Python一切皆对象(object)，每个对象都可能有多个属性(attribute)。Python的属性有一套统一的管理方案。

**属性的\_\_dict\_\_系统**

对象的属性可能来自于其类定义，叫做**类属性(class attribute)**。类属性可能来自类定义自身，也可能根据类定义继承来的。一个对象的属性还可能是该对象实例定义的，叫做**对象属性(object attribute)**。

对象的属性储存在对象的\_\_dict\_\_属性中。\_\_dict\_\_为一个词典，键为属性名，对应的值为属性本身。我们看下面的类和对象。chicken类继承自bird类，而summer为chicken类的一个对象。

class bird(object):

feather = True

class chicken(bird):

fly = False

def \_\_init\_\_(self, age):

self.age = age

summer = chicken(2)

print(bird.\_\_dict\_\_)

#{'\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'bird' objects>, '\_\_module\_\_': '\_\_main\_\_', '\_\_weakref\_\_': #<attribute '\_\_weakref\_\_' of 'bird' objects>, 'feather': True, '\_\_doc\_\_': None}

print(chicken.\_\_dict\_\_)

#{'fly': False, '\_\_module\_\_': '\_\_main\_\_', '\_\_doc\_\_': None, '\_\_init\_\_': <function \_\_init\_\_ at #0x0000000004808358>}

print(summer.\_\_dict\_\_)

#{'age': 2}

第一行为bird类的属性，比如feather。第二行为chicken类的属性，比如fly和\_\_init\_\_方法。第三行为summer对象的属性，也就是age。有一些属性，比如\_\_doc\_\_，并不是由我们定义的，而是由Python自动生成。此外，bird类也有父类，是object类(正如我们的bird定义，class bird(object))。这个object类是Python中所有类的父类。

可以看到，Python中的属性是分层定义的，比如这里分为object/bird/chicken/summer这四层。当我们需要调用某个属性的时候，Python会一层层向上遍历，直到找到那个属性。(某个属性可能出现再不同的层被重复定义，Python向上的过程中，会选取先遇到的那一个，也就是比较低层的属性定义)。

当我们有一个summer对象的时候，分别查询summer对象、chicken类、bird类以及object类的属性，就可以知道summer对象所有的\_\_dict\_\_，就可以找到通过对象summer可以调用和修改的所有属性了。下面两种属性修改方法等效：

summer.\_\_dict\_\_['age'] = 3

print(summer.\_\_dict\_\_['age'])

summer.age = 5

print(summer.age)

(上面的情况中，我们已经知道了summer对象的类为chicken，而chicken类的父类为bird。如果只有一个对象，而不知道它的类以及其他信息的时候，我们可以利用\_\_class\_\_属性找到对象的类，然后调用类的\_\_base\_\_属性来查询父类)

**特性**

同一个对象的不同属性之间可能存在依赖关系。当某个属性被修改时，我们希望依赖于该属性的其他属性也同时变化。这时，我们不能通过\_\_dict\_\_的方式来静态的储存属性。Python提供了多种即时生成属性的方法。其中一种称为特性(property)。特性是特殊的属性。比如我们为chicken类增加一个特性adult。当对象的age超过1时，adult为True；否则为False：

class bird(object):

feather = True

class chicken(bird):

fly = False

def \_\_init\_\_(self, age):

self.age = age

def getAdult(self):

if self.age > 1.0: return True

else: return False

adult = property(getAdult) # property is built-in

summer = chicken(2)

print(summer.adult) # True

summer.age = 0.5

print(summer.adult) # False

特性使用内置函数property()来创建。property()最多可以加载四个参数。前三个参数为函数，分别用于处理查询特性、修改特性、删除特性。最后一个参数为特性的文档，可以为一个字符串，起说明作用。

我们使用下面一个例子进一步说明:

class num(object):

def \_\_init\_\_(self, value):

self.value = value

def getNeg(self):

return -self.value

def setNeg(self, value):

self.value = -value

def delNeg(self):

print("value also deleted")

del self.value

neg = property(getNeg, setNeg, delNeg, "I'm negative")

x = num(1.1)

print(x.neg) # -1.1

x.neg = -22

print(x.value) # 22

print(num.neg.\_\_doc\_\_) # I'm negative

del x.neg # value also deleted

上面的num为一个数字，而neg为一个特性，用来表示数字的负数。当一个数字确定的时候，它的负数总是确定的；而当我们修改一个数的负数时，它本身的值也应该变化。这两点由getNeg和setNeg来实现。而delNeg表示的是，如果删除特性neg，那么应该执行的操作是删除属性value。property()的最后一个参数("I'm negative")为特性negative的说明文档。