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。

一些相关的BIF

hasattr(object,name)

> 判断是否有该属性

getattr(object,name[,default])

setattr(object,name,value)

delattr(object,name)

> 删除对象中指定的属性，如果不存在，跳出一个attributeerror的异常

property(fget = None,fset = None,fdel = None,doc = None)

魔法方法:构造和析构

***new*(cls[,...])**

在实例化是，第一个被默认执行的是**new(cls[,....])**

```
class Capstr(str):
    def __new__(cls,string):
        string = string.upper()
        return str.__new__(cls,string)

a = Capstr("i love you")
print a
```

***del*(self)**

垃圾回收机制自动调用该方法来销毁无用的变量

```
class C:
    def __init__(self):
        print("我是__init__  我被调用")
    def __del__(self):
        print("我是__del__  我被调用")
c1 = C()
del c1
```

魔法方法：算术运算

```
class New_int(int):
    def __add__(self,other):
        return int.__sub__(self,other)
    def __sub__(self,other):
        return int.__add__(self,other)
a = New_int(3)
b = New_int(5)
print a+b !
```

__add__(self, other)	定义加法的行为：+
__sub__(self, other)	定义减法的行为：-
__mul__(self, other)	定义乘法的行为：*
__truediv__(self, other)	定义真除法的行为：/
__floordiv__(self, other)	定义整数除法的行为：//
__mod__(self, other)	定义取模算法的行为：%

__divmod__(self, other)	定义当被 divmod() 调用时的行为
__pow__(self, other[, modulo])	定义当被 power() 调用或 ** 运算时的行为
__lshift__(self, other)	定义按位左移位的行为：<<
__rshift__(self, other)	定义按位右移位的行为：>>
__and__(self, other)	定义按位与操作的行为：&
__xor__(self, other)	定义按位异或操作的行为：^
__or__(self, other)	定义按位或操作的行为：|

魔法方法	含义
	基本的魔法方法
__new__(cls[, ...])	1. __new__ 是在一个对象实例化的时候所调用的第一个方法 2. 它的第一个参数是这个类，其他的参数是用来直接传递给 __init__ 方法 3. __new__ 决定是否要使用该 __init__ 方法，因为 __new__ 可以调用其他类的构造方法或者直接返回别的实例对象来作为本类的实例，如果 __new__ 没有返回实例对象，则 __init__ 不会被调用 4. __new__ 主要是用于继承一个不可变的类型比如一个 tuple 或者 string
__init__(self[, ...])	构造器，当一个实例被创建的时候调用的初始化方法
__del__(self)	析构器，当一个实例被销毁的时候调用的方法
__call__(self[, args...])	允许一个类的实例像函数一样被调用：x(a, b) 调用 x.__call__(a, b)
__len__(self)	定义当被 len() 调用时的行为
__repr__(self)	定义当被 repr() 调用时的行为
__str__(self)	定义当被 str() 调用时的行为
__bytes__(self)	定义当被 bytes() 调用时的行为
__hash__(self)	定义当被 hash() 调用时的行为
__bool__(self)	定义当被 bool() 调用时的行为，应该返回 True 或 False
__format__(self, format_spec)	定义当被 format() 调用时的行为

	有关属性
__getattr__(self, name)	定义当用户试图获取一个不存在的属性时的行为
__getattribute__(self, name)	定义当该类的属性被访问时的行为
__setattr__(self, name, value)	定义当一个属性被设置时的行为
__delattr__(self, name)	定义当一个属性被删除时的行为
__dir__(self)	定义当 dir() 被调用时的行为
__get__(self, instance, owner)	定义当描述符的值被取得时的行为
__set__(self, instance, value)	定义当描述符的值被改变时的行为
__delete__(self, instance)	定义当描述符的值被删除时的行为

	比较操作符
__lt__(self, other)	定义小于号的行为：x < y 调用 x.__lt__(y)
__le__(self, other)	定义小于等于号的行为：x <= y 调用 x.__le__(y)
__eq__(self, other)	定义等于号的行为：x == y 调用 x.__eq__(y)
__ne__(self, other)	定义不等号的行为：x != y 调用 x.__ne__(y)
__gt__(self, other)	定义大于号的行为：x > y 调用 x.__gt__(y)

__gt__(self, other)	定义大于号的行为：x > y 调用 x.__gt__(y)
__ge__(self, other)	定义大于等于号的行为：x >= y 调用 x.__ge__(y)

	算数运算符
__add__(self, other)	定义加法的行为：+
__sub__(self, other)	定义减法的行为：-
__mul__(self, other)	定义乘法的行为：*
__truediv__(self, other)	定义真除法的行为：/
__floordiv__(self, other)	定义整数除法的行为：//
__mod__(self, other)	定义取模算法的行为：%
__divmod__(self, other)	定义当被 divmod() 调用时的行为
__pow__(self, other[, modulo])	定义当被 power() 调用或 ** 运算时的行为
__lshift__(self, other)	定义按位左移位的行为：<<
__rshift__(self, other)	定义按位右移位的行为：>>
__and__(self, other)	定义按位与操作的行为：&
__xor__(self, other)	定义按位异或操作的行为：^
__or__(self, other)	定义按位或操作的行为：|

	反运算
__radd__(self, other)	（与上方相同，当左操作数不支持相应的操作时被调用）
__rsub__(self, other)	（与上方相同，当左操作数不支持相应的操作时被调用）
__rmul__(self, other)	（与上方相同，当左操作数不支持相应的操作时被调用）
__rtruediv__(self, other)	（与上方相同，当左操作数不支持相应的操作时被调用）
__rfloordiv__(self, other)	（与上方相同，当左操作数不支持相应的操作时被调用）
__rmod__(self, other)	（与上方相同，当左操作数不支持相应的操作时被调用）
__rdivmod__(self, other)	（与上方相同，当左操作数不支持相应的操作时被调用）
__rpow__(self, other)	（与上方相同，当左操作数不支持相应的操作时被调用）
__rlshift__(self, other)	（与上方相同，当左操作数不支持相应的操作时被调用）
__rrshift__(self, other)	（与上方相同，当左操作数不支持相应的操作时被调用）
__rxor__(self, other)	（与上方相同，当左操作数不支持相应的操作时被调用）
__ror__(self, other)	（与上方相同，当左操作数不支持相应的操作时被调用）

	增量赋值运算
__iadd__(self, other)	定义赋值加法的行为：+=
__isub__(self, other)	定义赋值减法的行为：-=
__imul__(self, other)	定义赋值乘法的行为：*=
__itruediv__(self, other)	定义赋值真除法的行为：/=
__ifloordiv__(self, other)	定义赋值整数除法的行为：//=
__imod__(self, other)	定义赋值取模算法的行为：%=
__ipow__(self, other[, modulo])	定义赋值幂运算的行为：**=
__ilshift__(self, other)	定义赋值按位左移位的行为：<<=
__irshift__(self, other)	定义赋值按位右移位的行为：>>=
__iand__(self, other)	定义赋值按位与操作的行为：&=
__ixor__(self, other)	定义赋值按位异或操作的行为：^=
__ior__(self, other)	定义赋值按位或操作的行为：|=

__ior__(self, other)	定义赋值按位或操作的行为：|=
	一元操作符
__neg__(self)	定义正号的行为：+x
__pos__(self)	定义负号的行为：-x
__abs__(self)	定义当被 abs() 调用时的行为
__invert__(self)	定义按位求反的行为：~x
	类型转换
__complex__(self)	定义当被 complex() 调用时的行为（需要返回恰当的值）
__int__(self)	定义当被 int() 调用时的行为（需要返回恰当的值）
__float__(self)	定义当被 float() 调用时的行为（需要返回恰当的值）
__round__(self[, n])	定义当被 round() 调用时的行为（需要返回恰当的值）
__index__(self)	1. 当对象是被应用在切片表达式中时，实现整形强制转换 2. 如果你定义了一个可能在切片时用到的定制的数值型,你应该定义 __index__ 3. 如果 __index__ 被定义，则 __int__ 也需要被定义，且返回相同的值
	上下文管理（with 语句）
__enter__(self)	1. 定义当使用 with 语句时的初始化行为 2. __enter__ 的返回值被 with 语句的目标或者 as 后的名字绑定
__exit__(self, exc_type, exc_value, traceback)	1. 定义当一个代码块被执行或者终止后上下文管理器应该做什么 2. 一般被用来处理异常，清除工作或者做一些代码块执行完毕之后的日常工作
	容器类型
__len__(self)	定义当被 len() 调用时的行为（返回容器中元素的个数）
__getitem__(self, key)	定义获取容器中指定元素的行为，相当于 self[key]
__setitem__(self, key, value)	定义设置容器中指定元素的行为，相当于 self[key] = value
__delitem__(self, key)	定义删除容器中指定元素的行为，相当于 del self[key]
__iter__(self)	定义当迭代容器中的元素的行为
__reversed__(self)	定义当被 reversed() 调用时的行为
__contains__(self, item)	定义当使用成员测试运算符（in 或 not in）时的行为