1. 类的继承：

继承关系：继承是相对两个类而言的父子关系，子类继承了父类的所有**公有属性和方法**，继承实现了代码重用。

* 1. **使用继承**

1. class Myclass(ParentClass)

如果父类定义了init方法，子类必须显式调用父类的init方法：

1. ParentClass.init(self,[args…])

如果子类需要扩展父类的行为，可以添加init方法的参数

* 1. **多重继承**
* 语法：

class class\_name(Parent\_c1,Parent\_c2,….)

* 注意：

当父类中出现多个自定义的init方法时，多重继承只执行第一个类的init方法，其它不执行。

1. class People(object):
2. color = 'yellow'
3. # def \_\_init\_\_(self, c):#参数大于等于2个，子类必须显式调用；只有一个参数的话，可以直接调用。单一继承
4. def \_\_init\_\_(self):#多重继承
5. print "Init..."
6. self.dwell = 'Earth'
7. self.color = 'yellow'
8. def think(self):
9. print "I am a %s" %self.color
10. print "My home is %s" %self.dwell
11. print "I am a thinker"
12. class Martian(object):
13. color = 'red'
14. def \_\_init\_\_(self):
15. self.dwell = 'Martian'
16. class Chinese(Martian, People): #多重调用跟位置关系有关；可通过显式调用改变
17. def \_\_init\_\_(self):
18. # People.\_\_init\_\_(self, 'red')
19. People.\_\_init\_\_(self)
20. # super(Chinese, self).\_\_init\_\_( 'red') #通过super函数继承父类
21. # pass
22. # def talk(self):
23. # print("I like talking")
24. # def think(self): #这里通过子类对父类的重写
25. # print("I like talking")
26. cn = Chinese()
27. # print cn.color #继承父类的color
28. cn.think()
29. # cn.talk()
30. 类的属性-总结

* 类的属性，也有公有属性
* 类的私有属性
* 对象的公有属性
* 对象的私有属性
* 内置属性
* 函数的局部变量

1. class MyClass(object):
2. var1 = '类属性，类的公有属性 var1'
3. \_\_var2 = '类的私有属性 \_\_var2'
4. def func1(self):
5. self.var3 = '对象的公有属性 var3' #对象属性，只能对象访问
6. self.\_\_var4 = '对象的私有属性 \_\_var4' #无法通过对象访问，类的外面
7. var5 = '函数的局部变量 var5' #只能在函数的内部访问
8. print self.\_\_var4
9. print var5
10. def func2(self):
11. print self.var1
12. print self.\_\_var2
13. print self.var3
14. print self.\_\_var4
15. mc = MyClass()
16. mc.func1()
17. print('\*\*\*\*\*\*' \* 10)
18. mc.func2()
19. print('\*\*\*\*\*\*' \* 10)
20. print mc.\_\_dict\_\_
21. print MyClass.\_\_dict\_\_
22. # # print mc.var1 #公有属性
23. # # print mc.\_MyClass\_\_var2 #私有属性
24. # # print mc.var3
25. 类的方法-总结

* 公有方法
* 私有方法
* 类方法
* 静态方法
* 内置方法

1. class MyClass(object):
2. name = 'Test'
3. def \_\_init\_\_(self): #只需进行类的实例化，即可被调用。其它可不用
4. self.func1()
5. self.\_\_func2()
6. self.classFun()
7. self.staticFun()
8. def func1(self):
9. print self.name,
10. print "我是公有方法"
11. # self.\_\_func2() #私有方法需通过内部调用，的方式间接调用
12. def \_\_func2(self):
13. print self.name,
14. print "我是私有方法"
15. *@classmethod*
16. def classFun(self):
17. print self.name,
18. print "我是类方法"
19. *@staticmethod*
20. def staticFun():
21. print MyClass.name,
22. print "我是静态方法"
23. mc = MyClass()
24. # mc.func1()
25. # MyClass.classFun()
26. # MyClass.staticFun()