24.9 私有权限

24.9.1 定义私有属性和方法

在 Python中，可以为实例属性和方法设置私有权限，即设置某个实例属性或实例方法不继承给子类。

> 故事：daqiu把技术传承给徒弟的同时，不想把自己的钱（200000个亿）继承给徒弟，这个时候就要为钱这个实例属性设置私有权限。

设置私有权限的方法：在属性名和方法名前面加上两个下划线__。

```
# 1、师傅类：属性和方法
class Master(object):
    def __init__(self):
        self.kongfu = '古法煎饼果子配方'

    def make_cake(self):
        print(f'运用{self.kongfu}制作煎饼果子')

class School(object):
    def __init__(self):
        self.kongfu = 'jjk煎饼果子配方'

    def make_cake(self):
        print(f'运用{self.kongfu}制作煎饼果子')


# 2、徒弟类：继承师傅类 和 学校类， 添加和父类同名的属性和方法
class Prentice(School,Master): # 想要继承谁，就把谁写在第一个位置
    # 加自己的初始化原因：如果不加这个自己的初始化，kongfu属性值是上一次调用的init内的kongfu属性值
    def __init__(self):
        self.kongfu = '独创的煎饼果子技术'
        #self.money = 220000 #
        self.__money = 220000 # 定义私有属性

    # 定义私有方法
    def __info_print(self):
        print('这是私有方法')


    def make_cake(self):
        self.__init__()
        print(f'运用{self.kongfu}制作煎饼果子')

    # 子类调用父类的同名属性和方法：把父类的同名属性和方法再次f封装
    def make_master_cake(self):
        # 父类类名.函数()
        # 再次调用初始化的原因：这里想要调用父类的同名方法和属性，属性在init初始化位置，所以需要再次调用init
        Master.__init__(self)
        Master.make_cake(self)

```

```
    def make_school_cake(self):
        School.__init__(self)
        School.make_cake(self)

class Tusun(Prentice):
    pass

# 3、用徒弟类创建对象，调用实例属性和方法结论验证
xiaoqiu = Tusun()
#print(xiaoqiu.money) # 220000
#xiaoqiu.__info_print() # 这是私有方法
```

24.9.2 获取和修改私有属性值

在 Python中，一般定义函数名 `get_xx` 用来获取私有属性，定义 `set_xx` 用来修改私有属性值。

```
# 1、师傅类：属性和方法
class Master(object):
    def __init__(self):
        self.kongfu = '古法煎饼果子配方'

    def make_cake(self):
        print(f'运用{self.kongfu}制作煎饼果子')

class School(object):
    def __init__(self):
        self.kongfu = 'jjk煎饼果子配方'

    def make_cake(self):
        print(f'运用{self.kongfu}制作煎饼果子')


# 2、徒弟类：继承师傅类 和 学校类， 添加和父类同名的属性和方法
class Prentice(School,Master): # 想要继承谁，就把谁写在第一个位置
    # 加自己的初始化原因：如果不加这个自己的初始化，kongfu属性值是上一次调用的init内的kongfu属性值
    def __init__(self):
        self.kongfu = '独创的煎饼果子技术'
        #self.money = 220000 #
        self.__money = 220000 # 定义私有属性

    # 定义函数：获取私有属性值  get_xx
    def get_money(self):
        return self.__money
    # 定义函数：修改私有属性值 set_xx
    def set_money(self):
        self.__money = 500

    # 定义私有方法
    def __info_print(self):
        print('这是私有方法')


    def make_cake(self):
        self.__init__()
        print(f'运用{self.kongfu}制作煎饼果子')

```

```python
    # 子类调用父类的同名属性和方法：把父类的同名属性和方法再次f封装
    def make_master_cake(self):
        # 父类类名.函数()
        # 再次调用初始化的原因：这里想要调用父类的同名方法和属性，属性在init初始化位置，所以需要再次调用init
        Master.__init__(self)
        Master.make_cake(self)

    def make_school_cake(self):
        School.__init__(self)
        School.make_cake(self)

class Tusun(Prentice):
    pass

# 3、用徒弟类创建对象，调用实例属性和方法结论验证
xiaoqiu = Tusun()
print(xiaoqiu.get_money())

xiaoqiu.set_money()
print(xiaoqiu.get_money())

"""
220000
500
"""
```

二十五、面向对象-其他

目标

- 面向对象三大特性
- 类属性和实例属性
- 类方法和静态方法

25.1 面向对象三大特性

- 封装

  将属性和方法书写到类的里面的操作即为封装

  封装可以为属性和方法添加私有权限

- 继承

  子类默认继承父类的所有属性和方法

  子类可以重写父类属性和方法

- 多态

  传入不同的对象，产生不同的结果

25.2 多态

25.1.1 了解多态

多态指的是一类事物有多种形态，（一个抽象类有多个子类，因而多态的概念依赖于继承）。

- 定义：多态是一种使用对象的方式，子类重写父类方法，调用不同子类对象的相同父类方法，可以产生不同的执行结果
- 好处：调用灵活，有了多态，更容易编写出通用的代码，做出通用的编程，以适应需求的不断变 化!
- 实现步骤：

  定义父类，并提供公共方法

  定义子类，并重写父类方法

  传递子类对象给调用者，可以看到不同子类执行效果不同

25.1.2 体验多态

```
# 需求：警务人员和警犬一起工作，警犬分两种：追击敌人和追查毒品，携带不同的警犬，执行不同的工作

# 1、定义父类，提供公共方法：警犬  和   人
class Dog(object):
    """父类"""
    def work(self):
        pass


# 2、定义子类，子类重写父类方法：定义2个类表示不同的警犬
class ArmyDog(Dog):
    def work(self):
        print('追击敌人....')

class DruDog(Dog):
    def work(self):
        print('追查毒品....')

# 定义人类
class Person(object):
    def work_with_dog(self,dog):
        dog.work()

# 3、创建对象，调用不同的对象，传入不同的对象，执行不同的结果
ad = ArmyDog()
dd = DruDog()

daqiu = Person()
daqiu.work_with_dog(ad)
daqiu.work_with_dog(dd)
```

25.3 类属性和实例属性

25.3.1 类属性

25.3.1.1 设置和访问类属性

- 类属性就是**类对象**所拥有的属性，它被**该类的所有实例对象所共有。**
- 类属性可以使用**类对象**或**实例对象**访问。

```python
# 1、定义类，定义类属性
class Doa(object):
    tooth = 20

# 2、创建对象
wangcai = Doa()
xiaohei = Doa()

# 3、访问类属性：类和对象
print(Doa.tooth) # 20
print(wangcai.tooth) #20
print(xiaohei.tooth) # 20
```

> 类属性的优点
>
> - 记录的某项数据始终保持一致时，则定义类属性。
> - **实例属性**要求**每个对象**为其**单独开辟一份内存空间**来记录数据，而**类属性**为全类所共有，**仅占用一份内存，更加节省内存空间。**

25.3.2 实例属性

修改类属性

类属性只能通过**类对象**修改，不能通过实例对象修改，如果通过实例对象修改类属性，表示的是创建了 一个实例属性。

```python
# 1、定义类，定义类属性
class Dog(object):
    tooth = 20

# 2、创建对象
wangcai = Dog()
xiaohei = Dog()

# 修改类属性
# 1、类 类.类属性 = 值
Dog.tooth = 1000
print(Dog.tooth) # 1000
print(wangcai.tooth) #1000
print(xiaohei.tooth) # 1000

# 2、测试通过对象修改类属性
```

```
wangcai.tooth = 200
print(Dog.tooth) # 20
print(wangcai.tooth) #200
print(xiaohei.tooth) # 20
```

25.3.2 类方法和静态方法

25.3.2.1 类方法的特点

需要用装饰器 `@classmethod` 来标识其为类方法，对于类方法，**第一个参数必须是类对象**，一般以 `cls` 作为第一个参数。

25.3.2.2 类方法使用场景

- 当方法中**需要使用类对象**（如访问私有类属性等）时，定义类方法
- 类方法一般和类属性配合使用

```
# 1、定义类：私有类属性，类方法获取私有类属性
class Dog(object):
    __tooth = 10

    # 定义类方法
    @classmethod
    def get_tooch(cls):
        return cls.__tooth

# 2、创建对象，调用类方法
wangcai = Dog()
result = wangcai.get_tooch()
print(result) # 10
```

25.3.2.3 静态方法

25.3.2.3.1 静态方法的特点

- 需要通过装饰器 `@staticmethod` 来进行修饰，**静态方法既不需要传递类对象也不需要传递实例对象 （形参没有 `self/cls` ）。**
- 静态方法也能够通过**实例对象**和**类对象**去访问。

25.3.2.3.2 静态方法使用场景

- 当方法中**既不需要使用实例对象**（如实例对象，实例属性），**也不需要使用类对象**（如类属性、类方 法、创建实例等）时，定义静态方法
- **取消不需要的参数传递**，有利于**减少不必要的内存占用和性能消耗。**

```python
# 1、定义类：定义静态方法
class Dog(object):
    @staticmethod
    def info_print():
        print('这是一个静态方法')
# 2、创建对象
wangcai = Dog()

# 3、调用静态方法：类 和 对象
wangcai.info_print()
Dog.info_print() # 静态方法也可以通过类来调用
```

二十六、异常

目标

- 了解异常
- 捕获异常
- 异常的else
- 异常 finally
- 异常的传递
- 自定义异常

26.1 了解异常

当检测到一个错误时，解释器就无法继续执行了，反而出现了一些错误的提示，这就是所谓的"异常"。 例如：以 `r` 方式打开一个不存在的文件。

26.2 异常的写法

26.2.1 语法

```python
try:
    可能发生错误的代码
except:
    如果发生异常执行的代码
```

26.2.1 快速体验

需求：尝试以 `r` 模式打开文件，如果文件不存在，则以 `w` 方式打开

```python
try:
    f = open('test.txt','r')
except:
    f = open('text.txt','w')
```

26.3 捕获指定异常

26.3.1 语法

```
try:
    可能发生错误的代码
except 异常类型:
    如果捕获到该异常类型执行的代码

```

26.3.2 体验

```
try:
    print(num)
except NameError:
    print('有错误')
```

> 注意:
>
> 1. 如果尝试执行的代码的异常类型和要捕获的异常类型不一致，则无法捕获异常。
> 2. 一般try下方只放一行尝试执行的代码。

26.3.3 捕获多个指定异常

当捕获多个异常时，可以把要捕获的异常类型的名字，放到 except后，并使用元组的方式进行书写。

```
try:
    print(1/0)
except (NameError,ZeroDivisionError):
    print('有错误')
```

26.3.4 捕获异常描述信息

```
try:
    print(num)
except (NameError,ZeroDivisionError) as result:
    print(result)
```

26.3.5 捕获所有异常

Exception是所有程序异常类的父类。

```
try:
    print(num)
except Exception as result:
    print(result)
```

26.4 异常的else

else表示的是如果没有异常要执行的代码。

```
try:
    print(1)
except Exception as result:
    print(result)
else:
    print('我是else，当没有异常的时候执行的代码')

"""
1
我是else，当没有异常的时候执行的代码

"""
```

26.5 异常的finally

finally表示的是无论是否异常都要执行的代码，例如关闭文件。

```
try:
    f = open('text.txt', 'r')
except Exception as result:
    f = open('test.txt', 'w')
else:
    print('没有异常，真开心')
finally:
    f.close()
```

26.6 异常的传递

需求：

1、尝试只读方式打开test.txt文件，如果文件存在则读取文件内容，文件不存在则提示用户即可。

2、读取内容要求：尝试循环读取文件，读取过程中如果检测到用户意外终止程序，则except捕获异常并提示用户。

```
# 需求1：尝试只读打开test.txt 文件存在读取内容，不存在提示用户
# 需求2：读取内容：循环读取，当无内容的时候退出循环，如果用户意外终止，提示用户已经被意外终止。
import time
try:
    f = open('test.txt')
    try:
        while True:
            content = f.readline()
            if len(content) ==0:
                break
            time.sleep(2)
```

```
            print(content)
    except:
        # 如果在读取文件的过程中，产生了异常，那么就会捕获到
        # 比如，按下了ctrl+c
        print('意外终止了读取数据')
    finally:
        f.close()
        print('关闭文件')
except:
    print('文件不存在')
```

26.7 自定义异常

在 Python中，抛出自定义异常的语法为 `raise` 异常类对象。

需求：密码长度不足，则报异常（用户输入密码，如果输入的长度不足3位，则报错，即抛出自定义异常，并捕获该异常）。

```
# 1、自定义异常类：继承Exception，魔法方法有init和str(设置异常描述信息)
# 2、抛出异常：尝试执行：用户输入密码，如果长度小于3，抛出异常
# 3、捕获异常

# 自定义异常类，继承Exception
class ShortInputError(Exception):
    def __init__(self,length, min_len):
        #用户输入的密码长度
        self.length = length
        #系统要求的最少长度
        self.min_len = min_len

    # 设置抛出异常的描述信息
    def __str__(self):
        return f'你输入的长度是{self.length},不能少于{self.min_len}个字符'

def main():
    try:
        con = input('请输入密码：')
        if len(con)<3:
            raise ShortInputError(len(con),3)
    except Exception as e:
        print(e)
    else:
        print('输入密码完成')

main()
```

26.8 异常总结

- 异常语法

```
try:
    可能发生异常的代码
except:
    如果发出异常执行的代码
else:
    没有异常执行的代码
finally:
    无论是否异常都要执行的代码
```

- 捕获异常

```
except 异常类型:
    代码
except 异常类型 as xx:
    代码
```

- 自定义异常

```
# 1、自定义异常类
class 异常类类名(Exception):
    代码
    #设置抛出异常的描述信息
    def __str__(self):
        return ...
# 2、抛出异常
raise 异常类名()
# 捕获异常
except Exception
```

二十七、模块和包

目标

- 了解模块
- 导入模块
- 制作模块
- `__all__`
- 包的使用方法

27.1 模块

Python模块（Module），是一个 Python文件，以py结尾，包含了 Python对象定义和 Python语句。

模块能定义函数，类和变量，模块里也能包含可执行的代码。

27.1.1 导入模块

27.1.1.1 导入模块的方式

- import 模块名
- from 模块名 import 功能名
- from 模块名 import *
- import 模块名as 别名
- from 模块名 import 功能名 as 别名

27.1.2 导入方式详解

27.1.2.1 import

- 语法

```
# 1、导入模块
import 模块名
import 模块名1，模块名2...

# 2、调用功能
模块名.功能名()
```

- 体验

```
import math
print(math.sqrt(9)) # 3.0
```

27.1.2.2 from...import...

- 语法

```
from 模块名 import 功能1，功能2，功能3...
```

- 体验

```
from math import sqrt
print(sqrt(9))
```

27.1.2.3 from..import *

- 语法

```
from 模块名 import *
```

- 体验

```python
from math import *
print(sqrt(9))
```

27.1.2.4 as定义别名

- 语法

```python
#模块定义别名
import 模块名 as 别名

# 功能定义别名
from 模块名 import 功能 as 别名
```

- 体验

```python
# 模块别名
import time as tt
tt.sleep(2)
print('hello')

# 功能别名
from time import sleep as sl
sl(2)
print('hello')
```

27.1.3 制作模块

在 Python中，每个 Python文件都可以作为一个模块，模块的名字就是文件的名字。**也就是说自定义模块名必须要符合标识符命名规则。**

27.1.3.1 定义模块

新建一个python文件，命名为 `my_module1.py` ，并定义 `testA` 函数。

```python
def testA(a,b):
    print(a+b)
```

27.1.3.2 测试模块

在实际开中，当一个开发人员编写完一个模块后，为了让模块能够在项目中达到想要的效果，这个开发 人员会自行在py文件中添加一些测试信息，例如，在 `my_module1.py` 文件中添加测试代码。

```python
def testA(a,b):
    print(a+b)

testA(1,1)

```

此时，无论是当前文件，还是其他已经导入了该模块的文件，在运行的时候都会自动执行 `testA` 函数的调用。 解决办法如下：

```
def testA(a,b):
    print(a+b)

# 只在当前文件中调用该函数，其他导入的文件内不符合该条件，则不执行testA函数调用
#__name__是系统变量，是模块的标识符，值是：如果是自身模块值是__main__,否则是当前模块的名字
if __name__ == '__main__':
   testA(1,1)
```

27.1.3.3 调用模块

```
# 1、导入模块
import my_module1
# 2、调用功能
my_module1.testA(2,2)
```

27.1.4 模块定位顺序

当导入一个模块， Python解析器对模块位置的搜索顺序是：

1. 当前目录
2. 如果不在当前目录， Python则搜索在shell变量PYTHONPATH下的每个目录。
3. 如果都找不到， Python会察看默认路径。UNIX下，默认路径一般为/usr/local/lib/python/

模块搜索路径存储在system模型的sys.path变量中。变量里包含当前目录，PYTHONPATH和由安装过程决定的默认目录。

- **注意**

  自己的文件名不要和已有的模块名重复，否则导入模块功能无法使用（大概意思就是你本地文件同目录下，不要有和模块名重复的文件名）

  使用from 模块名 import 功能 的时候，如果功能名字重复，调用到的是最后定义或导入的功能。

名字重复的严重性

```
问题：import 模块名  是否担心  功能名字重复的问题 ------不需要
import time
print(time)

time = 1
print(tim3) #1


# 问题：为什么变量也能覆盖模块？      -----在python语言中，数据是通过  引用  传递的。
```

27.1.5 `__all__` 列表

如果一个模块文件有 `__all__` 变量，当使用 `from xxx import *` 导入时，只能导入这个列表中的元素。

- my_module1模块代码

```
__all__ = ['testA']

def testA():
    print('testA')

def testB():
    print('testB')
```

说明：本来我们能导入这个模块中的所有功能，奈何模块中有：`__all__` ，导致只能导入 `__all__` 列表中的内容。

- 导入模块的文件代码

```
from my_module1 import *

testA()

# 因为testB函数没有添加到all函数，只有all列表里面的功能才能导入
# testB()  # NameError: name 'testB' is not defined
```

27.2 包

包将有联系的模块组织在一起，即放到同一个文件夹下，并且在这个文件夹创建一个名为 `__init__.py` 文件，那么这个文件夹就称之为包。

27.2.1 制作包

[new]—[python package]—输入包名—[ok]—新建功能模块(有联系的模块)

注意：新建包后，包内会自动创建 `__init__.py` 文件，这个文件控制这包的**导入行为。**

27.2.2 快速体验

1. 新建包 `mypackage`

2. 新建包内模块：`my_module1` 和 `my_module2`
3. 模块内代码加下

```python
# my_module1
print(1)

def info_print1():
    print('my_module1')
```

```python
# my_module2
print(2)

def info_print1():
    print('my_module2')
```

27.2.3 导入包

27.2.3.1 方法一

- 语法

```
import 包名.模块名

包名.模块名.目标
```

- 体验

```python
import my_package.my_module1

my_package._my_module1.info_print1()
```

27.2.3.2 方法二

注意：必须在 `__init__.py` 文件中添加 `__all__=[]`，**控制允许导入的模块列表。**

- 语法

```
from 包名 import *
模块名.目标
```

- 体验

```python
# __init__.py文件:
__all__ = ['my_module1']

# 测试文件
from my_package import *
my_module1.info_print1()

"""
1
my_module1
"""
```

二十八、面向对象版学院管理系统

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

28.1 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。

28.2 准备程序文件

28.2.1 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

28.2.2 创建程序文件

创建项目目录，例如：`StudentManageSystem`

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：manageSystem.py

28.2.3 书写程序

28.2.3.1 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

程序源码：

```
"""
 author:jjk
 datetime:2020/4/25
 coding:utf-8
 project name:Pycharm_workstation
 Program function:

"""
"""
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

"""

class Student(object):
    def __init__(self,name, gender,tel):
        # 学员信息包含：姓名、性别、手机号；
        self.name = name
        self.gender = gender
        self.tel = tel
    # 添加 __str__ 魔法方法，方便查看学员对象信息
    def __str__(self):
        return f'{self.name}, {self.gender}, {self.tel}'

# 测试代码
# aa = Student('aa','nv',121)
# print(aa)
```

28.2.3.2 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

定义类:

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

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

  根据用户执行不同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

```
# 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
        while True:
            # 2、显示功能菜单
            # 3、用户输入功能序号
            menu_num = int(input('您输入的功能序号：'))

            # 4、根据用户输入的序号执行不同的功能
            if menu_num ==1:
                # 添加学员
                pass
            elif menu_num ==2:
                # 删除学员
                pass
            elif menu_num ==3:
                # 修改学员信息
```

```
                pass
            elif menu_num ==4:
                # 查询学员信息
                pass
            elif menu_num ==5:
                # 显示所有学员信息
                pass
            elif menu_num ==6:
                # 保存学员信息
                pass
            elif menu_num ==7:
                # 退出系统----退出循环
                break
```

更新

```
"""
 author:jjk
 datetime:2020/4/25
 coding:utf-8
 project name:Pycharm_workstation
 Program function:

"""

class StudentSystem(object):
    def __init__(self):
        # 存储学员数据所用的列表
        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.add_student()
        while True:
            # 2、显示功能菜单
            self.show_menu() # 类里面调用方法self
            # 3、用户输入功能序号
            menu_num = int(input('您输入的功能序号：'))

            # 4、根据用户输入的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_menu()
            elif menu_num ==6:
                # 保存学员信息
                self.save_student()
            elif menu_num ==7:
                # 退出系统----退出循环
                break


    # 二、系统功能函数
    # 2.1 显示功能菜单 -- 打印序号的功能对应关系 ---静态
    @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):
        """添加学员"""
        print('添加学员')
    # 2.3 删除学员
    def del_student(self):
        print('删除学员')
    # 2.4 修改学员信息
    def modify_student(self):
        print('修改学员信息')
    # 2.5 查询学员信息
    def search_student(self):
        print('查询学员信息')
    # 2.6 显示所有学员信息
    def show_student(self):
        print('显示所有')
    # 2.7 保存学员信息
    def save_student(self):
        print('保存学员信息')

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

28.2.3.3 main.py

```
"""
 author:jjk
 datetime:2020/4/25
 coding:utf-8
 project name:Pycharm_workstation
```

```
 Program function:

"""

# 1、导入managerSystem模块
from managerSystem import *
# 2、启动学员管理系统
# 保证是当前文件运行才启动管理系统  if --创建对象并调用run方法
if __name__ == '__main__':
    student_manager = StudentSystem()
    student_manager.run()
```

28.2.3.4 定义系统功能函数

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

```
# 2.2 添加学员
def add_student(self):
    """添加学员"""
    #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  --类？ 类在student文件中，先导入student模块，再创建对象
    student = Student(name,gender,tel)
    #3、将该对象添加到学员列表
    self.student_list.append(student)
    print(self.student_list)
    print(student)
```

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
    def del_student(self):
        #1、用户输入目标学员姓名
        del_name = input('请输入需要删除的学员：')
```

```
        #2、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。
        for i in self.student_list:
            if del_name == i.name:
                # 删除该学员对象
                self.student_list.remove(i)
                break
        else:
            # 循环正常结束执行的代码：循环结束都没有删除任何一个对象，所以说明用户输入的目标学员不存在
            print('查无此人')
        print(self.student_list)
```

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):
    # 用户输入目标学员姓名；
    modify_name = input('请输入要修改的学员姓名：')

    #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。
    for i in self.student_list:
        if modify_name == i.name:
            i.name = input('请输入您的姓名：')
            i.gender = input('请输入您的性别：')
            i.tel = input('请输入您的手机号：')
            print(f'修改学员信息成功，  姓名：{i.name}, 性别：{i.gender},手机号：{i.tel}')
            break
            else:
```

查询学员信息功能

- 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

```python
# 2.5 查询学员信息
    def search_student(self):
        print('查询学员信息')
        # 1、用户输入目标学员姓名
        search_name = input('请输入要查询的学员姓名：')
        # 2、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。
        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')
```

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

```python
    # 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}')
```

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__`：**收集类对象或者实例对象的属性和方法以及对应的值**

```python
#1、定义类
#2、创建对象
#3、调用__dict__
class A(object):
    a = 0 # 类属性
    def __init__(self):
        self.b = 1 # 实例属性

aa = A()
# {'__module__': '__main__', 'a': 0, '__init__': <function A.__init__ at 0x000001C220638C80>, '__dict__': <attribute '__dict__' of 'A' objects>, '__weakref__': <attribute '__weakref__' of 'A' objects>, '__doc__': None}....
print(A.__dict__)
print(aa.__dict__) # {'b': 1}
```

保存学员数据

```python
    # 2.7 保存学员信息
    def save_student(self):
        # 1、打开文件
        f = open('student.data', 'w')
        # 2、文件写入数据
        # 2.1 [学员数据] 转换成 [字典]
        new_list = [i.__dict__  for i in self.student_list] # [{'name': 'jbji', 'gender': 'dfdf', 'tel': 'dfdf'}]
        # 2.2 文件写入  字符串数据
        f.write(str(new_list))
        # 3、关闭文件
        f.close()
```

加载学员数据

- 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):
    """加载学员数据"""
    # 1、打开文件：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
    try:
        f = open('student.data', 'r')
```

```
        except:
            f = open('student.data', 'w')
            else:
                # 如果文件存在则读取数据并存储数据
                # 读取数据：文件读取的数据是字符串，还原成列表类型： [{}] 转换 [学员对象]
                data = f.read() # 字符串
                new_list = eval(data) # 还原成原本的列表形式
                self.student_list = [Student(i['name'], i['gender'], i['tel']) for i in new_list]

                # 3、关闭文件
                finally:
                    f.close()
```