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。

1.1 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：

```
2. 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

1.2 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.
4.     # 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
5.     def __init__(self):#多重继承
6.         print "Init..."
7.         self.dwell = 'Earth'
8.         self.color = 'yellow'
9.
10.       def think(self):
11.           print "I am a %s" %self.color
12.           print "My home is %s" %self.dwell
13.           print "I am a thinker"
14.
15.   class Martian(object):
16.       color = 'red'
17.
```

```
        def __init__(self):
            self.dwell = 'Martian'

    class Chinese(Martian, People): #多重调用跟位置关系有关；可通过显式调用改变
        def __init__(self):
            # People.__init__(self, 'red')
            People.__init__(self)
            # super(Chinese, self).__init__( 'red') #通过 super 函数继承父类
        # pass
        # def talk(self):
        #     print("I like talking")
        # def think(self): #这里通过子类对父类的重写
        #     print("I like talking")

    cn = Chinese()
    # print cn.color #继承父类的 color
    cn.think()
    # cn.talk()
```

2. 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

```
class MyClass(object):
    var1 = '类属性，类的公有属性 var1'
    __var2 = '类的私有属性 __var2'

    def func1(self):
```

```
6.          self.var3 = '对象的公有属性 var3'   #对象属性，只能
   对象访问
7.          self.__var4 = '对象的私有属性 __var4'   #无法通过对
   象访问，类的外面
8.          var5 = '函数的局部变量 var5'   #只能在函数的内部访问
9.          print self.__var4
10.             print var5
11.
12.         def func2(self):
13.             print self.var1
14.             print self.__var2
15.             print self.var3
16.             print self.__var4
17.
18.     mc = MyClass()
19.     mc.func1()
20.     print('******' * 10)
21.     mc.func2()
22.     print('******' * 10)
23.     print mc.__dict__
24.     print MyClass.__dict__
25.     # # print mc.var1  #公有属性
26.     # # print mc._MyClass__var2 #私有属性
27.     # # print mc.var3
```

3. 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.
4.      def __init__(self): #只需进行类的实例化，即可被调用。其它可不用
5.          self.func1()
6.          self.__func2()
7.          self.classFun()
8.          self.staticFun()
9.
10.         def func1(self):
11.             print self.name,
12.             print "我是公有方法"
13.             # self.__func2() #私有方法需通过内部调用，的方式间接调用
14.
15.         def __func2(self):
16.             print self.name,
17.             print "我是私有方法"
18.
19.         @classmethod
20.         def classFun(self):
21.             print self.name,
22.             print "我是类方法"
23.
24.         @staticmethod
25.         def staticFun():
26.             print MyClass.name,
27.             print "我是静态方法"
28.
29.    mc = MyClass()
30.    # mc.func1()
31.    # MyClass.classFun()
32.    # MyClass.staticFun()
```