类是 Python 面向对象程序设计的主要工具，通过这种程序设计方法，我们可以把代码的冗余度降到最低，并且通过定制现有的代码来编写新的程序，而不是在原处修改，它提供了代码的定制和复用的机制

什么是类什么是实例?

类是实例的工厂 类的属性提供了行为 所有从类产生的实例都继承了该类的属性

实例代表程序领域中的具体元素 实例属性记录数据 每个特定对象的数据都不同

为什么使用类?

多重实例:

类是产生对象的工厂，每次调用一个类 就会产生一个具有独立命名空间的新对象，每个由类产生的对象都可以读取类的属性，并获得自己的命名空间来存储数据， 这个数据对于每个对象来说都是不同的

继承:

我们可以在类的外部重新定义其属性 来扩充这个类

组合:

类是一些组件的集合，这些组件以团队的形式共同工作，每个组件都定义了自己的行为以及和其他组件之间的关系

属性搜索继承

在 Python 中，关于类属性 类方法的操作都可以使用如下的表达式（类方法：类中的函数 类属性:类中定义的变量的赋值）

object.attribute　　这个表达式会在 python 中启动搜索，搜索对象连接数 来寻找 attribute 首次出现的对象，　object—>该对象上的所有类　从下到上 从左到右，我们称这个搜索程序为继承，因为树中位置较低的对象继承了树中位置较高对象拥有的属性。

编写类树

1. 每个 class 语句会生成一个新的类对象

2.每次类调用时 就会生成一个新的实例对象

3.实例自动连接到创建了这些实例的类

4.类连接到其超类的方式是 将超类列在类头部的括号内 其从左到右的顺序决定了类树中的次序

创建一个类树

c2　　c3

.x　　.w

.z　　.z

c1

.x

.y

i1　　i2

.name .name

有类 c2 c3 c1 c1 的实例 i1, i2

在执行 i2.w 的时候 会以 i2 c1 c2 c3 的顺序进行类树搜索 找到首个 w 之后就停止搜索，在此例中直到搜索 c3 时才找到 w 因为 w 只出现在了 c3 中， 我们可以称之为 i2 从 c3 继承了属性 w

i1.x i2.x 都会在 c1 中找到 x 并停止搜索 因为 c1 比 c2 位置更底

i1.z i2.z 都会在 c2 中找到 z 因为 c2 比 c3 更靠左

i2.name 会在 i2 中找到 name 不需要爬树

本例中使用的是多重继承 在一个类树中 类有一个以上的超类 在 Python 中如果 class 语句中的小括号内有一个以上的超类 它会以从左到右的次序决定超类搜索的顺序

类中的方法（函数） 是多个实例的工厂，类方法的第一个参数是 self，代表调用这个方法的那个实例如

```
class c1:
    def __init__(self, value):
        self.name = value

    def getName(self):
        print self.name
```

x = c1('sf-express')

x.getName()

[root@yeslab p

构造函数 __init__

每次从类产生实例的时候，Python 会自动调用名为__init__的方法， 在实例调用的过程中传入的参数会被__init__所捕获，第一个参数 self，代表调用了这个类的实例

回顾

1 python 面向对象的意义是什么

分解代码 最小化代码冗余 对现存的代码进行定制来编写程序 而不是修改代码重头开始

2.类对象和实例对象有什么不同

3 类方法函数的第一个参数为什么是 Self

4__init__方法的用途是什么

5 怎样创建类的实例

6 怎样创建类

8 怎样定义类的超类