day17：面向对象编程

笔记本：	Python基础		
创建时间：	2018/6/28 17:23	**更新时间：**	2018/7/2 19:39
作者：	liuchang_0412@163.com		

1.面向对象编程：面向对象编程是一种思想
①支持面向对象编程的语言有：C++, Java, Python, Swift, C# ....
②两个概念：
类：class
对象：对象 object/ 实例 instance
示例：

```
a = int(100)
a = str("hello")
```

2.类：
①类的创建语法：

```
class 类名 (继承列表):
    '类文档字符串'
    实例方法(类内的函数method) 定义
    类变量(class variable) 定义
    类方法(@classmethod) 定义
    静态方法(@staticmethod) 定义
```

②类的作用：
可以用类来创建对象（实例）
类内定义的变量和方法能被此类所创建的所有实例共同拥有
类通常用来创建具有共同属性的对象（实例）
③实例创建的表达式：
类名([创建传参])
作用：创建一个类的实例对象，并返回此实例
说明：
实例有自己的作用域和名字空间，可以为实例添加变量
实例可以调用类中的方法
实例可以访问类中的类变量
④实例变量：
调用语法：**实例.变量名**
在模块中调用：**模块名.实例.变量名**

#最简单的类：

```
class Dog:
    def infos(self):
        print("狗的种类",self.kinds,"狗的颜色",self.color)
```

#创建一个实例：

```
dog1 = Dog( )
dog1.kinds = "京巴"
dog1.color = "白色"
dog1.info( )

dog2 = Dog( )
dog2.kinds = "藏獒"
dog2.color = "棕色"
```

#练习：

1.写一个程序，用input函数读取一些文字（包含文字），将这些文字写入两个文件中，"GBK.txt"用于存储GBK编码的文档，"UTF_8.txt"用于存储UTF-8编码的文档。将以上两个文件复制到windows下，再用记事本打开查看内容，是否正确。

2.自己写一个类Student,此类的对象有属性name,age,score，用来保存学生的姓名，年龄，成绩，用input读入5个学生的相关信息，用对象来存储这些信息（不用字典），打印五个学生信息。

3.实例方法：

①创建方法

```
class 类名(继承列表):
    def 实例方法名(self, 形式参数1, 形式参数2, ...):
        "文档字符串"
        语句块
```

说明：

实例方法的实质是函数，是定义在类内的函数

实例方法属于类的属性

实例方法的第一个参数代表调用这个实例方法的对象，一般命名为"self"

实例方法如果没有return语句，则返回None

示例：

见dog.py

②实例方法的调用语法：

```
实例.实例方法名(调用参数)
或
类名.实例方法名(实例,调用参数)
```

#练习：自己定义一个类Human( )

有两个属性：姓名(name)，年龄(age)。

有三个方法：设置姓名(set_name)，设置年龄(set_age)，显示信息(infos)。

用此类来创建两个对象：

张三,20岁

李四,21岁

4.构造方法：创建对象时初始化实例变量

语法：

def __init__(self [,形式参数]):

语句块

说明：

1.构造方法名必须为__init__，不可改变

2.在一个类中只能有一个__init__构造方法（有多个时，最后一个起作用）

3.构造方法会在实例创建时自动调用，且将实例自身通过第一个参数self传入__init__方法

4.构造方法如果没有return语句，则返回self自身

5.析构方法：

语法：

__del__(self):

语句块

说明：

1.析构方法会在对象销毁时被自动调拥

2.Python语言不建议在对象销毁时做任何事情，因为此方法的调用时间难以确定

#练习：创建一个车类(Car)，车有属性：品牌(brand)，型号(model)，颜色(color)

车的方法有：

run(speed)　#以X公里/小时的速度行驶

infos( )　#显示属性信息

change_color(c)　#改变车的颜色

例：

```
a4 = Car("红色", "奥迪", "A4")
a4.run(199)　#红色奥迪A4正在以199km/h的速度行驶
a4.change_color("黑色")
a4.run(230)　#黑色奥迪A4正在以230km/h的速度行驶
```

6.预制实例属性：

__dict__属性：

每一个对象（实例）都有一个__dict__属性，__dict__属性绑定一个存储此实例自身的属性字典

__doc__属性：

用于保存类的文档字符串，用help( )中显示实例的文档字符串，和类的文档字符串相同

__class__属性：

class属性绑定创建此对象（实例）的类对象

作用：

1.可以借助于此属性来创建同类的实例
2.可以借助于此属性来访问类变量

__model__属性：
绑定此实例所属的模块
在主模块当中，此值为'__main__'，不在主模块当中，此值为模块名。

6.类变量：
1）是指在类class内定义的变量，此变量属于类，不属于此类的对象
2）类变量，可以通过该类直接使用
3）类变量，可以通过类的实例直接访问
4）类变量可以通过此类的对象的__class__属性简介访问

__slots__属性：
作用：
1.限定一个类创建的实例，只能有固定的实例属性，不允许对象添加列表以外的实例属性
2.防止用户因错写属性名而发生程序错误
说明：
__slots__属性是一个列表，列表的值是字符串
含有__slots__属性的类所创建的实例对象没有__dict__属性，即此实例不用字典来存储属性

7.对象（实例）的属性管理：
函数：
1) **getattr(obj, name, [,default])** 从一个对象得到对象 ， getattr(x,'y') 等于x.y。当属性不存在时，如果给出default，则返回default，如果没有给出default，则产生一个AttributeError错误
2) **hasattr(obj, name)** 用给定的name返回对象obj是否有此属性，此做法可以避免getattr( )函数引发错误
3) **setattr(obj, name ,value)** 给对象obj的名为name的变量设置相应的值，setattr(x,'y',v)等同于x.y=v
4) **delattr(obj, name)** 删除对象obj的name属性，delattr(x,'y') 等同于 del x.y

8.用于类的函数：
isinstance(obj, 类或类的元组) 返回这个对象obj是否为某个类或某些类的对象。
type(obj) 返回对象的类型 # type(1) --> int type('1') --> str

9. 类方法 @classmethod
定义：
类方法是只能访问类变量的方法
类方法需要使用@classmethod装饰器定义
类方法的第一个参数是类的示例，约定写为cls

说明：
类实例和对象实例都可以调用类方法
类方法不能访问实例变量
示例：
见classmethod.py

类方法和实例方法的对比：
1.类方法能够访问类变量，不能访问实例变量；实例方法可以访问类变量和实例变量；
2.类方法可以用实例调用也可以用类调用，实例方法只能用实例调用。

10.静态方法 @staticmethod
定义：
静态方法是普通的函数
静态方法定义在类的内部，只能凭借该类和实例调用
静态方法需要使用@staticmethod装饰器定义
静态方法与普通函数定义相同，不需要传入self实例参数和cls类参数
说明：
类实例和对象实例都可以调用静态方法
静态方法不能访问类变量和实例变量
示例：
staticmethod.py

#实例方法，类成员方法，静态方法总结
不想访问类变量和实例变量（属性），用静态方法
只想访问类内变量，不想访问实例属性用类成员方法
既想访问类内变量，也想访问实例变量用实例方法

11.特性属性 @property
作用：
用来模拟一个属性
通过@property装饰器可以对模拟属性赋值和取值加以控制
实现其他语言所拥有的 getter 和 setter 功能
示例：
见property.py

#练习：
1.写一个正方形类，有三个属性：边长(l)，周长(c)，面积(a)。
用此类生成对象，此三个属性，其中一个变化，其余的会跟着变化。（用@property实现）
2.修改学生管理程序，用类和对象存储学生信息，每个学生有：
infos( )方法：用于显示学生信息