**题目一：简易图书馆管理系统**

（一）题目描述：

编写一个简易的图书馆管理系统，该系统应具备以下功能：

1. 添加书籍：用户可以输入书籍的信息（书名、作者、ISBN、借阅状态），系统将该书籍信息保存起来。

2. 查询书籍：用户可以输入书名或作者，系统将显示匹配的书籍信息。

3. 借阅书籍：用户可以输入书名，系统将更新该书籍的借阅状态为“已借出”。

4. 归还书籍：用户可以输入书名，系统将更新该书籍的借阅状态为“可借阅”。

5. 显示所有书籍：用户可以查看系统中所有书籍的详细信息。

6. 退出系统：用户可以选择退出系统。

（二）功能要求

1. 添加书籍：

用户输入书名、作者、ISBN和借阅状态后，系统应将这些信息保存在一个数据结构中。

如果输入的ISBN已经存在，系统应提示“该ISBN的书籍已存在”。

2. 查询书籍：

用户可以选择按书名或作者查询。

输入书名或作者后，系统应显示所有匹配的书籍信息。

如果没有找到匹配的书籍，系统应提示“没有找到匹配的书籍”。

3. 借阅书籍：

用户输入书名后，系统应更新该书籍的借阅状态为“已借出”。

如果输入的书名不存在或该书籍已借出，系统应提示“没有找到该书名的书籍”或“该书籍已借出”。

4. 归还书籍：

用户输入书名后，系统应更新该书籍的借阅状态为“可借阅”。

如果输入的书名不存在或该书籍已归还，系统应提示“没有找到该书名的书籍”或“该书籍已归还”。

5. 显示所有书籍：

系统应列出所有已保存的书籍信息，每条记录前应有编号。

6. 退出系统：

用户可以选择退出系统，系统应显示“感谢使用图书馆管理系统，再见！”并结束程序。

示例数据

为了简化开发，可以使用预定义的数据结构来存储书籍信息。例如：

# 示例书籍数据

books = [

{"title": "Python编程", "author": "张三", "isbn": "1234567890", "status": "可借阅"},

{"title": "数据结构", "author": "李四", "isbn": "0987654321", "status": "已借出"}

]

（三）实现要求

1. 使用控制台菜单提供用户交互：

程序启动时，显示一个菜单，列出所有可用的功能选项。

用户可以通过输入数字选择相应的功能。

2. 使用函数来组织代码：

每个功能都应由一个独立的函数实现。

函数名称应清晰明了，例如 add\_book(title, author, isbn, status), query\_books(keyword), borrow\_book(title), return\_book(title), show\_all\_books()。

3. 使用流程控制语句：

使用 if 和 while 语句来处理用户的输入和选择。

确保程序在用户输入无效时给出适当的提示。

4. 使用数据结构：

使用列表 books 存储每个书籍的详细信息。

每个书籍的信息可以用字典表示。

**题目二：简易学生信息管理系统**

（一）题目描述

编写一个简易的学生信息管理系统，该系统应具备以下功能：

1. 添加学生信息：用户可以输入学生的姓名、年龄和成绩，系统将这些信息保存起来。

2. 查询学生信息：用户可以输入学生的姓名，系统将显示该学生的详细信息。

3. 删除学生信息：用户可以输入学生的姓名，系统将删除该学生的记录。

4. 显示所有学生信息：用户可以查看系统中所有学生的详细信息。

5. 退出系统：用户可以选择退出系统。

（二）功能要求

1. 添加学生信息：

用户输入学生的姓名、年龄和成绩后，系统应将这些信息保存在一个数据结构中。

如果输入的学生姓名已经存在，系统应提示“该学生已存在”。

2. 查询学生信息：

用户输入学生的姓名后，系统应显示该学生的详细信息，包括姓名、年龄和成绩。

如果没有找到匹配的学生，系统应提示“没有找到该学生的信息”。

3. 删除学生信息：

用户输入学生的姓名后，系统应删除该学生的记录。

如果输入的学生姓名不存在，系统应提示“没有找到该学生的信息”。

4. 显示所有学生信息：

系统应列出所有已保存的学生信息，每条记录前应有编号。

5. 退出系统：

用户可以选择退出系统，系统应显示“感谢使用学生信息管理系统，再见！”并结束程序。

示例数据

为了简化开发，可以使用预定义的数据结构来存储学生信息。例如：

# 示例学生数据

students = [

{"name": "张三", "age": 20, "score": 85},

{"name": "李四", "age": 21, "score": 90}

]

（三）实现要求

1. 使用控制台菜单提供用户交互： 程序启动时，显示一个菜单，列出所有可用的功能选项。

用户可以通过输入数字选择相应的功能。

2. 使用函数来组织代码：

每个功能都应由一个独立的函数实现。

函数名称应清晰明了，例如 add\_student(name, age, score), query\_student(name), delete\_student(name), show\_all\_students()。

3. 使用流程控制语句：

使用 if 和 while 语句来处理用户的输入和选择。

确保程序在用户输入无效时给出适当的提示。

4. 使用数据结构：

使用列表 students 存储每个学生的详细信息。

每个学生的信息可以用字典表示。

**题目三：简易餐厅点餐系统**

（一）题目描述

编写一个简易的餐厅点餐系统，该系统应具备以下功能：

1.添加菜品：管理员可以输入菜品的信息（菜品名、价格、库存量），系统将该菜品信息保存起来。

2.查看菜单：顾客可以查看当前所有可用的菜品及其价格。

3.点餐：顾客可以选择菜品并输入数量，系统计算总价并更新菜品库存。

4.取消订单：在结账前，顾客可以取消已点的菜品。

5.结账：顾客可以查看订单详情并支付总价。

6.退出系统：管理员或顾客可以选择退出系统。

（二）功能要求

1.添加菜品：

管理员输入菜品名、价格和库存量后，系统应将这些信息保存在一个数据结构中。

如果输入的菜品名已经存在，系统应提示“该菜品已存在”。

2.查看菜单：

系统应列出所有可用的菜品及其价格，可以按价格排序显示。

3.点餐：

顾客输入菜品名和数量后，系统应检查库存是否足够，并计算总价。

如果输入的菜品名不存在、库存不足或数量输入无效，系统应给出相应的提示。

4.取消订单：

顾客输入要取消的菜品名，系统应从订单中移除该菜品并更新总价和库存。

如果输入的菜品名不存在于当前订单中，系统应提示“没有找到该菜品在订单中”。

5.结账：

系统应显示订单详情，包括菜品名、数量、单价和总价。

顾客确认支付后，系统应模拟支付过程（如打印支付成功信息），并清空订单。

6.退出系统：

管理员或顾客可以选择退出系统，系统应显示“感谢使用餐厅点餐系统，再见！”并结束程序。

示例数据

为了简化开发，可以使用预定义的数据结构来存储菜品信息和订单。例如：

python复制代码

# 示例菜品数据

dishes = [

{"name": "宫保鸡丁", "price": 35, "stock": 50},

{"name": "麻婆豆腐", "price": 25, "stock": 40}

]

# 示例订单数据

order = {

"items": [], # 存储已点菜品的信息，如 {"dish\_name": "宫保鸡丁", "quantity": 2}

"total\_price": 0 # 总价

}

（三）实现要求

使用控制台菜单提供用户交互：

程序启动时，显示一个菜单，列出所有可用的功能选项（包括管理员和顾客的功能）。

用户可以通过输入数字选择相应的功能。管理员和顾客的功能应有所区分。

使用函数来组织代码：

每个功能都应由一个独立的函数实现。

函数名称应清晰明了，例如 add\_dish(dish\_name, price, stock), view\_menu(), place\_order(dish\_name, quantity), cancel\_order(dish\_name), check\_out(), exit\_system()。

使用流程控制语句：

使用 if、elif 和 while 语句来处理用户的输入和选择。

确保程序在用户输入无效时给出适当的提示。

使用数据结构：

使用列表 dishes 存储每个菜品的信息。

使用字典 order 存储订单信息，其中 items 键对应一个列表，存储已点菜品的详细信息；total\_price 键对应一个整数，表示总价。

模拟支付过程：

在结账功能中，模拟支付过程可以简单地通过打印支付成功信息来实现。不需要实际的支付接口。

管理员和顾客的角色区分：

系统应能够区分管理员和顾客的角色，并相应地显示不同的功能选项。这可以通过在程序开始时要求用户输入角色（如“管理员”或“顾客”）来实现。

**题目四：天气预报查询系统**

（一）题目描述：  
创建一个天气预报查询系统。

（二）功能要求：

1.输入城市及当天天气。

2.查询城市天气：输入城市名，查询并显示该城市的当前天气情况（包括温度、湿度、风速、3.天气状况等）。

4.快捷查询：显示常用城市及编号，输入编号，即可显示城市当日天气。

5.保存常用城市：允许用户将查询过的城市保存为常用城市，方便后续快速查询。

6.显示常用城市列表：列出所有已保存的常用城市。

7.删除常用城市：通过输入城市名删除常用城市记录。

8.退出系统：结束程序运行。

（三）实现要求：

1.每个功能点应封装成函数。

2.实现控制台菜单，用户通过输入选项数字选择操作。

3.常用城市列表可以使用列表存储，但需在程序重启后仍能保留。

### **题目五：简易电子商务购物车系统**

（一）题目描述：

设计并实现一个简易的电子商务购物车系统

（二）功能要求：

1.显示商品列表：显示商店提供的所有商品及其价格。

2.添加商品到购物车：用户可以选择商品并输入购买数量，商品被添加到购物车中。

3.查看购物车：显示购物车中所有商品的名称、数量和总价。

4.修改购物车：用户可以修改购物车中商品的数量或删除已选商品。

5.结算购物车：用户确认结算后，系统生成订单号，显示订单详情（包括商品、数量、总价），并模拟扣款成功。

6.退出系统：结束程序运行。

（三）技术要求：

1.使用函数实现每个功能点。

2.在控制台实现一个简单的菜单选择界面，用户通过输入数字选择操作。

3.商品数据可以存储在列表或字典中，每个商品应包含名称、价格和库存数量。

4.购物车数据可以存储在字典中，其中键可以是商品名称，值可以是购买数量。

5.订单数据可以包含订单号、商品列表、总价等信息。

6.订单号可以简单生成，例如使用当前时间戳和随机数。

7.扣款模拟可以简单实现，例如打印一条消息表示扣款成功。

示例数据：

商品数据：[{"name": "手机", "price": 2999, "stock": 10}, {"name": "电脑", "price": 5999, "stock": 5}, ...]

购物车数据（示例）：{"手机": 1, "电脑": 2}

订单数据（示例）：{"order\_id": "2023101612345678", "items": [{"name": "手机", "quantity": 1, "price": 2999}, {"name": "电脑", "quantity": 2, "price": 5999}], "total\_price": 14997}

特殊考虑：

在添加商品到购物车时，应检查库存数量，如果库存不足，应提示用户并拒绝添加。

在结算购物车时，应检查购物车是否为空，如果为空，应提示用户并拒绝结算。

可以添加一些额外的功能点，如商品搜索、用户登录（简单模拟，不使用数据库）等，以提升系统的复杂性和实用性。