类

例：程序class\_test/class\_test1.py

class Dog:  
 def \_\_init\_\_(self, name, age):  
 *"""初始化"""* # 类中定义了两个属性，类似C++中的成员变量  
 # name是public，前面加\_\_是private  
 # 类中所有的属性必须提供初始值  
 self.name = name  
 self.\_\_age = age  
  
 # 函数第一个参数永远是self  
 def sit(self):  
 print(self.name.title() + " is now sitting")  
  
  
my\_dog = Dog("willie", 6)  
your\_dog = Dog("Tom", 7)  
my\_dog.sit()  
your\_dog.sit()  
print(my\_dog.name)  
# print(my\_dog.\_\_age) # Error  
  
# 可以给类的实例绑定属性  
# 但只有该实例可用  
my\_dog.weight = 9  
print(my\_dog.weight)  
print()  
# print(your\_dog.weight) # Error，没有weight属性  
  
  
class Car:  
 def \_\_init\_\_(self, make, model, year):  
 self.make = make  
 self.model = model  
 self.year = year  
 self.odometer\_reading = 0 # 默认值  
  
 def get\_descriptive\_name(self):  
 full = str(self.year) + ' ' + self.make + ' ' + self.model  
 return full.title()  
  
 def read\_odometer(self):  
 print("This car has " + str(self.odometer\_reading) + " miles on it.")  
  
 def update\_odometer(self, mileage):  
 if mileage > self.odometer\_reading:  
 self.odometer\_reading = mileage  
 else:  
 print("You can`t roll back an odometer!")  
  
  
my\_new\_car = Car('audi', 'a4', 2016)  
print(my\_new\_car.get\_descriptive\_name())  
my\_new\_car.update\_odometer(23)  
my\_new\_car.read\_odometer()

输出为：

Willie is now sitting

Tom is now sitting

willie

9

2016 Audi A4

This car has 23 miles on it.

类的继承：

例：程序class\_test/class\_test2.py

# 基类  
class Car:  
 def \_\_init\_\_(self, make, model, year):  
 self.make = make  
 self.model = model  
 self.year = year  
 self.odometer\_reading = 0 # 默认值  
  
 def get\_descriptive\_name(self):  
 full = str(self.year) + ' ' + self.make + ' ' + self.model  
 return full.title()  
  
 def read\_odometer(self):  
 print("This car has " + str(self.odometer\_reading) + " miles on it.")  
  
 def update\_odometer(self, mileage):  
 if mileage > self.odometer\_reading:  
 self.odometer\_reading = mileage  
 else:  
 print("You can`t roll back an odometer!")  
  
  
# 子类  
# 基类\_\_init\_\_的参数，子类  
# 最好也有，否则有可能出现错误  
class ElectricCar(Car):  
 def \_\_init\_\_(self, make, model, year):  
 super().\_\_init\_\_(make, model, year)  
 self.battery\_size = 70  
 print("init electric car")  
  
 def describe\_battery(self):  
 print("This car has a" + str(self.battery\_size) + "-kWh battery")  
  
  
my\_tesla = ElectricCar('tesla', 'model s', 2016)  
my\_tesla.read\_odometer()  
my\_tesla.describe\_battery()

输出为：

init electric car

This car has 0 miles on it.

This car has a70-kWh battery

重写父类的方法：

例：程序class\_test/class\_test3.py

# 父类  
class Car:  
 def fill\_gas\_tank(self):  
 print("This car has fill gas")  
  
  
# 子类重写了父类的方法  
# 如果通过子类对象调用方法  
# 就会调用子类的方法  
class ElectricCar(Car):  
 def fill\_gas\_tank(self):  
 print("This car doesn't need gas")  
  
  
my\_car = Car()  
my\_car.fill\_gas\_tank()  
  
my\_tesla = ElectricCar()  
my\_tesla.fill\_gas\_tank()

输出为：

This car has fill gas

This car doesn't need gas

类的实例作为属性：

例：程序class\_test/class\_test4.py

# 基类  
class Car:  
 def \_\_init\_\_(self, make, model, year):  
 self.make = make  
 self.model = model  
 self.year = year  
 self.odometer\_reading = 0 # 默认值  
  
 def get\_descriptive\_name(self):  
 full = str(self.year) + ' ' + self.make + ' ' + self.model  
 return full.title()  
  
 def read\_odometer(self):  
 print("This car has " + str(self.odometer\_reading) + " miles on it.")  
  
 def update\_odometer(self, mileage):  
 if mileage > self.odometer\_reading:  
 self.odometer\_reading = mileage  
 else:  
 print("You can`t roll back an odometer!")  
  
  
class Battery:  
 def \_\_init\_\_(self, battery\_size=70):  
 self.battery\_size = battery\_size  
  
 def describe\_battery(self):  
 print("This car has a " + str(self.battery\_size) + "-kWh battery.")  
  
 def get\_range(self):  
 if self.battery\_size == 70:  
 battery\_range = 240  
 elif self.battery\_size == 85:  
 battery\_range = 280  
  
 message = "This car can go approximately " + str(battery\_range)  
 message += " miles on a full charge"  
 print(message)  
  
  
class ElectricCar(Car):  
 def \_\_init\_\_(self, make, model, year):  
 super().\_\_init\_\_(make, model, year)  
 self.battery = Battery() # 类的实例做为属性  
  
  
my\_tesla = ElectricCar('tesla', 'model s', 2016)  
my\_tesla.battery.describe\_battery()  
my\_tesla.battery.get\_range()

输出为：

This car has a 70-kWh battery.

This car can go approximately 240 miles on a full charge