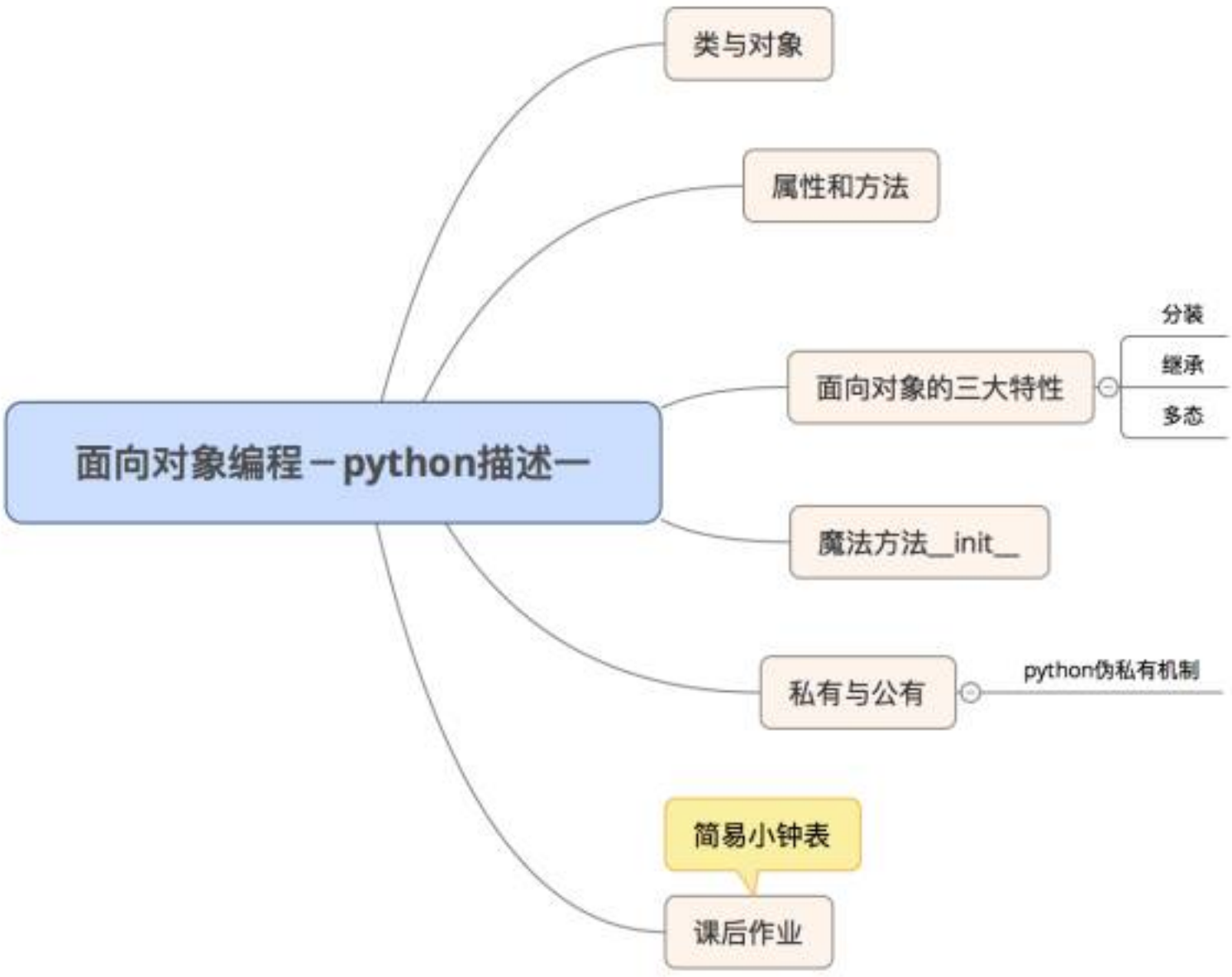
python面向对象描述（上）一起走进“巨人的花园”

Original 2016-12-18 哈哈小 菜猿观世界

通过本节的学习，你将了解一下内容：



面向对象一词，早闻已久，却不知其所以然。

希望通过本节的阅读，让你日后再见，明朗清晰于心。

1.类与对象（class and object）

让我们从艺术才子王尔德创作的童话故事《巨人的花园》讲起。

“每天下午，孩子们放学回来之后，总爱到巨人的花园里去游玩。

这是一个美丽的大花园。草丛中盛开着美丽的花朵。另外园里还有12株果树。春天来临时，树上开满红色和白色的花朵；秋天到来时，树上果实累累。鸟儿在树上歌颂，唱得那么动听，孩子们有时会停止游戏，来倾听鸟儿唱歌。他们彼此欢叫着：“咱们在这儿多快乐呀！”－引用自文章一，二段

通过上面这两段的描述，我们现在聚焦这巨人的花园。

如果用程序对这个花园做初步的描述，可以是这样的：

```
class Garden: #start with upppercase
    """this is a class named Garden"""
    size=10000#花园大小,单位平方米，大花园，哈哈
    currentSeason="spring"#假设现在的花园处于鸟语花香的春季

    def switchSeason(self):   #美丽的花园一定是能四季更替的
        print("season is switched here")
```

上面程序片段，看作一个整体，就是今天要讲的第一个概念－类。

何为类？在这之前，我们先来看看类在python中的创建方式

语法：

```
class 类名:
    # 类的一些属性
    # 类的一些方法
```

类的定义，看起来也不过如此简单。

一个类主要包括两个部分，属性和方法。

通过上面花园的代码样例，想比大家对属性和方法也是初有所识的。

属性也叫成员变量，是对一个类的一些固有属性或者叫静态属性的表示，好比，现在这个花园有两个静态属性，大小size和当下季节－春季。在类中定义属性，我们平时怎么定义变量，这属性就怎么定义。

相反，方法也叫成员函数，是对该类所具备哪些动态功能的描述，正如花园具备四季更替的功能，所以就包

括switchSeason这么一个函数。在类中定义方法，我们平时怎么定义函数，这方法就差不多怎么定义，差一点在于后面哈哈小会提到的self。

如果还是不甚理解或不闲哈哈小啰嗦的，这里再打个比方，就以人为例，好比人这么一个抽象的概念就是这里花园，可以看做一个类。像人的属性，身高，体重，性别此而等等都是人的静态属性的描述。而人又有哪些功能呢？走路是吧，还有吃饭，还有？ 睡觉，没错，此而等等就可以说是人这么一个类的动态动能的描述了。

讲到这里，大家可依花园画人了，尝试着像哈哈小定义花园类一样将人这么一个类定义出来吧。

可以说，类本身就是抽象的，类这个抽象下的具体就是哈哈小现在要谈到的对象了。

对象是类下面的具体，这句话如何理解呢？打个比方，好比这个花园，刚才讲到的类就可以看做这个花园的设计图，他只是一种形式的存在，设计图只是对其做了形式的描述，不具备让孩子们在其中嬉戏玩耍，游客参观等实实在在真真切切的功能。这对象就好比是依照这个图纸建设出来具备实用功能的一座具体的花园。所以，真正的功能比如swithReason就是靠对象来实现的。同时，通过上段的描述，我们也不难知道，因为一张图纸可以建造各式各样的花园，所以通过类也是可以构建出多个互相独立的对象的。

细心的朋友有发现，在成员方法switchSeason后面跟着一个奇怪的单词self，这是何用?

刚才哈哈小讲过，由于一个类可能会创建出很多对象，而self指的是当前对象的指引。

switchSeason有了self参数，你就可以在改方法中通过`self.currentSeason`轻松访问到花园的`currentSeason`属性了。

```
print("season is switched here",self.currentSeason)
```

如果对上面这么拉长的一段还是稍感困惑，没关系，请接着看下文。

首先，我们需要来解决第一个问题，就是如何如何来创建一个类的对象，因为只有对象出来了，我们就好搞事情了。

有了解像java这样的面向对象语言的同学，知道是通过关键字new来创建一个对象的。

在python中，不好意思了，只好对像new这样的关键字说拜拜了。

```
>>> gardenObject1=Garden()
>>> gardenObject2=Garden()
```

直接用类名就就简单地创建了两个花园对象。

对象拿到手了，这下就该发挥一下花园的功能了。

> 查看一下，现在花园是处于哪个季节

```
>>> gardenObject1.current
'spring'
```

> 切换一下季节看看

```
>>> gardenObject1.switchSeason()
"the season is switched here"
```

虽然不是什么真正的季节切换，这里意思意思一下就好了，哈哈。

2.面向对象三大特性

不管是哪门面向对象的语言，当说到面向对象概念时，我想都逃不过对其三大特性的描述吧。

2.1 分装（encapsulation）

分装，顾它名思它义，举一个贴切生活的例子，分装就好比，你现在手里握着的手机，我们知道当它没电时，拿个数据线，一头对接上手机的电源接口，另一头接上电源，这就达到了为手机续航充电的目的。如果问出这样一个问题，手机它是具体经过了什么过程最终将电量储存在手机电板上的？我估计十个有八个对于这个问题，只能干抓头皮了，然后来一句：鬼知道它经历了什么。

其实，这里就有分装的概念，手机制造者对手机的功能进行了分装，对外界可见的也就是我们这里讲到的接口了。像它底层的实现，它的电路板怎么设计的等等这些问题就不是我们这些非专业人员所需了解的了，我们只要知道它的接口怎么用就行了。

再举一个程序的例子：

```
>>> list1=[8,3,2,1,0,9]
>>> list1.sort()
>>> list1
[0, 1, 2, 3, 8, 9]
>>>
```

还记得之前讲列表时，这个非常好用的sort方法吧。

既然讲了类的概念了，在这里可以把列表看做一个类，我们这里使用的sort方法就看作是它对外界提供的一个接口。

记得最初，哈哈小只告诉大家，这个方法具有将由数字组成的列表进行排序的功能，其他的你们就暂时不要关心了。其实这句话放在这里说的意思是，sort方法是列表将其所有功能分装后对外界提供的一个接口方法，我们无需关心其具体的实现是怎么样的。

真的不需要关心其它体实现吗?

no，要知道我们可是要成为程序员的哎，如果我们自己不具备分装程序的能力，那还不得落得个做手机的却不知道怎么装手机的尴尬境地了。

这里哈哈小插播一句话：你的实力有多厚，就看你“装”的有多高级。

分装总之一句话，就是对底层具体的隐藏。

2.2 继承（inheritance）

继承，程序抽象于生活又具体作用于生活，所以，这里还是依同此理，顾它名思它意。

如果说某位富二代继承了他爸的家业，他继承了他父亲的什么，所有他父亲的江山，对不对?

嗯哼，没错的。

回到文首提到的巨人的花园的例子，再用程序进一步说明。

现在我有另一种花园，是带围墙的花园。那我要声明这个类，是不是得照着上面的样再写一遍?

答案是否定了。这个带围墙的花园要具备花园所有的属性和方法，所以只需继承上面的花园类就行了。

> class 类名（父类）：
>
> #...

```
>>> class WalledGarden(Garden):
    pass # 这里我们暂时不想对WalledGarden申明其具备的属性或方法，直接pass就行了
```

这里的WalledGarden类我们称作Garden的子类，相反Garden成为WalledGarden的父类，子类继承的是父类的可见的所有。

一个类可以继承多个父类（父类一，父类二），父类之间逗号隔开。就好比那位富二代也可以继承他叔或他舅的江山。

这里初次提到继承，只是为了让大家对继承有个初步的了解，后续会更加深入理解。

2.3 多态（polymorphism）

多态，指的是不同对象对同一方法的实现的行动不同

如果大家觉得有点懵，哈哈小表示有必要举个例子解释一下

假如现在在WalledGarden子类中重写了switchSeason方法

```
class WalledGarden(Garden):
```

```
    def switchSeason(self):
        print("WalledGarden:season is switched here")
```

> 我们来试着玩一玩

```
>>> walled=WalledGarden()
>>> walled.switchSeason()
WalledGarden:season is not switched here
>>> garden=Garden()
>>> garden.switchSeason()
season is switched here
>>>
```

有发现，当使用walled对象调用switchSeason方法时，调用的是子类WalledGarden的switchSeason方法。当使用garden对象调用switchSeason方法时，调用的是父类Garden的switchSeason方法。不同对象对同一方法的实现的行动不同，讲的就是这个道理。

3.魔法方法__init__

魔法方法，听起来有点神奇，哈哈小第一次听到也是这样充满好奇的。

魔法不魔法，不过是程序在作怪。

在python定义的类中，有很多这样的魔法方法，今天暂时就讲一种__init__（注意一共是四个下划线）。

像这种带四只脚的“怪物”，拥有魔法也不奇怪啦。

之所以把__init__叫魔法方法，是因为这个方法是在“暗处”被调用的。看下面这个例子。

假如现在要对花园类增加一个叫owner的属性，而且要求是这个owner默认为“the gaint”，并且当在创建一个graden对象时，可以给owner值初始化，给任何你想给的值。

程序可以改造成如下样子：

```
class Garden: #start with upppercase
    """this is a class named Garden"""
    size=10000#花园大小,单位平方米，大花园，哈哈
    currentSeason="spring"#假设现在的花园处于鸟语花香的春季


    def __init__(self,owner="the gaint"):
```

```
        self.owner=owner

    def switchSeason(self):   #美丽的花园一定是能四季更替的
        print("season is switched here")
```

> 我们再来创建一个带初始化owner的花园变量

```
>>> garden=Garden('the children')

>>> garden.owner

'the children'

>>>
```

通过上面的程序，我们不难发现，当在创建一个对象，也就是执行garden=Garden('the children')

这一句代码时，其本质是调用了__init__ 方法。这里的__init__ 方法，我们称作为花园类的构造方法。

上例构造方法只能对一个成员变量初始化，当然，你也可以定义多个参数，对多个成员变量初始化。

4.共有和私有

共有的，到目前为止，我们接触的所有成员变量和方法都是共有的，因为他们都是可以被外界访问的到的。

这里指的被外界访问的到，意思是可以通过创建出来的对象索引的到。

比如，

```
>>> gardenObject=Garden()
>>> gardenObject.size
1000
```

这里能访问的到size，就是因为size在Garden 类中被定义成共有的了。

现在如果要让size定义成私有的，不被对象索引到，只需要在size变量前加两条下划线即可

Garden类中的size属性变成如下样子：

__size=1000

这时，我们再尝试访问size时

```
>>> gardenObject=Garden()
>>> gardenObject.size

Traceback (most recent call last):

  File "<pyshell#6>", line 1, in <module>

    g.size

AttributeError: 'Garden' object has no attribute 'size'
```

不好意思，已经访问不到size属性了。同样的方式，在成员方法前加两条下划线也能使其变成私有。

这里哈哈小额外提一下，如果我们硬要在这个时候去size变量，其实也是有法可寻的。

像这样，

```
>>> gardenObject._Garden__size

10000

>>>
```

对象._类名__ 私有成员，以这么一种方式照样行得通。

文章最后，稍作思考，又是一个珍贵的“原来如此”。

pyhton的私有机制不过是把私有成员改了个名字而已，好你个伪私有机制。

5.课后作业：

今天的课后作业是，结合今天所学面向对象相关知识，实现一个简易小钟表。

声明一个叫Clock的类

该类有：

三个属性，hour，minute，second。

四个方法，incrementHour, incrementMinute, incrementMinute，printTime。

构造对象时，对时间初始化，给三个属性赋值，比如时分秒分别为12，59，59。

通过构造出来的对象调用该类的前三个方法。

最后调用printTime方法测试最后的输出。

拓展题：

尝试改写switchSeason方法，完成一个真正春夏秋冬四季转换的功能。

“巨人的花园”才初探端倪， 未完待续。

结束语：我们不做知识的搬运工，幸能成为思想的启迪者。

注：阅读《巨人的花园》全文，请点击阅读全文。

Read more