课程：面向对象版学员管理系统

目标

- 了解面向对象开发过程中类内部功能的分析方法
- 了解常用系统功能
    - 添加
    - 删除
    - 修改
    - 查询

一. 系统需求

使用面向对象编程思想完成学员管理系统的开发，具体如下：

- 系统要求：学员数据存储在文件中
- 系统功能：添加学员、删除学员、修改学员信息、查询学员信息、显示所有学员信息、保存学员信息及退出系统等功能。

二. 准备程序文件

2.1 分析

- 角色分析
    - 学员
    - 管理系统

> 工作中注意事项
>
> 1. 为了方便维护代码，一般一个角色一个程序文件；
> 2. 项目要有主程序入口，习惯为`main.py`

2.2 创建程序文件

创建项目目录，例如：`StudentManagerSystem`

程序文件如下：

- 程序入口文件：main.py
- 学员文件：student.py
- 管理系统文件：managerSystem.py

三. 书写程序

3.1 student.py

需求：

- 学员信息包含：姓名、性别、手机号；
- 添加`__str__`魔法方法，方便查看学员对象信息

3.1.2 程序代码

```
class Student(object):
    def __init__(self, name, gender, tel):
        self.name = name
        self.gender = gender
        self.tel = tel

    def __str__(self):
        return f'{self.name}, {self.gender}, {self.tel}'
```

3.2 managerSystem.py

需求：

- 存储数据的位置：文件(student.data)
    - 加载文件数据
    - 修改数据后保存到文件
- 存储数据的形式：列表存储学员对象
- 系统功能
    - 添加学员
    - 删除学员
    - 修改学员
    - 查询学员信息
    - 显示所有学员信息
    - 保存学员信息

- 退出系统

3.2.1 定义类

```
class StudentManager(object):
    def __init__(self):
        # 存储数据所用的列表
        self.student_list = []
```

3.2.2 管理系统框架

需求：系统功能循环使用，用户输入不同的功能序号执行不同的功能。

- 步骤
  - 定义程序入口函数
    - 加载数据
    - 显示功能菜单
    - 用户输入功能序号
    - 根据用户输入的功能序号执行不同的功能
  - 定义系统功能函数，添加、删除学员等

```
class StudentManager(object):
    def __init__(self):
        # 存储数据所用的列表
        self.student_list = []

    # 一. 程序入口函数，启动程序后执行的函数
    def run(self):
        # 1. 加载学员信息
        self.load_student()

        while True:
            # 2. 显示功能菜单
            self.show_menu()

            # 3. 用户输入功能序号
            menu_num = int(input('请输入您需要的功能序号：'))

            # 4 根据用户输入的功能序号执行不同的功能
            if menu_num == 1:
                # 添加学员
                self.add_student()
            elif menu_num == 2:
                # 删除学员
```

```
                self.del_student()
            elif menu_num == 3:
                # 修改学员信息
                self.modify_student()
            elif menu_num == 4:
                # 查询学员信息
                self.search_student()
            elif menu_num == 5:
                # 显示所有学员信息
                self.show_student()
            elif menu_num == 6:
                # 保存学员信息
                self.save_student()
            elif menu_num == 7:
                # 退出系统
                break

    # 二. 定义功能函数
    # 2.1 显示功能菜单
    @staticmethod
    def show_menu():
        print('请选择如下功能-----------------')
        print('1:添加学员')
        print('2:删除学员')
        print('3:修改学员信息')
        print('4:查询学员信息')
        print('5:显示所有学员信息')
        print('6:保存学员信息')
        print('7:退出系统')

    # 2.2 添加学员
    def add_student(self):
        pass

    # 2.3 删除学员
    def del_student(self):
        pass

    # 2.4 修改学员信息
    def modify_student(self):
        pass

    # 2.5 查询学员信息
    def search_student(self):
        pass

    # 2.6 显示所有学员信息
    def show_student(self):
        pass
```

```
    # 2.7 保存学员信息
    def save_student(self):
        pass

    # 2.8 加载学员信息
    def load_student(self):
        pass
```

3.3 main.py

```
# 1. 导入managerSystem模块
from managerSystem import *


# 2. 启动学员管理系统
if __name__ == '__main__':
    student_manager = StudentManager()

    student_manager.run()
```

3.4 定义系统功能函数

3.4.1 添加功能

- 需求：用户输入学员姓名、性别、手机号，将学员添加到系统。
- 步骤
  - 用户输入姓名、性别、手机号
  - 创建该学员对象
  - 将该学员对象添加到列表
- 代码

```
# 添加学员函数内部需要创建学员对象，故先导入student模块
from student import *


class StudentManager(object):
    ......

    # 2.2 添加学员
    def add_student(self):
        # 1. 用户输入姓名、性别、手机号
```

```
        name = input('请输入您的姓名：')
        gender = input('请输入您的性别：')
        tel = input('请输入您的手机号：')

        # 2. 创建学员对象：先导入学员模块，再创建对象
        student = Student(name, gender, tel)

        # 3. 将该学员对象添加到列表
        self.student_list.append(student)

        # 打印信息
        print(self.student_list)
        print(student)
```

3.4.2 删除学员

- 需求：用户输入目标学员姓名，如果学员存在则删除该学员。
- 步骤
  - 用户输入目标学员姓名
  - 遍历学员数据列表，如果用户输入的学员姓名存在则删除，否则则提示该学员不存在。
- 代码

```
    # 2.3 删除学员：删除指定姓名的学员
    def del_student(self):
        # 1. 用户输入目标学员姓名
        del_name = input('请输入要删除的学员姓名：')

        # 2. 如果用户输入的目标学员存在则删除，否则提示学员不存在
        for i in self.student_list:
            if i.name == del_name:
                self.student_list.remove(i)
                break
        else:
            print('查无此人！')

        # 打印学员列表，验证删除功能
        print(self.student_list)
```

3.4.3 修改学员信息

- 需求：用户输入目标学员姓名，如果学员存在则修改该学员信息。
- 步骤
  - 用户输入目标学员姓名；

    - 遍历学员数据列表，如果用户输入的学员姓名存在则修改学员的姓名、性别、手机号数据，否则则提示该学员不存在。
- 代码

```
    # 2.4 修改学员信息
    def modify_student(self):
        # 1. 用户输入目标学员姓名
        modify_name = input('请输入要修改的学员的姓名：')
        # 2. 如果用户输入的目标学员存在则修改姓名、性别、手机号等数据，否则提示学员不存在
        for i in self.student_list:
            if i.name == modify_name:
                i.name = input('请输入学员姓名：')
                i.gender = input('请输入学员性别：')
                i.tel = input('请输入学员手机号：')
                print(f'修改该学员信息成功，姓名{i.name},性别{i.gender}，手机号{i.tel}')
                break
        else:
            print('查无此人！')
```

3.4.5 查询学员信息

- 需求：用户输入目标学员姓名，如果学员存在则打印该学员信息
- 步骤
    - 用户输入目标学员姓名
    - 遍历学员数据列表，如果用户输入的学员姓名存在则打印学员信息，否则提示该学员不存在。
- 代码

```
    # 2.5 查询学员信息
    def search_student(self):
        # 1. 用户输入目标学员姓名
        search_name = input('请输入要查询的学员的姓名：')

        # 2. 如果用户输入的目标学员存在，则打印学员信息，否则提示学员不存在
        for i in self.student_list:
            if i.name == search_name:
                print(f'姓名{i.name},性别{i.gender}，手机号{i.tel}')
                break
        else:
            print('查无此人!')
```

3.4.6 显示所有学员信息

- 打印所有学员信息
- 步骤
  - 遍历学员数据列表，打印所有学员信息
- 代码

```
    # 2.6 显示所有学员信息
    def show_student(self):
        print('姓名\t性别\t手机号')
        for i in self.student_list:
            print(f'{i.name}\t{i.gender}\t{i.tel}')
```

3.4.7 保存学员信息

- 需求：将修改后的学员数据保存到存储数据的文件。
- 步骤
  - 打开文件
  - 文件写入数据
  - 关闭文件

> 思考
>
> 1. 文件写入的数据是学员对象的内存地址吗?
> 2. 文件内数据要求的数据类型是什么?

- 拓展`__dict__`

```
class A(object):
    a = 0

    def __init__(self):
        self.b = 1


aa = A()
# 返回类内部所有属性和方法对应的字典
print(A.__dict__)
# 返回实例属性和值组成的字典
print(aa.__dict__)
```

在Python中

- 代码

```python
    # 2.7 保存学员信息
    def save_student(self):
        # 1. 打开文件
        f = open('student.data', 'w')

        # 2. 文件写入学员数据
        # 注意1：文件写入的数据不能是学员对象的内存地址，需要把学员数据转换成列表字典数据再做存储
        new_list = [i.__dict__ for i in self.student_list]
        # [{'name': 'aa', 'gender': 'nv', 'tel': '111'}]
        print(new_list)

        # 注意2：文件内数据要求为字符串类型，故需要先转换数据类型为字符串才能文件写入数据
        f.write(str(new_list))

        # 3. 关闭文件
        f.close()
```

3.4.8 加载学员信息

- 需求：每次进入系统后，修改的数据是文件里面的数据
- 步骤
  - 尝试以`"r"`模式打开学员数据文件，如果文件不存在则以`"w"`模式打开文件
  - 如果文件存在则读取数据并存储数据
    - 读取数据
    - 转换数据类型为列表并转换列表内的字典为对象
    - 存储学员数据到学员列表
  - 关闭文件
- 代码

```python
    # 2.8 加载学员信息
    def load_student(self):
        # 尝试以"r"模式打开数据文件，文件不存在则提示用户；文件存在（没有异常）则读取数据
        try:
            f = open('student.data', 'r')
        except:
            f = open('student.data', 'w')
        else:
            # 1. 读取数据
            data = f.read()

```

```
            # 2. 文件中读取的数据都是字符串且字符串内部为字典数据，故需要转换数据类型再转换字典为对象后存储到学员列表
            new_list = eval(data)
            self.student_list = [Student(i['name'], i['gender'], i['tel']) for i in new_list]
        finally:
            # 3. 关闭文件
            f.close()
```

四. 总结

- 函数
  - 定义和调用
  - 参数的使用
- 面向对象
  - 定义类
  - 创建对象
  - 定义和调用实例属性
  - 定义和调用实例方法
- 数据类型
  - 列表
    - 增加删除数据
    - 列表推导式
  - 字典
  - 字符串
- 文件操作
  - 打开文件
  - 读取或写入
  - 关闭文件