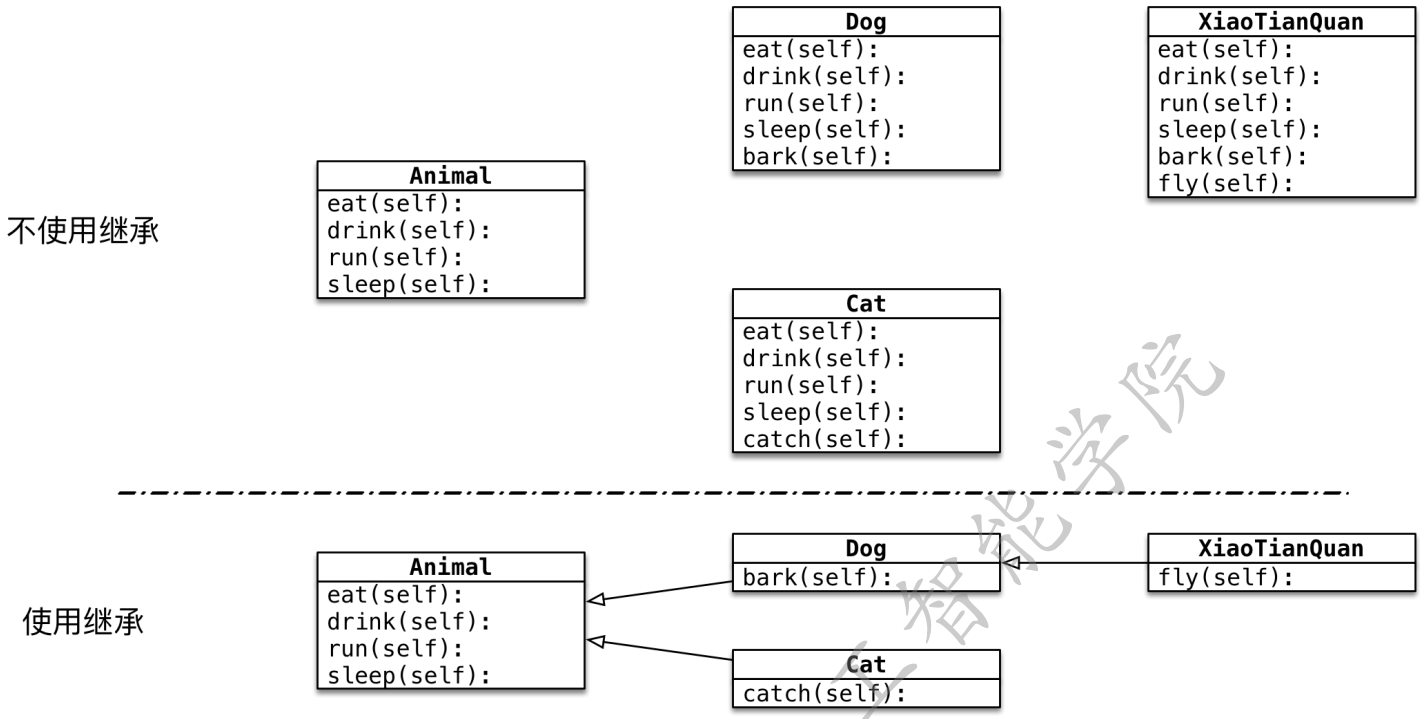
在Python中，继承可以分为单继承、多继承和多层继承。

单继承：子类只继承一个父类

继承概念：子类用于父类的所有的方法和属性。



继承语法:

```
class 类名(父类名):
    pass
```

- 子类继承自父类，可以享受父类中已经封装好的方法，不需要再次定义
- 子类中应该根据职责，封装子类特有的属性和方法。

继承的传递性

Dog类继承自Animal,XiaoTianQuan又继承自Dog类，那么XiaoTianQuan类就具有了Animal类里的所有属性和方法。

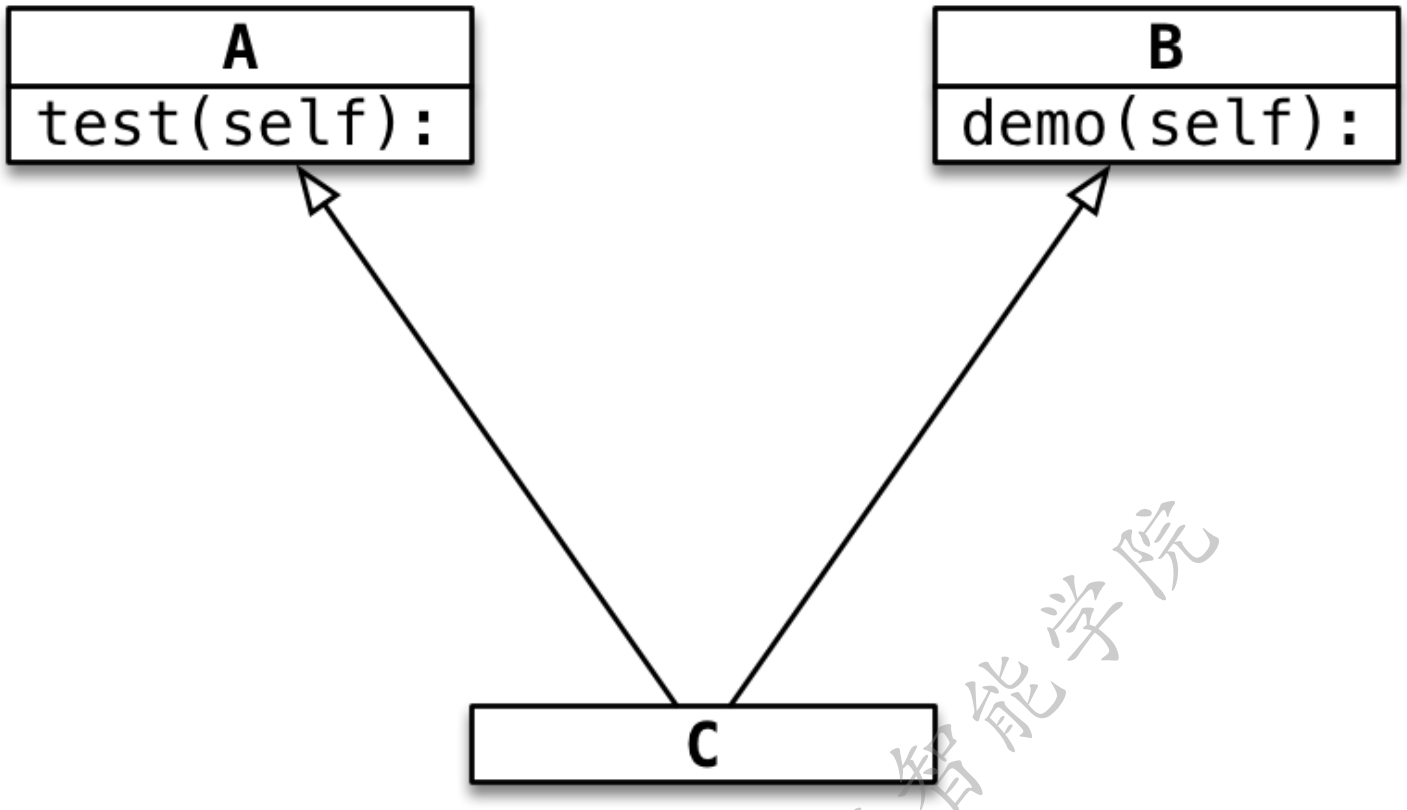
子类拥有父类以及父类的父类中封装的所有属性和方法。

思考:

> XiaoTianQuan能否调用Animal的run()方法？ XiaoTianQUan能够调用Cat里的方法?

多继承

子类可以拥有多个父类，并且具有所有父类的属性和方法。



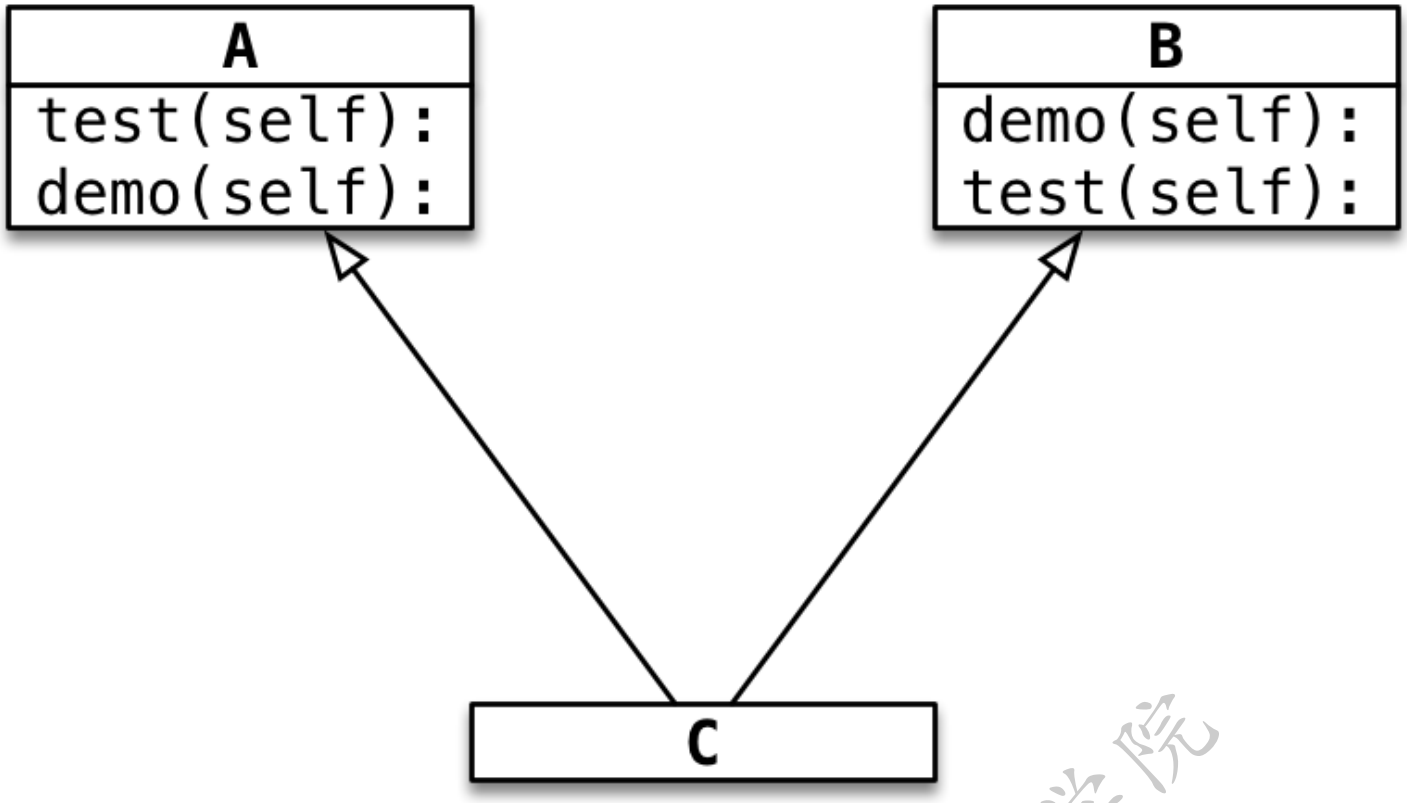
语法格式：

```
class 子类名(父类名1,父类名2...)
    pass
```

多继承的使用注意事项

思考:

> 如果不同的父类中存在同名的方法，子类对象在调用方法时，会调用哪个父类的方法？ 说明：开发中，应该尽量避免这种容易产生混淆的情况。如果多个父类之间存在同名的属性后者方法，应该尽量避免使用多继承。



Python中的MRO

- Python中针对类提供了一个内置属性 `__mro__` 可以用来查看方法的搜索顺序。
- MRO 是 `method resolution order` 的简称，主要用于在多继承时判断方法属性的调用顺序。

```
print(C.__mro__)
```

输出结果:

```
(<class '__main__.C'>, <class '__main__.A'>, <class '__main__.B'>, <class 'object'
>)
```

- 在调用方法时，按照 `__mro__` 的输出结果从左至右的顺序查找。
- 如果再当前类中找到方法，就直接执行，不再向下搜索。
- 如果没有找到，就顺序查找下一个类中是否有对应的方法，如果找到，就直接执行，不再继续向下搜索。
- 如果找到了最后一个类，依然没有找到方法，程序就会报错。

新式类和旧式（经典）类

`object` 是Python中所有对象的基类，提供了一些内置的属性和方法，可以时候用 `dir` 函数查看。

- 新式类：以 `object` 为基类的类，**推荐使用**

- 经典类：不以object为基类的类，**不推荐使用**
- 在 Python3.x 以后定义类时，如果没有指定父类，这个类会默认继承自 object,所以，python3.x版本定义的类都是新式类。
- 在Python2.x中定义类时，如果没有指定父类，则不会继承自object.

为了保证代码在Python2.x和Python3.x中都能够运行，在定义类时，如果一个类没有父类，建议统一继承自'object'

```
class 类名(object):
    pass
```