ch9-类

定义

```
class Dog():
    """一次模拟小狗的简单尝试"""
    def __init__(self, name, age):
    """初始化属性name和age"""
        self.name = name
        self.age = age
    def sit(self):
    """模拟小狗被命令时蹲下"""
        print(self.name.title() + " is now sitting.")
    def roll_over(self):
    """模拟小狗被命令时打滚"""
        print(self.name.title() + " rolled over!")
```

方法(method):类中的函数(function)=方法(method)
init()=构造函数
self=this
类中的方法def xxx(self):

python2.7中创建类

```
class ClassName(object):
```

继承

```
class Car():
    """一次模拟汽车的简单尝试"""

    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
        self.odometer_reading = 0

    def get_descriptive_name(self):
        long_name = str(self.year) + ' ' + self.make + ' ' + self.model
        return long_name.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!")

    def increment_odometer(self, miles):
```

```
        self.odometer_reading += miles


class ElectricCar(Car):
    """电动汽车的独特之处"""

    def __init__(self, make, model, year):
        """初始化父类的属性"""
        super().__init__(make, model, year)


my_tesla = ElectricCar('tesla', 'model s', 2016)
print(my_tesla.get_descriptive_name())
```

核心在于

```
class ElectricCar(Car):
    """电动汽车的独特之处"""

    def __init__(self, make, model, year):
        """初始化父类的属性"""
        super().__init__(make, model, year)
```

python2.7**中的继承**

```
class Car(object):
    def __init__(self, make, model, year):
        --snip - -


class ElectricCar(Car):
    def __init__(self, make, model, year):
        super(ElectricCar, self).__init__(make, model, year)
        --snip - -
```

函数super() 需要两个实参：子类名和对象self。为帮助Python将父类和子类关联起来，这些实参必不可少。另外，在Python 2.7中使用继承时，务必在定义父类时在括号内指定object 。

使用代码模拟实物时，你可能会发现自己给类添加的细节越来越多：属性和方法清单以及文件都越来越长。在这种情况下，可能需要将类的一部分作为一个独立的类提取出来。
你可以将大型类拆分成多个协同工作的小类。

OrderedDict:有序字典
random模块 randint()随机数