python 小甲鱼视频笔记

类和对象

python无处不对象

对象

对象 = 属性 + 方法

面向对象编程的特征：

- 封装 信息隐藏技术
- 继承 子类自动共享父类的数据和方法的机制
- 多态 不同对象对同一方法相应不同的操作

```
OOA  面向对象分析
OOD  面向对象设计
OOP  面向对象编程
```

python的self = C++的this

```
class Ball:
    def setName(self,name):
        self.name = name
    def kick(self):
        print("我叫%s"%self.name)

a = Ball()
a.setName("dsdsdf")
b = Ball()
b.setNAme("sdf")
c = Ball()
c.setName("sdsdfsf")
a.kick()
c.kick()
```

init 构造函数

```
class Ball:
    def __init__(self,name):
        self.name = name
    def kick(self):
        print("我叫%s"%self.name)
b = Ball("asdas")
b.kick()
```

共有和私有

对象的属性都是公开的，可以通过点操作符来访问。 为了实现类似私有的特征，python实现了**name mangling**来操作。

```
class Person():
    name = "xiao"
    __newname = "sad"

    def getName(self):
        return self.__name
p = Person()
print p.name
print p.__name    wrong
p.getName()
print p._Person__name
```

python是伪私有。

继承

```
class Parent:
```

```
    def hello(self):
        print("调用父类的方法")

class Parent2:
    def hello2(self):
        print("调用父类2的方法")
class Child(Parent):
    pass

class Child2(Parent):
    def hello(self):
        print("调用子类的方法")

class Child3(Parent,Parent2):
    pass
p = Parent()
p.hello()
c = Child()
c.hello()

c2 = Child2()
c2.hello()
```

继承方法：

- 调用未绑定的父类方法
- 使用super()函数

当不确定必须要使用多重继承时，要避免使用它。

类，类对象和实例对象

```
class C:
    count = 0
a = C()
b = C()
c = C()
print a.count
print b.count
print c.count

c.count += 10
print c.count
C.count += 100
print a.count
print c.count
```

类属性和类对象相互绑定，不会受实例对象影响。如果实例对象的属性和方法名相同，会覆盖方法。

注意

- 不要试图在一个类里边定义出所有能想到的特性和方法，应该利用继承和组合机制来扩展。
- 用不同的词性命名，如属性名用名词，方法名用动词。

绑定

python严格要求方法需要有实例才能被调用，这种限制起始就是python所谓的绑定概念。

```
class CC:
    def setXY(self,x,y):
       self.x = x
       self.y = y
    def printxy(self):
        print(self.x,self.y)

dd = CC()
print dd.__dict__
```

```
    print CC.__dict__
    dd.setXY(4,5)
    print dd.__dict__
    print CC.__dict__
```

类中定义的属性,方法是静态变量，即使删除了类，实例也可调用方法。