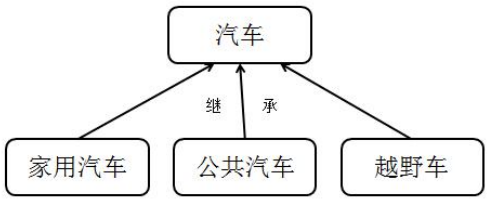
4. 继承

OOP 的三大特征是继承、封装和多态，继承可以解决编程中代码冗余的问题，是实现代码重用的重要手段。

（1）继承概述

理解继承可以跟现实中的继承一样，例如汽车，分成家用汽车、公共汽车、越野车等等，他们都具有基本的“汽车”特征，都具有品牌、颜色等属性，都具有运输功能，但是各自又都有各自的特点，家用汽车主要家庭出行功能，公共汽车用来公共运输乘客，越野车主要用来越野。这就是继承的现实意义。



在 OOP 中，“汽车”称为父类，“家用汽车、公共汽车、越野车”称为子类，子类继承父类已有的属性和方法，还可以自定义特有属性和方法。

继承的语法如下：

【语法】

```
class  子类名(父类名):
    #子类的变量和方法
```

（2）子类继承父类的属性和方法

子类可以直接继承父类的属性和方法，直接“拿来”使用。

示例：宠物类 Pet，子类 Cat，其中子类 Cat 没有定义自己的变量或方法

```
#父类 Car
class Car():
    def __init__(self,purpose = '运输'):
        self.purpose = purpose
    def show(self):
        print('这辆车的用途：',self.purpose)
#子类，继承父类 Car
class Home_Car(Car):
    pass
```

实例化子类的两种车：car_1 和 car_2，他们自动拥有了父类的 purpose 变量和 show() 方法。

```
#实例化两种车
car_1 = Home_Car()
car_2 = Home_Car('公共运输')
car_1.show()
car_2.show()
```

输出结果为：

这辆车的用途： 运输

这辆车的用途： 公共运输

（3）构造方法的继承

如果子类本身没有定义自己的构造方法__init__()，就自动继承父类的构造方法。

如果子类本身定义了自己的构造方法__init__()，那么就不会自动调用父类的构造方法。

试一试，如果把子类 Home_Car 修改为如下，其他代码不变化，会输出什么？

```
#子类，继承父类 Car
class Home_Car(Car):
    def __init__(self,seat = 5):
        self.seat = seat
```

输出结果：

AttributeError: 'Home_Car' object has no attribute 'purpose'

（4）子类的方法

子类除了可以继承父类的方法外，当然还可以拥有自己的方法，例如子类公共汽车 Bus 类，继承自 Car，Bus 类拥有自己的方法载客量 busload()方法。

```
#子类 Bus
class Bus(Car):
    def busload(self,passenger):
        print('公共汽车的载客量%d'%passenger)

bus_1 =Bus()
bus_1.busload(45)
```

实例化对象 bus_1，调用自己的方法 busload()，输出结果为：

公共汽车的载客量 45

（5）多继承

Python 支持多继承，也就是子类可以有多个父类。多继承的类定义语法如下:

【语法】

class 子类名(父类 1,父类 2):

#子类的变量和方法

在多继承的情况下，子类有多个父类，如果子类没有自己的构造方法，则子类会按照继承列表中的父类顺序，找到第一个定义了构造方法的父类，并继承它的构造方法。