charlie", 5000, "3 years")      print(f"\nC Group Staff - Name: {c\_staff.get\_name()}")      print(f"C Group Staff - Wage: {c\_staff.get\_wage()}")      print(f"C Group Staff - Seniority: {c\_staff.show\_seniority()}")  运行结果如下： | | | | | |
| 3.设计一个登陆界面并实现购物，验证用户登陆信息，如果用户的账号和密码正确则可以执行并进入到购物界面，如果不正确则提示用户名或密码错误。（自己设计出一个购物系统即可）  代码如下：  import sys  from PyQt6.QtWidgets import (      QApplication, QWidget, QLabel, QLineEdit, QPushButton,      QVBoxLayout, QMessageBox, QListWidget, QHBoxLayout  )  USER\_DATABASE = {      'alice': '123456',      'bob': 'password'  }  PRODUCTS = [      ('苹果', 3.0), ('香蕉', 2.5), ('牛奶', 5.0), ('面包', 4.0), ('鸡蛋', 6.0),      ('橙子', 3.5), ('西瓜', 12.0), ('酸奶', 4.5), ('饼干', 3.0), ('巧克力', 6.5)  ]  ITEMS\_PER\_PAGE = 5  class LoginWindow(QWidget):      def \_\_init\_\_(self):          super().\_\_init\_\_()          self.setWindowTitle("登录")          self.setGeometry(100, 100, 300, 150)          self.username\_label = QLabel("用户名:")          self.username\_input = QLineEdit()          self.password\_label = QLabel("密码:")          self.password\_input = QLineEdit()          self.password\_input.setEchoMode(QLineEdit.EchoMode.Password)          self.login\_button = QPushButton("登录")          self.login\_button.clicked.connect(self.check\_login)          layout = QVBoxLayout()          layout.addWidget(self.username\_label)          layout.addWidget(self.username\_input)          layout.addWidget(self.password\_label)          layout.addWidget(self.password\_input)          layout.addWidget(self.login\_button)          self.setLayout(layout)      def check\_login(self):          username = self.username\_input.text()          password = self.password\_input.text()          if USER\_DATABASE.get(username) == password:              self.accept\_login()          else:              QMessageBox.warning(self, "登录失败", "用户名或密码错误！")      def accept\_login(self):          self.hide()          self.shop\_window = ShopWindow(username=self.username\_input.text())          self.shop\_window.show()  class ShopWindow(QWidget):      def \_\_init\_\_(self, username):          super().\_\_init\_\_()          self.setWindowTitle("购物系统")          self.setGeometry(100, 100, 500, 400)          self.username = username          self.cart = []          self.current\_page = 0          self.label = QLabel(f"欢迎你，{self.username}！请选择你想购买的商品：")          self.product\_list = QListWidget()          self.update\_product\_list()          self.cart\_list = QListWidget()          self.add\_button = QPushButton("添加到购物车")          self.add\_button.clicked.connect(self.add\_to\_cart)          self.checkout\_button = QPushButton("结算")          self.checkout\_button.clicked.connect(self.checkout)          self.view\_cart\_button = QPushButton("查看购物车")          self.view\_cart\_button.clicked.connect(self.show\_cart)          self.prev\_page\_button = QPushButton("上一页")          self.prev\_page\_button.clicked.connect(self.prev\_page)          self.next\_page\_button = QPushButton("下一页")          self.next\_page\_button.clicked.connect(self.next\_page)          layout = QVBoxLayout()          layout.addWidget(self.label)          layout.addWidget(self.product\_list)          nav\_layout = QHBoxLayout()          nav\_layout.addWidget(self.prev\_page\_button)          nav\_layout.addWidget(self.next\_page\_button)          layout.addLayout(nav\_layout)          layout.addWidget(self.add\_button)          layout.addWidget(self.view\_cart\_button)          layout.addWidget(self.checkout\_button)          layout.addWidget(QLabel("购物车内容:"))          layout.addWidget(self.cart\_list)          self.setLayout(layout)      def update\_product\_list(self):          self.product\_list.clear()          start = self.current\_page \* ITEMS\_PER\_PAGE          end = start + ITEMS\_PER\_PAGE          for name, price in PRODUCTS[start:end]:              self.product\_list.addItem(f"{name} - ￥{price:.2f}")      def add\_to\_cart(self):          selected\_item = self.product\_list.currentItem()          if selected\_item:              text = selected\_item.text()              name = text.split(" - ")[0]              price = float(text.split("￥")[1])              self.cart.append((name, price))              QMessageBox.information(self, "已添加", f"{name} 已加入购物车！")      def show\_cart(self):          self.cart\_list.clear()          if not self.cart:              QMessageBox.information(self, "购物车", "你的购物车是空的。")          else:              for name, price in self.cart:                  self.cart\_list.addItem(f"{name} - ￥{price:.2f}")      def checkout(self):          if not self.cart:              QMessageBox.information(self, "结算", "购物车为空，无法结算。")          else:              total = sum(price for \_, price in self.cart)              QMessageBox.information(self, "结算成功", f"总金额：￥{total:.2f}，感谢你的购买！")              self.cart.clear()              self.cart\_list.clear()      def next\_page(self):          if (self.current\_page + 1) \* ITEMS\_PER\_PAGE < len(PRODUCTS):              self.current\_page += 1              self.update\_product\_list()      def prev\_page(self):          if self.current\_page > 0:              self.current\_page -= 1              self.update\_product\_list()  if \_\_name\_\_ == "\_\_main\_\_":      app = QApplication(sys.argv)      login\_window = LoginWindow()      login\_window.show()      sys.exit(app.exec())  运行结果如下：   |  |  | | --- | --- | |  |  | |  |  |   4.使用DFS实现机器人寻路程序（自己设计出一个机器人寻找最短路径算法即可）。  代码如下：  def dfs(grid, start, target):      rows, cols = len(grid), len(grid[0])      visited = set()      path = []      min\_path = []      def backtrack(current):          nonlocal min\_path          if (current[0] < 0 or current[0] >= rows or current[1] < 0 or current[1] >= cols or                  grid[current[0]][current[1]] == 1 or tuple(current) in visited):              return          path.append(current)          visited.add(tuple(current))          if current == target:              if not min\_path or len(path) < len(min\_path):                  min\_path = path[:]          else:              directions = [(0, 1), (1, 0), (0, -1), (-1, 0)]              for dx, dy in directions:                  new\_x, new\_y = current[0] + dx, current[1] + dy                  backtrack([new\_x, new\_y])          path.pop()      backtrack(start)      return min\_path if min\_path else "没有找到从起点到目标点的路径"  grid = [      [0, 0, 0, 0],      [0, 1, 0, 0],      [0, 0, 0, 0],      [0, 0, 0, 0]  ]  start = [0, 0]  target = [3, 3]  result = dfs(grid, start, target)  print(result)  运行结果如下： | | | | | |
| 5.设计人机猜拳游戏，将游戏过程分解为玩家的动作、机器的动作以及人和机器的互动，分别用类实现。玩家赢则玩家得一分，机器赢则机器得一分。游戏结束后，统计总的猜拳次数，比较玩家和机器的得分，得分高的判为游戏胜利。  代码如下：  import random  class Player:      def \_\_init\_\_(self):          self.score = 0      def make\_move(self):          while True:              move = input("请输入你的选择（石头、剪刀、布）：")              if move in ['石头', '剪刀', '布']:                  return move              else:                  print("无效的输入，请重新输入。")  class Machine:      def \_\_init\_\_(self):          self.score = 0      def make\_move(self):          choices = ['石头', '剪刀', '布']          return random.choice(choices)  class Game:      def \_\_init\_\_(self):          self.player = Player()          self.machine = Machine()          self.rounds = 0      def play\_round(self):          player\_move = self.player.make\_move()          machine\_move = self.machine.make\_move()          print(f"你出了：{player\_move}")          print(f"机器出了：{machine\_move}")          self.rounds += 1          if player\_move == machine\_move:              print("平局！")          elif (player\_move == '石头' and machine\_move == '剪刀') or \                  (player\_move == '剪刀' and machine\_move == '布') or \                  (player\_move == '布' and machine\_move == '石头'):              print("你赢了！")              self.player.score += 1          else:              print("机器赢了！")              self.machine.score += 1      def play\_game(self):          while True:              self.play\_round()              play\_again = input("是否继续游戏？（是/否）：")              if play\_again.lower() != '是':                  break          print(f"游戏结束，总共进行了{self.rounds}轮。")          print(f"你的得分：{self.player.score}")          print(f"机器的得分：{self.machine.score}")          if self.player.score > self.machine.score:              print("你赢得了游戏胜利！")          elif self.player.score < self.machine.score:              print("机器赢得了游戏胜利！")          else:              print("游戏平局！")  if \_\_name\_\_ == "\_\_main\_\_":      game = Game()      game.play\_game()  运行结果如下： | | | | | |
| **四、小结和思考**  **通过本次Python课程的学习和实践，我掌握了Python语言的基本语法、控制结构、函数定义及模块使用，能使用Python编写中小型程序，解决一些实际问题。特别是在项目实战中，我收获颇多在学习过程中也遇到了一些挑战，比如PyQt的信号与槽机制、数据分页逻辑的设计、DFS中路径回溯的细节控制等，但通过查阅资料和不断调试，我逐步解决了这些问题，也提升了独立分析和解决问题的能力。**  **总的来说，这门课程不仅提高了我的编程技术水平，也增强了我对程序逻辑结构的把握能力。未来我将继续深入学习Python及其相关框架，为今后从事数据分析、人工智能或软件开发打下坚实的基础。** | | | | | |
| 成绩 |  | 批阅日期 |  | 批阅人 |  |