第14课 音乐人与Rapper——类

1、类的继承和定制

1-1、类的继承（财产被继承了）

类的继承，即让子类拥有了父类拥有的所有属性和方法

1-2、类的定制（“我就是我，是颜色不一样的烟火”）

子类也可以在继承的基础上进行个性化的定制，包括：

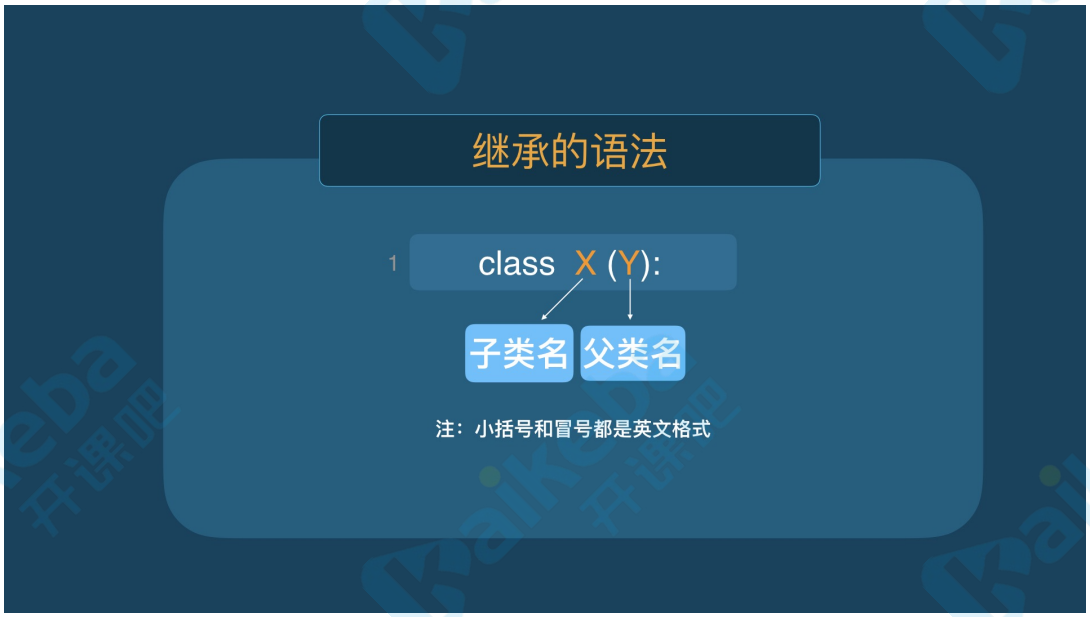
- 创建新属性、新方法
- 修改继承到的属性或方法

2、继承的编写规则

2-1、继承的基础语法

```
class X():
    name='小K'
class Y(X):
    pass
```

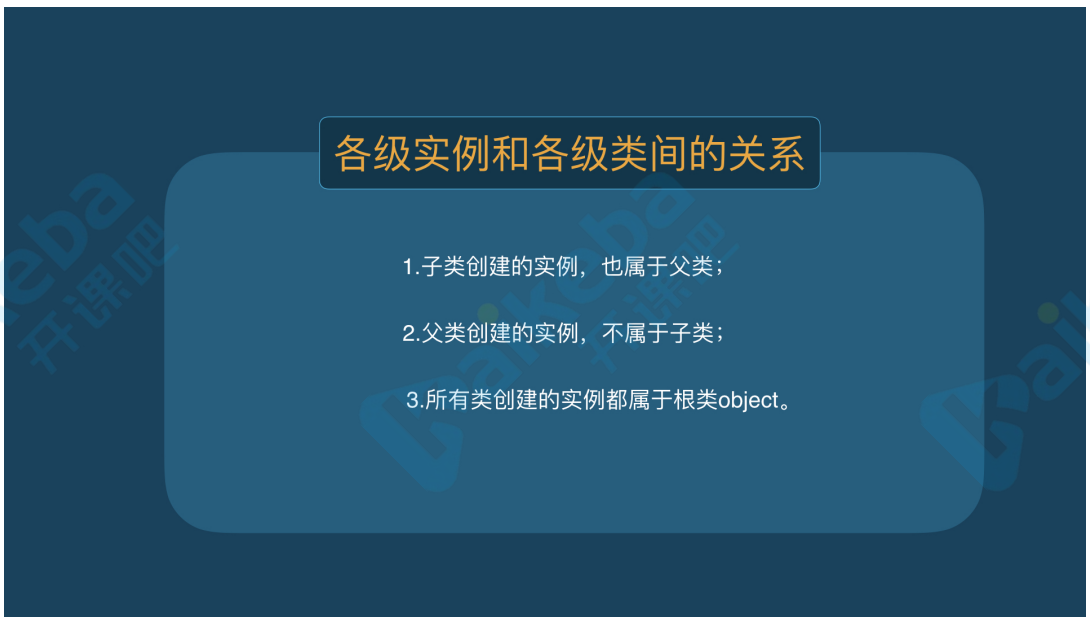
以上代码的第 3 行， class Y(X) 就是我们的继承语句，其中 Y是子类， X 代表着我们的父类。



我们的子类继承了父类，那就是说，父类的所有东西，我们子类都可以拿来用，下面我们来看一下例子：

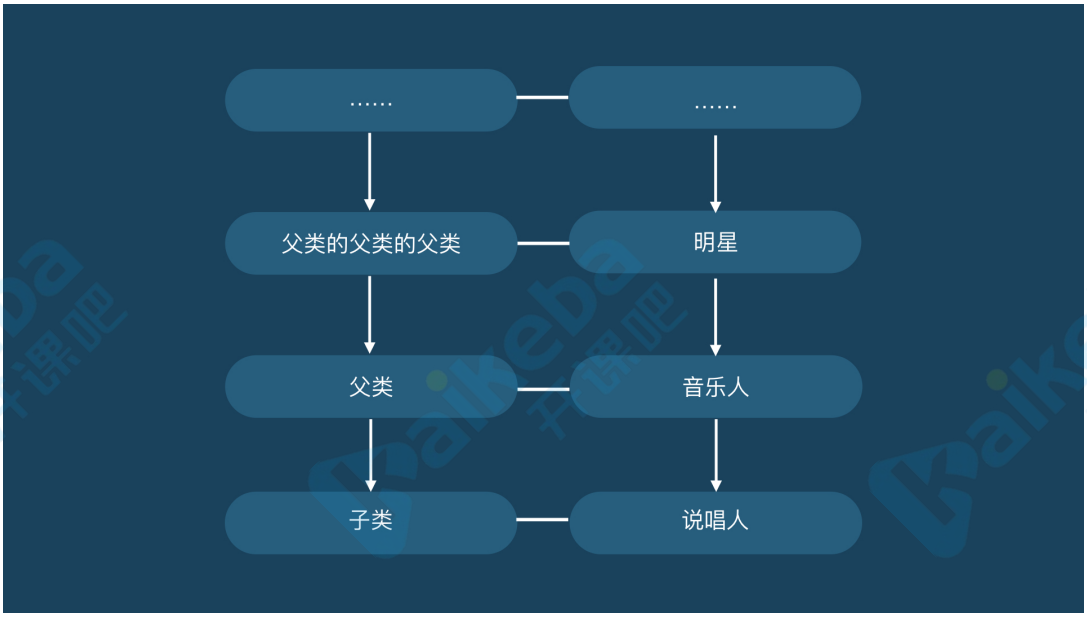
```
class Car:
    wheel = 4
    def run(self):
        print('有%d个轮子,可以飞速的行驶'%self.wheel)
class BMW(Car):
    pass
BMW320 = BMW()
print(BMW320.wheel)
```

我们父类 Car 定义了变量wheel 并赋值4，而子类 BMW 继承了 Car但是什么都没操作（pass），我们实例化函数 BMW 后，实例化对象 BMW320可以调用我们父类 Car中的 wheel 属性。



2-2、类的多层继承

继承不仅可以发生在两个层级之间（即父类-子类），还可以有父类的父类、父类的父类的父类……




```
class Star:
    glasses = "墨镜"
#音乐人继承了明星
class Musician(Star):
    loveMusic = True
# Rapper继承了音乐人
class Rapper(Musician):
    pass
csunYuk = Rapper()
print(csunYuk.glasses)
print(csunYuk.loveMusic)
```

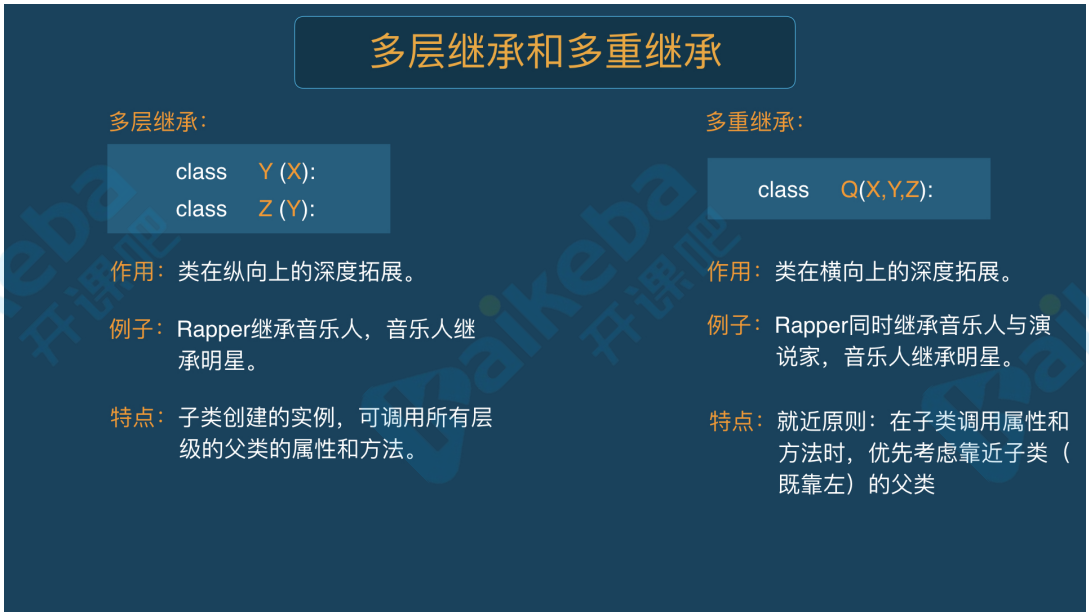
上面例子中，类 Musician继承了类 Star，类 Rapper 再继承了类 Musician ，即类 Rapper继承了前面两个的所有属性，所以可以正常打印出我们的“墨镜”,True。

2-3、类的多重继承

一个类，可以同时继承多个类，语法为“class A(B,C,D):”。和子类更相关的父类会放在更左侧，子类创建的实例在调用属性和方法时，会先在左侧的父类中找，找不到才会去右侧的父类找。（可理解为“就近原则”）

```
class panpan():
    teacher='潘潘'
class momo():
    teacher='墨墨'
class fudao():
    teacher='辅导'
class main(panpan,momo,fudao):
    pass
team = main()
print('老师'+team.teacher+'最棒')
```

以上代码中，类 main 同时继承了我们的类 panpan,momo,fudao，而根据我们的“就近原则”，我们在调用属性 teacher 的时候，类panpan 离我们的子类最近且有定义该类，所以打印出来的结果为“老师潘潘最棒”。



3、定制的编写规则

3-1、新增代码

在子类下新建属性或方法，让子类可以用上父类所没有的属性或方法。这种操作，属于定制中的一种：新增代码

```
#音乐人
class Musician():
    loveMusic = True
    def intr(self):
        print("我喜欢音乐")
        print("我来自音乐界")
    def sing(self):
        print("我在唱歌")
# Rapper继承音乐人
class Rapper(Musician): #类的继承
    garrulous = True     #类的定制 ，  增加属性
    def composition(self):  #类的定制，  增加方法
        print("我可以创作简单的歌词")
csunYuk = Rapper()
print(csunYuk.loveMusic)
print(csunYuk.garrulous)
csunYuk.composition()
#True
#True
我可以创作简单的歌词
```

以上代码中，类 Rapper 继承了类 Musician，但是它增加了属性garrulous，增加了方法composition，这就是类的定制。

我们实例化了类 Rapper 后，可以调用父类的属性和方法外，也可以给子类新增子类的属性和方法。

3-2、重写代码

①、重写代码，是在子类中，对父类代码的方法修改：

```
#音乐人
class Musician():
    loveMusic = True
    def intr(self):
        print("我喜欢音乐")
        print("我来自音乐界")
    def sing(self):
        print("我在唱歌")
# Rapper继承音乐人
class Rapper(Musician): #类的继承
    def sing(self):  #类的定制，  更改方法
        print("我以说的形式进行歌唱")
csunYuk = Rapper()
csunYuk.sing()
```

以上代码中，我们子类 Rapper 继承了父类 Musician，并且重写了方法 sing，即我们用类 Rapper 的实例化对象去调用方法 sing 的时候，会打印出“我以说的形式进行歌唱”。

②、在子类中，对父类代码的实例化属性修改：

```
class Musician():
    def __init__(self,name='周杰伦'):
        self.name=name
    def Name(self):
        print('我的名字是'+self.name)
class Rapper(Musician):
    def __init__(self,name='潘玮柏'):
        Musician.__init__(self,name)
b=Rapper()
b.Name()
#我的名字是潘玮柏
```

以上代码中，我们子类 Rapper 继承了父类 Musician，调用父类的初始化函数后，重新给参数 name 赋值，再给这个 name 给实例化对象 name，具体传参路线如下：

```
class Musician():
    def __init__(self,name='周杰伦'):
        self.name=name
    def Name(self):
        print('我的名字是'+self.name)
class Rapper(Musician):
    def __init__(self,name='潘玮柏'):
        Musician.__init__(self,name)
b=Rapper()
b.Name()
#我的名字是潘玮柏
```