Python的类中成员有三种。

**字段**：字段分为两种。一种是静态字段，另一个是普通字段。

普通字段例如self.name，字段属于每一个实例对象，保存在每一个实例对象当中。访问普通字段使用对象.普通字段访问。

静态字段例如name,没有用self修饰，静态字段属于类，保存在类中，被所有对象共用。一般情况下使用类.静态字段访问。另外Python还可以使用对象.静态字段访问。

**属性**：

Python中的属性第一种表现形式就是将方法以字段的形式去获取值，设置值，以及删除值。

***第一种表现形式中***，对方法进行@property修饰符修饰。例如:

Class Example:

@property

def test(self):

print(“AAA”)

@test.setter

def test(self,value):

print(value)

@test.deleter

def test(self):

print(“BBB”)

example = Example()

[当使用@property以后那么可以 example.test那么python将会执行Example类中的被@property 修饰test](mailto:当使用@property以后那么可以%20example.test那么python将会执行Example类中的被@property%20修饰test)方法。[当使用@test.setter以后，可以使用example.test](mailto:当使用@test.setter以后，可以使用example.test) = “GGG” 那么Python将执行Example中被@test.setter的方法。[当使用@test.deleter以后，可以使用del example.test那么Python将执行Example](mailto:当使用@test.deleter以后，可以使用del%20example.test那么Python将执行Example)中被@test.deltter修饰的方法。

P.s 需要注意的是通过属性，Python的主要目的是使得可以用字段的形式去调用方法。建立起一个映射关系。但是调用的方法执行的是什么，完全自己定义。所以del example.test [调用的被@test.deleter](mailto:调用的被@test.deleter)修饰的方法中是否真的去删除什么字段，或者只是简单的输出都是允许的。而对于真正的字段而言，del 字段则只会真的从这个实例中，删除这个字段。

P.s 另外我们用@property修饰test作为字段，在接下来的setter修饰器，和deleter修饰器中定义的方法名都要以test作为方法名。并且我们无法直接通过example.test()去调用 test()方法了。

第二种表现实现，第二种形式和第一种形式一样。目的都是可以用字段的形式去调用方法。

Class Example:

def test1(self):

print(“AAA”)

def test2(self,value):

print(value)

def test3(self):

print(“BBB”)

foo = property(fget=test1,fset=test2,fdel=test3)

example = Example()

这种形式的执行属性方法比较灵活。一样的现在为Example类指定了一个属性foo,当执行Example.foo, 设置Example.foo或者del Example.foo以后都会映射到某一个函数。这种形式的灵活在于当执行完属性以后还可以使用example.test1()去分别调用test1方法。

**方法**：Python类中有三种方法。

第一种是普通方法，就是由类的实例对象来调用。

第二种是静态方法，参数中不含self,并且由@staticmethod修饰。静态方法可以由类来调用，也可以由实例来调用。（一般如果类方法中不需要用到实例self的字段，那么可以改成静态方法）

第三种是类方法，[参数中必须有一个参数cls并且由@classmethod修饰。并且cls和self](mailto:参数中必须有一个参数cls并且由@classmethod修饰。并且cls和self)一样是自动传递的。类方法与静态方法不同在于，类方法会自动传递类cls作为参数。类方法方法可以由类来调用，也可以由实例来调用。