**PYTHON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。面向过程的程序 = 算法 + 数据结构； 关心解决问题的步骤。

**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，著名的例子有Linux內核，git，以及Apache HTTP Server等。

**面向对象**的程序设计的核心是对象（上帝式思维），要理解对象为何物，必须把自己当成上帝，上帝眼里世间存在的万物皆为对象，不存在的也可以创造出来。面向对象的程序 = 对象 + 交互； 关心谁在解决问题。

**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。

面向对象涉及的名词：类、对象、实例、实例化

类：具有相同特征的一类事物(人、狗、老虎) 一个抽象的概念，即为生活中的”类别”。

对象／实例：具体的某一个事物（隔壁老王、楼下旺财）对象：类的具体实例，即归属于某个类别的”个体”。 同类型的多个对象，行为相同，数据不同。

实例化：类——>对象的过程（这在生活中表现的不明显，我们在后面再慢慢解释）

python中一切皆为对象，类型的本质就是类。

**在python中，用变量表示特征，用函数表示技能，因而具有相同特征和技能的一类事物就是‘类’，对象是则是这一类事物中具体的一个。**

### 主要思想

**分而治之** --- 将一个大的需求分解为许多类，每个类处理一个独立的模块。

拆分好处：独立模块便于分工，每个模块便于复用，可扩展性强。

**封装变化** --- 变化的地方独立封装，避免影响其他模块。

**高 内 聚** --- 类中各个方法都在完成一项任务(单一职责的类)。

复杂的实现封装在内部，对外提供简单的调用。

**低 耦 合** --- 类与类的关联性依赖度要低(每个类独立)。

让一个模块的改变，尽少影响其他模块。

[例如：硬件高度集成化，又要可插拔]

最高的内聚莫过于类中仅包含1个方法，将会导致高内聚高耦合。

最低的耦合莫过于类中包含所有方法，将会导致低耦合低内聚。

### 面向对象的三大特性

**封装：**

数据角度讲，将一些基本数据类型复合成一个自定义类型。

方法角度讲，向类外提供功能，隐藏实现的细节。

**设计角度讲**，分而治之，高内聚低耦合，封装变化。

**作用**

1. 松散耦合，降低了程序各部分之间的依赖性。
2. 简化编程，使用者不必了解具体的实现细节，只需要调用对外提供的功能。
3. 增强安全性，以特定的访问权限来使用类成员，保护成员不被意外修改。

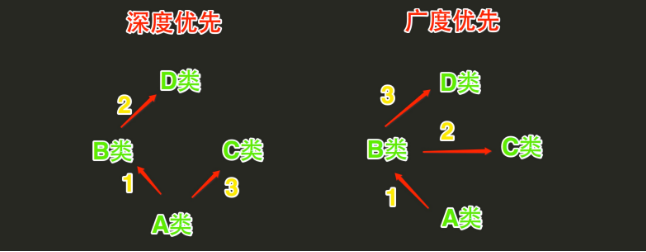
**继承：**

统一概念。

重用现有类的功能，在此基础上进行扩展。

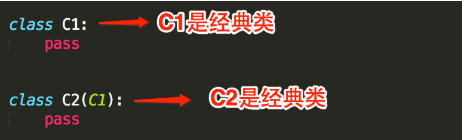
1、Python的类可以继承多个类，Java和C#中则只能继承一个类

2、Python的类如果继承了多个类，那么其寻找方法的方式有两种，分别是：**深度优先**和**广度优先**



* 当类是经典类时，多继承情况下，会按照深度优先方式查找
* 当类是新式类时，多继承情况下，会按照广度优先方式查找

经典类和新式类，从字面上可以看出一个老一个新，新的必然包含了跟多的功能，也是之后推荐的写法，从写法上区分的话，如果 **当前类或者父类继承了object类**，那么该类便是新式类，否则便是经典类。



经典类：首先去**A**类中查找，如果A类中没有，则继续去**B**类中找，如果B类中么有，则继续去**D**类中找，如果D类中么有，则继续去**C**类中找，如果还是未找到，则报错。

新式类：首先去**A**类中查找，如果A类中没有，则继续去**B**类中找，如果B类中么有，则继续去**C**类中找，如果C类中么有，则继续去**D**类中找，如果还是未找到，则报错。

注意：在上述查找过程中，一旦找到，则寻找过程立即中断，便不会再继续找了

**优点**

1. 复用代码的一种方式。
2. 统一概念，以层次化的方式管理类。

**缺点**

耦合度高：父类的改变直接影响到所有的子类，而不需要通知子类。

**适用性**

1. 多个类具有相同的数据或行为。
2. 多个类从概念上是一致的，且需要进行统一处理。

**多态：**

父类的同一种动作或者行为，在不同的子类上有不同的实现。

(父类调用同一方法，在不同的子类上有不同的执行效果)

实现手段：虚方法、抽象方法、接口方法。