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

一、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：

```
PS C:\Users\Lanyi\Desktop\Project\SXIT-Python_Program> & C:/User
s/Lanyi/Desktop/Project/SXIT-Python_Program/Lectures/Final_proje
Please enter the first number: 29
Please enter the second number: 4
The greatest common divisor of 29 and 4 is: 1
PS C:\Users\Lanyi\Desktop\Project\SXIT-Python_Program>
```

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：

```
PS C:\Users\Lanyi\Desktop\Project\SXIT-Python_Program> & C:/Users/Lanyi/AppData/Local/Programs/Python/Python313/python.exe c:/User
s/Lanyi/Desktop/Project/SXIT-Python_Program/Lectures/Final_project/company.py
Name: John
Wage: 5000
Modified Wage: 6000

A Group Staff - Name: Alice
A Group Staff - Wage: 7500.0
A Group Staff - Special Skill: Data Analysis

B Group Staff - Name: Bob
B Group Staff - Wage: 6000.0
B Group Staff - Project Experience: 5 years of project experience

C Group Staff - Name: Charlie
C Group Staff - Wage: 5000.0
C Group Staff - Seniority: 3 years
PS C:\Users\Lanyi\Desktop\Project\SXIT-Python_Program>
```

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
)

# 模拟用户数据库
USER_DATABASE = {
    'alice': '123456',
    'bob': 'password'
}

# 商品及价格列表
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  # 每页展示数量

# 登录窗口
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:")
```

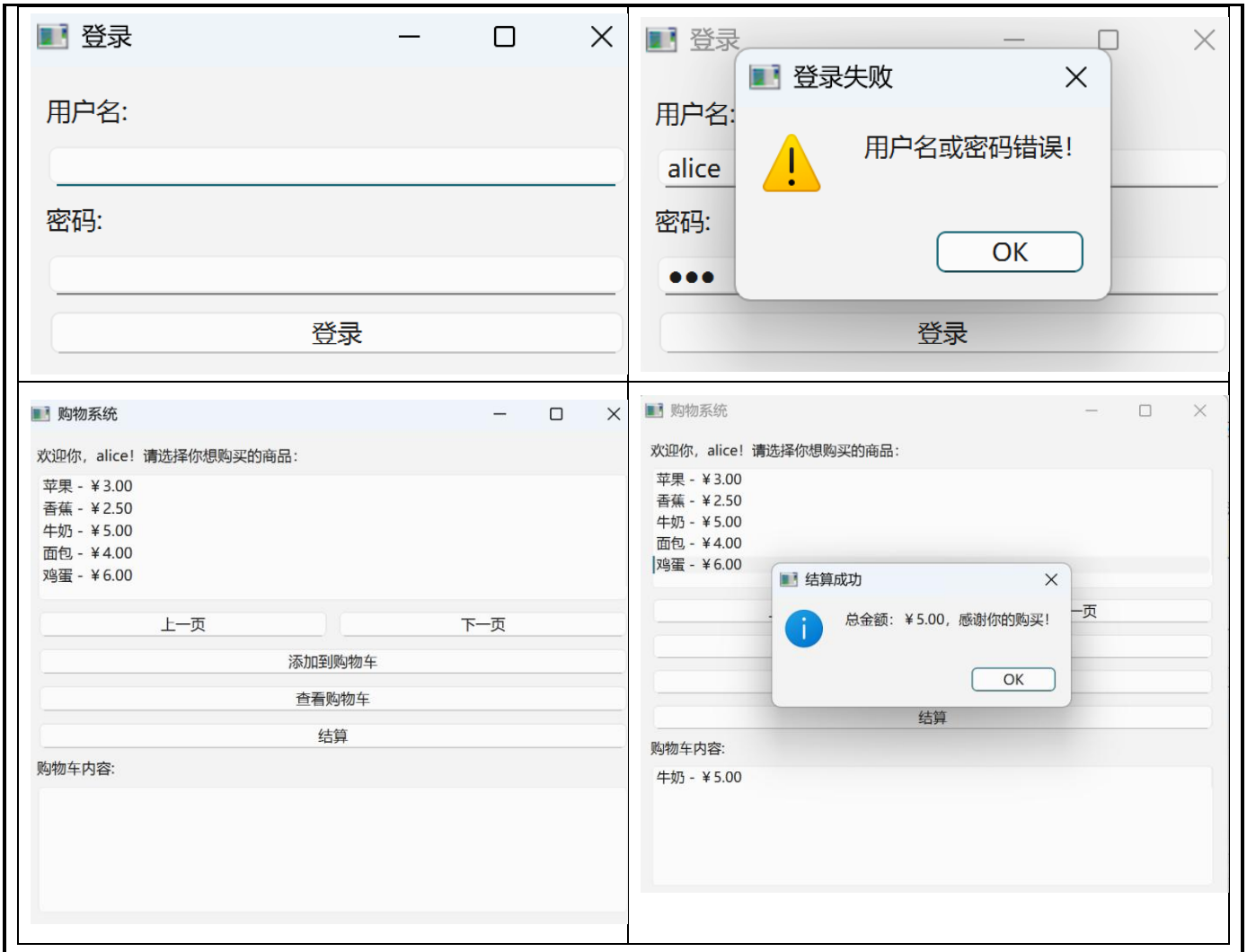
```
        self.password_input = QLineEdit()
        self.password_input.setEchoMode(QLineEdit.EchoMode.Password)

        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()

# 购物窗口
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()

        # 操作按钮
        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)

        # 布局
        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()

# 主程序入口
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：

```
PS C:\Users\Lanyi\Desktop\Project\SXIT-Python_Program> & C:/Users/
Project/SXIT-Python_Program/Lectures/Final_project/DFS.py
[[0, 0], [0, 1], [0, 2], [0, 3], [1, 3], [2, 3], [3, 3]]
PS C:\Users\Lanyi\Desktop\Project\SXIT-Python_Program>
```

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。

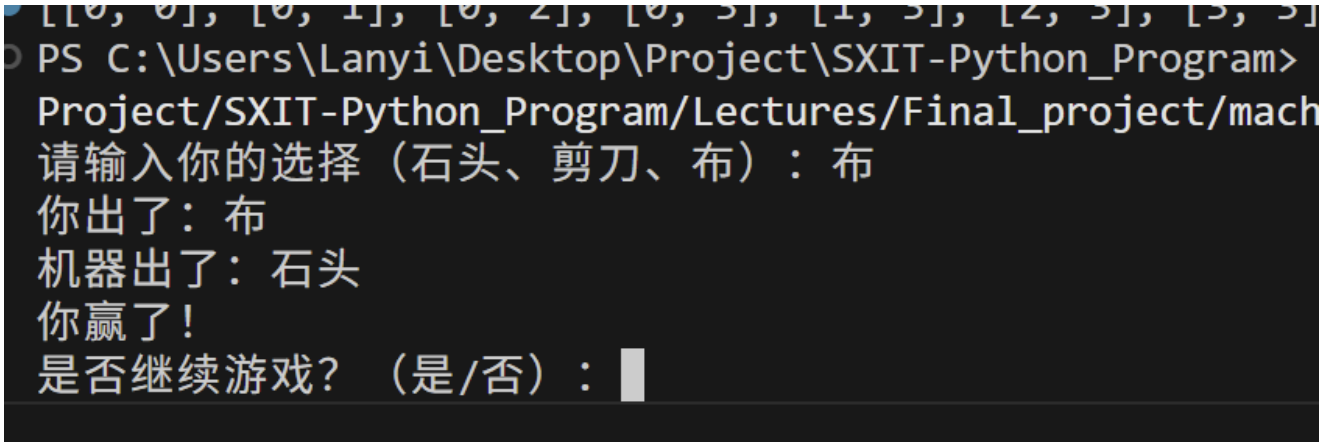
代码如下：

```
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	