Python编程：从入门到实践9

创建和使用类

1. 通过首字母大写的形式告诉Python创建一个类。
2. 方法__init__()是每一个类的默认方法，其中的形参self必不可少.
3. self.name 获取存储在形参name中的值

```
class Dog():
    def __init__(self,name,age):
        self.name = name
        self.age = age
    def sit(self):
        print(self.name.title() + ' is now siting.')
    def roll_over(self):
        print(self.name.title() + ' rolled over')

my_dog = Dog('willie', 6)
print('My dog`s name is ' + my_dog.name.title() + '.')
my_dog.sit()
my_dog.roll_over()
```

使用类和实例

1. 修改实例中有默认值的形参，可以通过句点表示法直接访问的方式修改
2. 也可以直接编写一个修改属性值的方法（update_odometer()）

```
class Car():
    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 not roll back an odometer!')


my_new_car = Car('audi', 'a4', 2016)
print(my_new_car.get_descriptive_name())
my_new_car.read_odometer()
my_new_car.odometer_reading = 100
my_new_car.read_odometer()
my_new_car.update_odometer(300)
my_new_car.read_odometer()
my_new_car.update_odometer(200)
```

继承

1. 当要编写的类是另一个现成类的特殊版本，可以直接继承现成的类；原有的类称为父类(superclass)，新的类称为子类。
2. 定义子类时，括号内包含父类。同时使用super()方法。
3. 子类也可以重写父类已有的但是不适合子类的方法。
4. 实例也可以作为属性添加。（self.battery=Battery()）

```
class ElectricCar(Car):
    def __init__(self, make, model, year):
        super().__init__(make,model,year)
        self.battery_size = 70
    def describe_battery(self):
        print('This car has a ' + str(self.battery_size) + '-KWH battery.')
my_tesla = ElectricCar('tesla', 'model s', 2016)
my_tesla.describe_battery()


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.')
class ElectricCar(Car):
    def __init__(self, make, model, year):
        super().__init__(make,model,year)
        self.battery = Battery()
```

```python
my_tesla = ElectricCar('tesla', 'model s', 2016)
my_tesla.battery.describe_battery()
```