面向对象梳理和案例分析

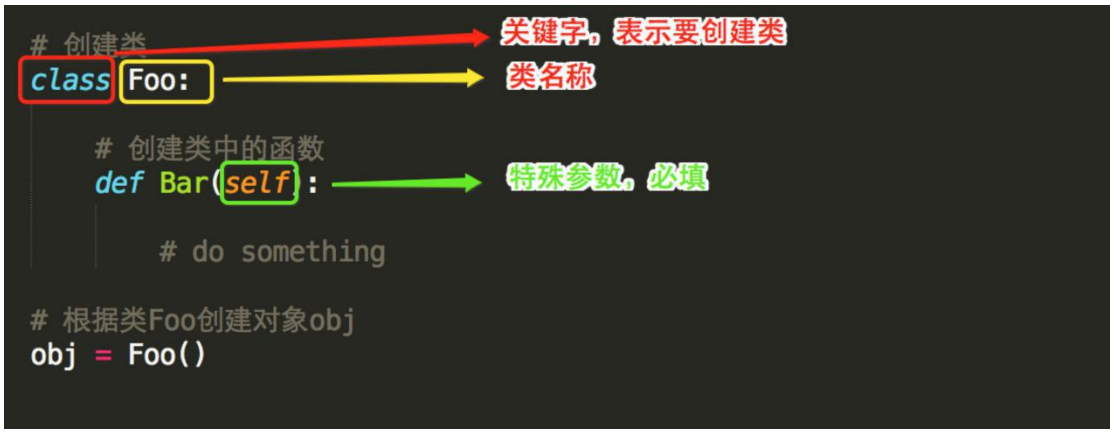
面向过程：根据业务逻辑从上到下写垒代码

面向对象：对函数进行分类和封装，

面向对象编程是一种编程方式，此编程方式的落地需要使用“类”和“对象”来实现，所以，面向对象编程其实就是对“类”和“对象”的使用。

类就是一个模板，模板里可以包含多个函数，函数里实现一些功能

对象则是根据模板创建的实例，通过实例对象可以执行类中的函数



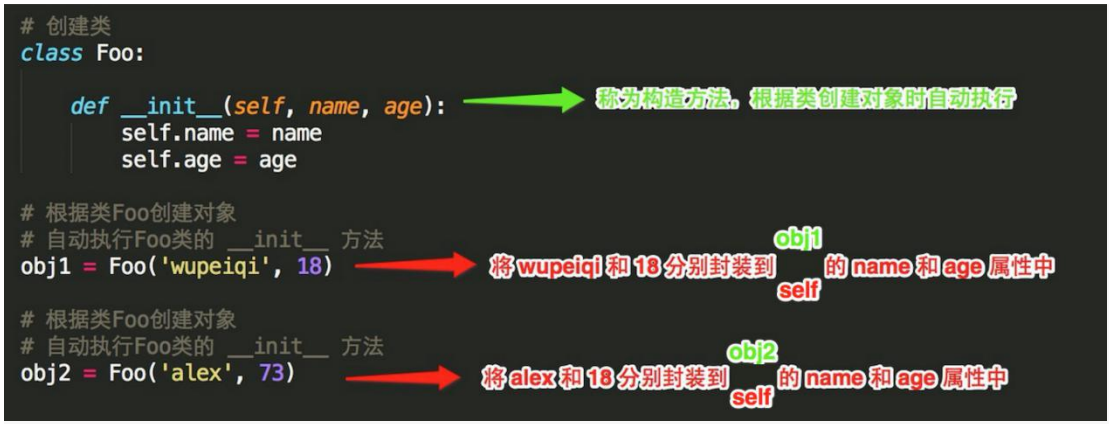

```python
# 创建类
class Foo:

    # 创建类中的函数
    def Bar(self):

        # do something

# 根据类Foo创建对象obj
obj = Foo()
```

面向对象三大特性

面向对象的三大特性是指：封装、继承和多态。

一、封装

封装，顾名思义就是将内容封装到某个地方，以后再去调用被封装在某处的内容。



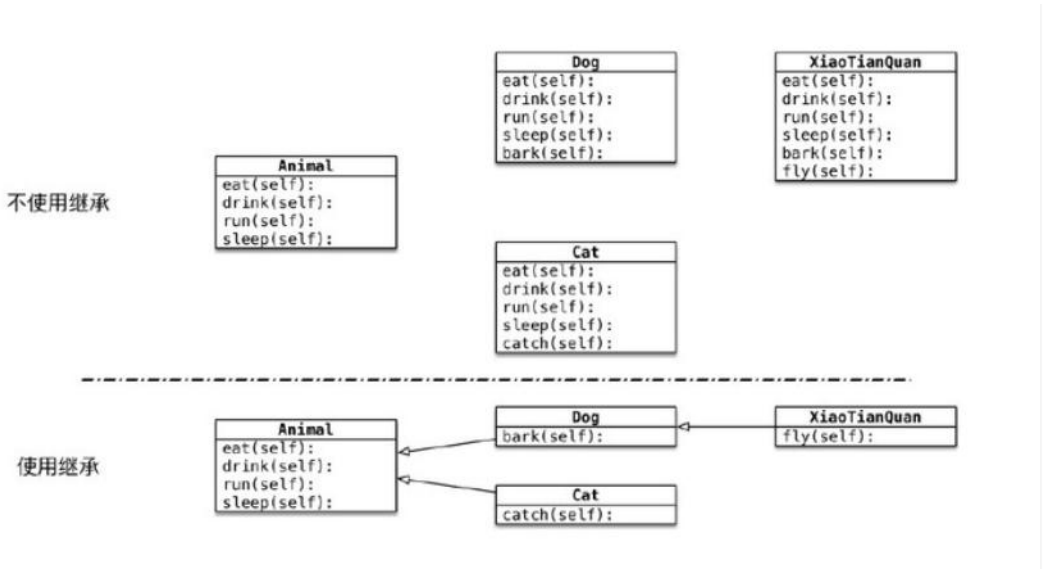

```python
# 创建类
class Foo:

    def __init__(self, name, age):
        self.name = name
        self.age = age

# 根据类Foo创建对象
# 自动执行Foo类的 __init__ 方法
obj1 = Foo('wupeiqi', 18)

# 根据类Foo创建对象
# 自动执行Foo类的 __init__ 方法
obj2 = Foo('alex', 73)
```

二，继承

继承的概念：子类拥有父类的所有方法和属性。子类继承自父类，可以直接享受父类中已经封装好的方法，子类中应该根据职责，封装子类特有的属性和方法。

解释：假如我需要定义几个类，而类与类之间有一些公共的属性和方法，这时我就可以把相同的属性和方法作为基类的成员，而特殊的方法及属性则在本类中定义。这样子类只需要继承父类，子类就可以访问到父类的属性和方法了，它提高了代码的可扩展性和重用行。

如：猫，狗 都属于动物，它们行为相似性高，都会叫，会吃，会喝，会拉，会撒娇。



`Dog` 类是 `Animal` 类的子类，`Animal` 类是 `Dog` 类的父类，`Dog` 类从 `Animal` 类继承

提问

哮天犬 能够调用 `Cat` 类中定义的 `catch` 方法吗？

继承的重写：
子类可以重新定义父类中的方法，这样就会覆盖父类中的方法，也称为重写。

```
class Person:
    def __init__(self,name,age):
        self.name=name
        self.__age=age
    def say_age(self):
        print("我的年龄是",self.__age)
    def say_introduce(self):
        print("我的名字是{}".format(self.name))
class Student(Person):
    def __init__(self,name,age,score):
        Person.__init__(self,name,age)
        self.score=score
    def say_introduce(self):
        '''重写了父类的方法'''
        print("报告老师，我的名字是{}".format(self.name))
s=Student("张无忌",18,100)
s.say_age()
s.say_introduce()

#打印结果
我的年龄是 18
报告老师，我的名字是张无忌
```

三，多态

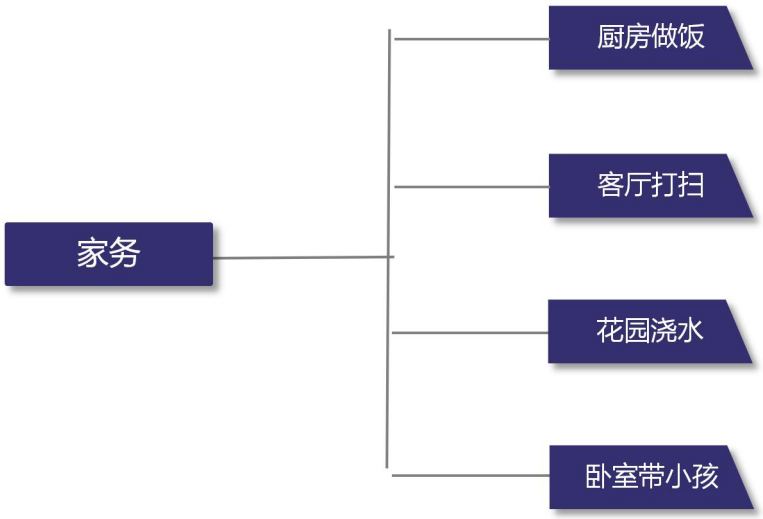
多态从字面意思上看就是多种形态，比如人有黑种人，黄种人，白种人等等，这就是一类事物的不同形态，在我们 python 的面向对象里就是不同的对象在接收相同方法或者函数时会产生不同的行为，也就是说，每个对象可以用自己的方式去响应共同的函数，不同的方式实现不同的结果，多态的使用一般是和类的继承绑定在一起，实现一类事物中的不同形态。

```
class Man:
    def eat(self):
        print("饿了，吃饭了!")
class Chinese(Man):
    def eat(self):
        print("中国人用筷子吃饭")

class English(Man):
    def eat(self):
        print("英国人用刀叉吃饭")

class Indian(Man):
    def eat(self):
        print("印度人用右手吃饭")


def manEat(m):
    if isinstance(m,Man):
        m.eat()
    else:
        print("不能吃饭")


#子类Chinese实例化的
manEat(Chinese())
#子类English实例化的
manEat(English())

#Person这个实例化对象,父类实例化的
Person=Man()
manEat(Person)
#运行结果
```

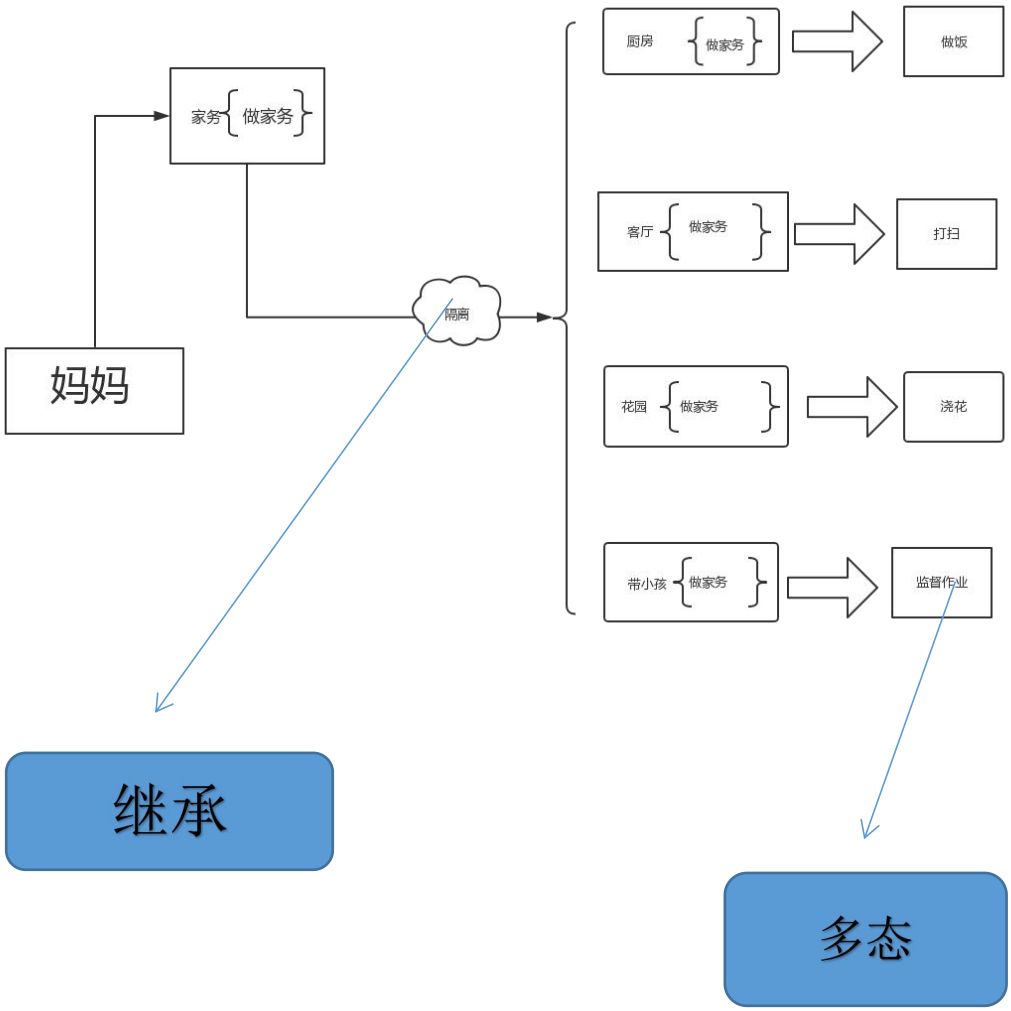
案例分析

背景：李妈妈每天在家都很辛苦，从早上起来，先要做早饭，之后要打扫卫生，然后在给植物浇水，最后还要接小孩放学，事情又多又杂，每天这么多事，导致妈妈心情不好，心情不好做的饭就不好吃，也不愿打扫卫生，最后可能孩子都不要了。

过程：邻居陈大爷见到这个情况就说，我的儿子小陈也许能帮李妈妈解决目前的困境。哦！原来小陈最近刚在达内学习了面向对象的知识，让我们看看小陈是怎么帮助李妈妈解决这些问题的。



封装



案例中体现的六大原则：
开闭原则：家务的增加或者减少，妈妈都不用改变
单一原则：将各项家务依据特点，进行分类，互相独立
依赖倒置：妈妈依赖的是“家务”的“做家务”功能，不直接调用厨房，花园等功能
组合复用：妈妈做家务，是通过家务这个父类，而不是直接调用各项子类
里式替换：可以在子类对做家务进行重写
迪米特法则：各个做家务的子类互相独立，功能重合性低

结果：小陈帮助李妈妈对做家务这项工作进行重新分析和梳理，让李妈妈从繁杂的家务中脱离出来，只需要对“家务”这一项活动负责，李妈妈的心态也调整过来了，不觉得自己每天事情又多又杂，每天都是元气满满的一天啊！