面向过程 VS 面向对象

面向过程的程序设计的核心是过程（流水线式思维），过程即解决问题的步骤，面向过程的设计就好比精心设计好一条流水线，考虑周全什么时候处理什么东西。

优点是：极大的降低了写程序的复杂度，只需要顺着要执行的步骤，堆叠代码即可。

缺点是：一套流水线或者流程就是用来解决一个问题，代码牵一发而动全身。

应用场景：一旦完成基本很少改变的场景

面向对象的程序设计

优点是：解决了程序的扩展性。对某一个对象单独修改，会立刻反映到整个体系中，如对游戏中一个人物参数的特征和技能修改都很容易。

缺点：可控性差，无法向面向过程的程序设计流水线式的可以很精准的预测问题的处理流程与结果，面向对象的程序一旦开始就由对象之间的交互解决问题，即便是上帝也无法预测最终结果。于是我们经常看到一个游戏人某一参数的修改极有可能导致阴霸的技能出现，一刀砍死3个人，这个游戏就失去平衡。

应用场景：需求经常变化的软件，一般需求的变化都集中在用户层，互联网应用，企业内部软件，游戏等都是面向对象的程序设计大显身手的好地方。在python 中面向对象的程序设计并不是全部。

面向对象编程可以使程序的维护和扩展变得更简单，并且可以大大提高程序开发效率 ，另外，基于面向对象的程序可以使它人更加容易理解你的代码逻辑，从而使团队开发变得更从容。

封装

【封装】

隐藏对象的属性和实现细节，仅对外提供公共访问方式。

【好处】

1. 将变化隔离；
2. 便于使用；
3. 提高复用性；
4. 提高安全性；

【封装原则】

1. 将不需要对外提供的内容都隐藏起来；
2. 把属性都隐藏，提供公共方法对其访问。

封装与扩展性

封装在于明确区分内外，使得类实现者可以修改封装内的东西而不影响外部调用者的代码；而外部使用用者只知道一个接口(函数)，只要接口（函数）名、参数不变，使用者的代码永远无需改变。这就提供一个良好的合作基础——或者说，只要接口这个基础约定不变，则代码改变不足为虑。

继承与抽象（先抽象再继承）

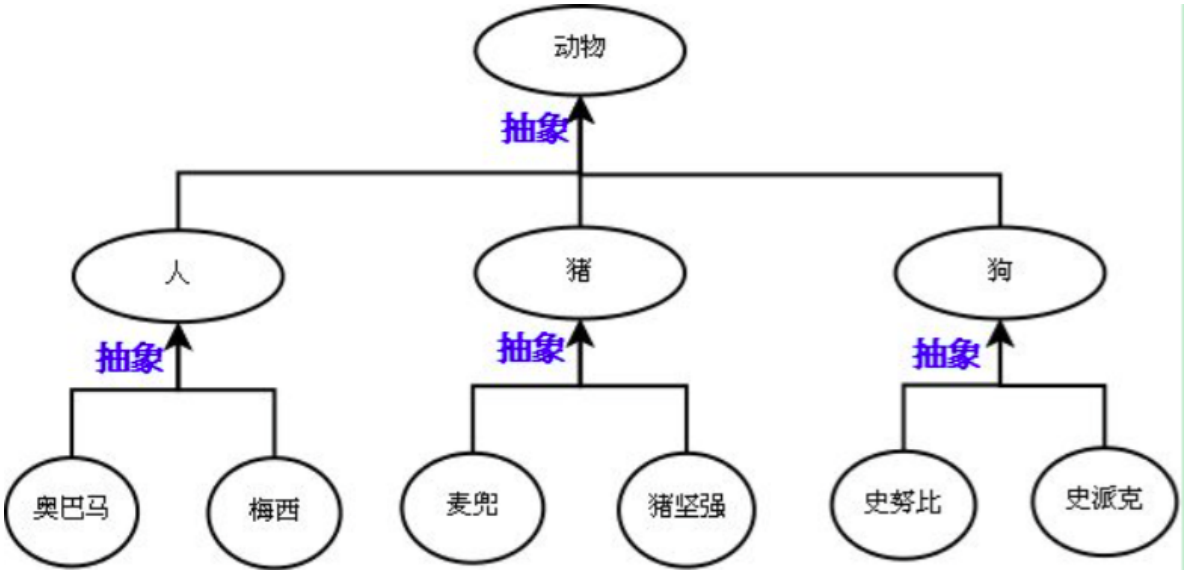
抽象即抽取类似或者说比较像的部分。

抽象分成两个层次：

1.将奥巴马和梅西这俩对象比较像的部分抽取成类；

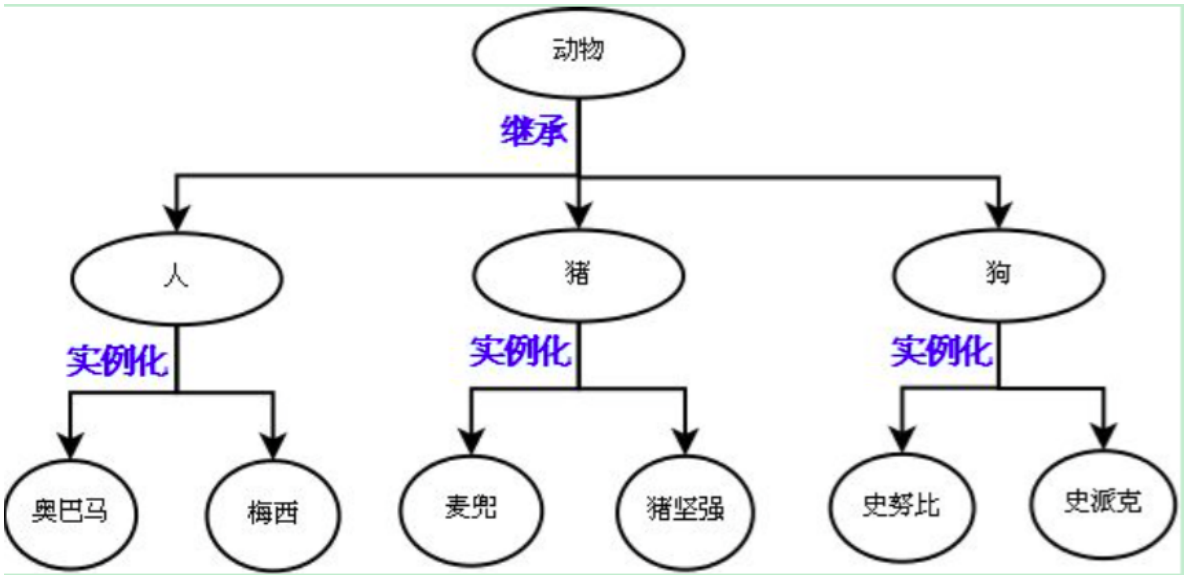
2.将人，猪，狗这三个类比较像的部分抽取成父类。

抽象最主要的作用是划分类别（可以隔离关注点，降低复杂度）



继承：是基于抽象的结果，通过编程语言去实现它，肯定是先经历抽象这个过程，才能通过继承的方式去表达出抽象的结构。

抽象只是分析和设计的过程中，一个动作或者说一种技巧，通过抽象可以得到类



多态性

一 什么是多态动态绑定（在继承的背景下使用时，有时也称为多态性）

多态性是指在不考虑实例类型的情况下使用实例

在面向对象方法中一般是这样表述多态性：
向不同的对象发送同一条消息（！！！ obj.func()：是调用了obj的方法func，又称为向obj发送了一条消息func），不同的对象在接收时会产生不同的行为（即方法）。
也就是说，每个对象可以用自己的方式去响应共同的消息。所谓消息，就是调用函数，不同的行为就是指不同的实现，即执行不同的函数。

比如：老师.下课铃响了（），学生.下课铃响了()，老师执行的是下班操作，学生执行的是放学操作，虽然二者消息一样，但是执行的效果不同