编程思想:面向过程

面向过程一般适用于编写 小系统（业务流程不复杂），使用函数形式组织脚本。优点：编码比较简单， 缺点：无法应对大型复杂的系统。

例子 ：游戏系统

面向过程开发（函数）： 1.打怪 2.升级 3.PK 4.养宠物。。。。。

面向对象开发：1.战士（对象） 2.怪兽（对象） 3.猫咪（对象）

不同对象组合，比如战士与怪兽组合，可以产生打怪的场景；战士与怪兽组合，也可以产生升级的场景；战士与战士组合，可以产生PK的场景；战士与猫咪组合，可以产生养宠物的场景。

面向对象：

优点： 擅长于 开发比较复杂的系统

缺点： 复杂，学习成本高

举例: 张三 炒菜

面向过程:

定义一个函数(炒菜) ==> 传入参数（张三），然后执行函数

面向对象:（从上帝角度出发）

1. 创建人类
2. 在人类中指定张三
3. 张三 炒菜

**（重要）面向对象的一般方法:**

1. **对事物进行抽象==》创建类 class**
2. **实例化==》创建对象**
3. **调用对象的方法**

类的组成

1.属性 （强调静态） 姓名 年龄 性别……

2.方法 （函数--强调动态） 炒菜 工作……

类是抽象出来的，不是具体的东西，是对象的模板，强调事物的共性。 类中属性的值是没有确定的，只是声明了从类中产生的对象都有该属性，属性在类中的任意地方可以使用。

对象具备类中的属性和方法，是对类的具体实现，强调事物的个性。

**面向对象方式实现炒菜****#1.对事物进行抽象化，定义类****class** person:  
 *#1.规定属性* **def** \_\_init\_\_(self,name,age,id):  
 self.name=name  
 self.age=age  
 self.id=id  
  
 **def** chaocai(self,cai):  
 *#输出谁在炒什么菜* print(self.name,**"在炒"**,cai)  
  
 **def** work(self,profession):  
 *#输出谁是从事什么职业* print(self.name,**"是从事"**,profession)  
  
**#2.实例化**p1=person(**"张三"**,18,420684)  
p2=person(**"李四"**,35,230605)  
  
**#3.调用对象的方法**p1.chaocai(**"辣子鸡"**)  
p1.work(**"厨师"**)  
print(p1.name)  
p2.chaocai(**"宫保鸡丁"**)  
p2.work(**"软件测试工程师"**)  
print(p2.name)

**1.类定义语法**

1. **使用class关键字来定义类，class关键字之后是一个空格，然后是类的名字，再然后是一个冒号，最后换行并定义类的内部实现。**

**class** person:

1. **规定属性==》构造函数\_\_init\_\_**

**（1）作用：\_\_init\_\_ 构造函数（方法）专门用于构造实例化对象**

**（2）形参： self和其他自定义参数，其中self 必须要有，且必须是第一个形参，self 参数代表将来要创建的对象本身，调用函数时不管该形参**

**（3）类的属性: self.属性=形参 实现类的实例属性的定义**

**def** \_\_init\_\_(self,name,age,id):  
 self.name=name  
 self.age=age  
 self.id=id

1. **定义类的方法**

**def** chaocai(self,cai):  
 *#输出谁在炒什么菜* print(self.name,**"在炒"**,cai)  
  
**def** work(self,profession):  
 *#输出谁是从事什么职业* print(self.name,**"是从事"**,profession)

说明：**self参数**

* + **类的所有方法都必须至少有一个self 的参数，并且必须是方法的第一个形参，self 参数代表将来要创建的对象本身**
  + **在类的方法中访问属性时需要以self 为前缀，也就是self.属性**
  + **在外部通过对象名调用对象方法时不需要传递self参数**

**2.类实例化语法**

**通过类名（调用构造函数）实现实例化，传入属性的具体值**

p1=person(**"张三"**,18,420684)  
p2=person(**"李四"**,35,230605)

**3.对象调用方法语法**

**通过 对象名.方法 的方式来访问类中的属性或方法**

**也可以使用类名调用方法 格式:类名.方法名(self,其他形参) 注意：需要传入self形参，也就是传入对象。例：**person.chaocai(p1,**"辣子鸡"**)

p1.chaocai(**"辣子鸡"**) #访问对象的方法

person.chaocai(p1,**"辣子鸡"**) #另一种调用方法，通过类访问对象的方法

p1.work(**"厨师"**)  
print(p1.name) #访问对象的属性  
p2.chaocai(**"宫保鸡丁"**)  
p2.work(**"软件测试工程师"**)  
print(p2.name)

**案例：**

**使用面向对象的思想 开发一个游戏 ：**

**对象 张飞 男 18 初始战斗力 1000**

**吕布 男 20 初始战斗力 2000**

**曹操 男 35 初始战斗力 2500**

**场景(功能):**

**草丛战斗 消耗 200 战斗力**

**自我修炼 增长 100 战斗力**

**多人对战 消耗 500 战斗力**

1. 抽象类 ====》人类

属性： 姓名 name

性别 gender

年龄age

战斗力fight

方法： 草丛战斗 grassland

自动修炼 practice

多人对战 incest

1. 实例化对象
2. 调用对象方法

**代码：**

**class** person:  
 **def** \_\_init\_\_(self,name,gender,age,fight):  
 self.name=name  
 self.gender=gender  
 self.age=age  
 self.fight=fight  
 print(self.name,**"创建成功"**)  
  
 **def** grassland(self):  
 self.fight-=200  
  
 **def** practice(self):  
 self.fight+=100  
  
 **def** incest(self):  
 self.fight-=500  
  
 *#查看对象信息* **def** info(self):  
 print(**"姓名：%s,性别：%s,年龄：%d,战斗力：%d"**%(self.name,self.gender,self.age,self.fight))  
  
*#2.实例化*p1=person(**"张飞"**,**"男"**,18,1000)  
p2=person(**"吕布"**,**"男"**,20,2000)  
p3=person(**"曹操"**,**"男"**,35,2500)  
  
*#3.调用对象方法*print(**"开始玩游戏啦！！！"**)  
p1.grassland()  
p2.practice()  
p1.incest()  
print(**"游戏结束！该去搬砖了！！！"**)  
p1.info()  
p2.info()  
p3.info()

**面向对象的特征: 封装、继承、多态**

**1.封装**

需求：学生类

个人信息属性：name age grade

但是，希望选择部分属性（name，age）公开，选择部分属性（grade）不公开

对于类来说，什么是公开？什么是不公开呢？

公开：在类的外面可以任意访问属性 ==》公有属性

不公开：在类的外面不能访问 ==》私有属性

解决： 使用面向对象的封装特征，将类中的属性封装起来，不让任意访问。

**python属性私有化语法：在类中属性的前面加上两个下划线**

代码：

**class** Student:  
 **def** \_\_init\_\_(self,name,age,grade):  
 self.\_\_name=name  
 self.\_\_age=age  
 self.\_\_grade=grade

说明：私有化属性，在属性前加\_\_(两个下划线)，在类中可以任意使用，但是类的外部不可访问。

那么如何让私有化属性在类外部访问呢？

==》通过对私有属性 **增加get和set方法**

get 用来输出私有属性的值 ==》在类外部通过调用get方法获取私有属性的值

set 用来修改私有属性的值 ==》在类外部通过调用set方法修改私有属性的值

代码：

**class** Student:  
 **def** \_\_init\_\_(self,name,age,grade):  
 *#私有化属性，在属性前加\_\_(两个下划线)，在类中可以任意使用，但是类的外部不可访问* self.\_\_name=name  
 self.\_\_age=age  
 self.\_\_grade=grade  
 **def** getName(self):  
 **return** self.\_\_name  
 **def** setName(self,name):  
 self.\_\_name=name  
  
 **def** getAge(self):  
 **return** self.\_\_age  
 **def** setAge(self,age):  
 self.\_\_age = age  
  
s1=Student(**"张三"**,18,99)  
s1.setName(**"李四"**)  
s1.setAge(20)  
print(s1.getName(),s1.getAge())

**总结：封装——私有成员与公有成员**

* **Python为了对类中成员实现访问保护机制，可以对成员进行私有化。在定义类的属性时，如果属性名以两个下划线“\_\_”开头则表示是私有属性，否则是公有属性。**
* **私有属性是为了数据封装和保密而设的属性，一般只能在类的成员方法（类的内部）中使用访问。通过对属性增加get和set方法，实现类外属性的获取和修改。**
* **公有属性是可以公开使用的，既可以在类的内部进行访问，也可以在类外部使用。**

**2.继承**

例子： 动物类 （姓名 睡觉 吼叫） ===》**父类**

猫类/狗类/猴子类 也有（姓名 睡觉 吼叫）==》**子类**

子类（猫类/狗类/猴子类） 满足使用（继承）父类（动物类）的特征

**说明：**

1. **类的继承：子类 是指在父类的基础上构造出新的类**
2. **子类可以使用父类的属性和方法（私有属性除外）**
   1. **儿子继承了父亲的遗产，那么儿子就拥有了父亲的所有遗产**
3. **儿子自己拥有的东西，则使用儿子自己的东西**
4. **优点： 增强代码的复用性**

**格式：**

**class 子类（父类）：**

代码：

*#父类-动物类***class** animal:  
 **def** \_\_init\_\_(self,name):  
 self.name=name  
 **def** sheep(self):  
 print(self.name,**"在睡觉"**)  
 **def** shout(self):  
 print(self.name,**"在大叫"**)  
  
*#子类1-猫类***class** cat(animal):  
 **pass***#子类2-狗类***class** dog(animal):  
 **pass***#子类3-猴子类***class** monkey(animal):  
 **pass**

c1=cat(**"波斯猫"**)  
print(c1.name)  
c1.sheep()  
  
d1=dog(**"泰迪"**)  
print(d1.name)  
d1.shout()  
  
m1=monkey(**"金丝猴"**)  
print(d1.name)  
m1.sheep()

**父类 强调共性==》吼叫**

**子类 强调个性 吼叫 猫—喵喵叫 狗—汪汪叫 猴子—吱吱和唧唧叫**

**那么怎么实现个性？**

**在子类中重新编写（重写：在子类中重新定义父类的方法 ==> 子类方法名和父类方法名是一致）**

*#子类1-猫类***class** cat(animal):  
 **def** \_\_init\_\_(self,name,color):

# 如果在子类中显式使用了构造方法，那么在子类构造方法中必须声明调用父类构造方法  
 # 如果在子类中没有编写构造方法，子类会默认调用父类的构造方法

animal.\_\_init\_\_(self,name)  
 self.color=color  
 **def** shout(self):  
 print(self.color,self.name,**"在喵喵叫"**)

c1=cat(**"波斯猫"**,**"白色"**)  
print(c1.name,c1.color)  
c1.shout()

**总结：**

**1.子类继承父类 语法：class 子类（父类）：**

**2.父类——突出共性，子类——突出个性**

**2.1 子类通过 重写 父类方法实现个性**

**1）重写时方法名要与父类方法名一致；**

**2）对象调用时候优先使用子类中的方法**

**2.2 在子类中定义新的方法——直接在子类增加一个方法即可**

**2.3 在子类中定义新的属性——通过调用父类构造函数**

**1）如果子类不编写构造函数，那么子类实例化时默认调用父类的构造函数**

**2）如果子类编写构造函数，那么必须调用父类的构造函数来给从父类继承的属性赋值**

**3.子类可以继承父类的公有成员，但是不能继承其私有成员。**

**4.如果需要在子类中调用父类的方法，可以通过“父类名.方法名(self,其他参数)”的方式来实现这一目的。**

**5.父类也叫基类，子类也叫派生类。**

**3.多态** 多种形态，比如对象波斯猫，即属于动物类，又属于猫类

使用内置函数 **isinstance（被判断的对象，类型类名）** 判断对象的类型

*#判断对象c1（波斯猫），是否时动物类，是否是猫类，判断对象d1（泰迪）是否时猫类，是否时狗类*

print(isinstance(c1,cat))  
print(isinstance(c1,animal))  
print(isinstance(d1,cat))  
print(isinstance(d1,dog))