

Python语言基础与应用

高级特性 / 面向对象：类的继承

陈斌 北京大学 gischen@pku.edu.cn



面向对象：类的继承

- › 类的继承机制
- › 子类与父类
- › 关于self

类的继承机制

› 继承(inheritance)

如果一个类别A**继承**自另一个类别B，就把继承者A称为**子类**，被继承的类B称为**父类**、**基类**或**超类**

› 代码复用

利用继承可以从已有类中衍生出新的类，添加或修改部分功能

新类具有旧类中的各种属性和方法，而不需要进行任何复制

类的继承机制

```
45 class Car:
46     def __init__(self, name):
47         self.name = name
48         self.remain_mile = 0
49
50     def fill_fuel(self, miles): # 加燃料里程
51         self.remain_mile = miles
52
53     def run(self, miles): # 跑miles英里
54         print(self.name, end=': ')
55         if self.remain_mile >= miles:
56             self.remain_mile -= miles
57             print("run %d miles!" % (miles,))
58         else:
59             print("fuel out!")
60
61
62 class GasCar(Car):
63     def fill_fuel(self, gas): # 加汽油gas升
64         self.remain_mile = gas * 6.0 # 每升跑6英里
65
66
67 class ElecCar(Car):
68     def fill_fuel(self, power): # 充电power度
69         self.remain_mile = power * 3.0 # 每度电3英里
```

类的继承机制

```
71  gcar=GasCar("BMW")  
72  gcar.fill_fuel(50.0)  
73  gcar.run(200.0)
```

```
75  ecar=ElecCar("Tesla")  
76  ecar.fill_fuel(60.0)  
77  ecar.run(200.0)
```

```
BMW: run 200 miles!  
Tesla: fuel out!
```

子类与父类

› 定义

如果两个类具有“一般-特殊”的逻辑关系，那么特殊类就可以作为一般类的“子类”来定义，从“父类”继承属性和方法

```
class <子类名>(<父类名>):  
    def <重定义方法>(self,...):
```

子类与父类

› 覆盖(Override)

子类对象可以调用父类方法，除非这个方法在子类中重新定义了

如果子类同名方法覆盖了父类的方法，仍然还可以调用父类的方法

› 子类还可以添加父类中没有的方法和属性

```
class GasCar(Car):  
    def __init__(self, name, capacity): # 名称和排量  
        super().__init__(name) # 父类初始化方法，只有名称  
        self.capacity = capacity # 增加了排量属性
```

关于self

› 在类定义中，所有方法的首个参数一般都是self

› self的作用

在类内部，实例化过程中传入的所有数据都赋给这个变量

关于self

› self实际上代表对象实例

⟨对象⟩.⟨方法⟩(⟨参数⟩)

等价于:

⟨类⟩.⟨方法⟩(⟨对象⟩, ⟨参数⟩)

这里的对象就是self

› 如下图line81和82

```
79  gcar = GasCar("BMW")
80  gcar.fill_fuel(50.0)
81  gcar.run(200.0)
82  GasCar.run(gcar, 200.0)
```