类和实例

2.7旧版教程

面向对象最重要的概念就是类（Class）和实例（Instance），必须牢记类是抽象的模板，比如Student类，而实例是根据类创建出来的一个个具体的“对象”，每个对象都拥有相同的方法，但各自的数据可能不同。

仍以Student类为例，在Python中，定义类是通过`class`关键字：

```
class Student(object):
    pass
```

`class`后面紧接着是类名，即`Student`，类名通常是大写开头的单词，紧接着是`(object)`，表示该类是从哪个类继承下来的，继承的概念我们后面再讲，通常，如果没有合适的继承类，就使用`object`类，这是所有类最终都会继承的类。

定义好了`Student`类，就可以根据`Student`类创建出`Student`的实例，创建实例是通过类名+()实现的：

```
>>> bart = Student()
>>> bart
<__main__.Student object at 0x10a67a590>
>>> Student
<class '__main__.Student'>
```

可以看到，变量`bart`指向的就是一个`Student`的实例，后面的`0x10a67a590`是内存地址，每个object的地址都不一样，而`Student`本身则是一个类。

可以自由地给一个实例变量绑定属性，比如，给实例`bart`绑定一个`name`属性：

```
>>> bart.name = 'Bart Simpson'
>>> bart.name
'Bart Simpson'
```

由于类可以起到模板的作用，因此，可以在创建实例的时候，把一些我们认为必须绑定的属性强制填写进去。通过定义一个特殊的`__init__`方法，在创建实例的时候，就把`name`，`score`等属性绑上去：

```
class Student(object):

    def __init__(self, name, score):
        self.name = name
        self.score = score
```

注意到`__init__`方法的第一个参数永远是`self`，表示创建的实例本身，因此，在`__init__`方法内部，就可以把各种属性绑定到`self`，因为`self`就指向创建的实例本身。

有了`__init__`方法，在创建实例的时候，就不能传入空的参数了，必须传入与`__init__`方法匹配的参数，但`self`不需要传，Python解释器自己会把实例变量传进去：

```
>>> bart = Student('Bart Simpson', 59)
>>> bart.name
'Bart Simpson'
>>> bart.score
59
```

和普通的函数相比，在类中定义的函数只有一点不同，就是第一个参数永远是实例变量`self`，并且，调用时，不用传递该参数。除此之外，类的方法和普通函数没有什么区别，所以，你仍然可以用默认参数、可变参数、关键字参数和命名关键字参数。

数据封装

面向对象编程的一个重要特点就是数据封装。在上面的`Student`类中，每个实例就拥有各自的`name`和`score`这些数据。我们可以通过函数来访问这些数据，比如打印一个学生的成绩：

```
>>> def print_score(std):
...     print('%s: %s' % (std.name, std.score))
...
>>> print_score(bart)
Bart Simpson: 59
```

但是，既然`Student`实例本身就拥有这些数据，要访问这些数据，就没有必要从外面的函数去访问，可以直接在`Student`类的内部定义访问数据的函数，这样，就把“数据”给封装起来了。这些封装数据的函数是和`Student`类本身是关联起来的，我们称之为类的方法：

```
class Student(object):

    def __init__(self, name, score):
        self.name = name
        self.score = score

    def print_score(self):
        print('%s: %s' % (self.name, self.score))
```

要定义一个方法，除了第一个参数是`self`外，其他和普通函数一样。要调用一个方法，只需要在实例变量上直接调用，除了`self`不用传递，其他参数正常传入：

```
>>> bart.print_score()
Bart Simpson: 59
```

这样一来，我们从外部看`Student`类，就只需要知道，创建实例需要给出`name`和`score`，而如何打印，都是在`Student`类的内部定义的，这些数据和逻辑被“封装”起来了，调用很容易，但却不用知道内部实现的细节。

封装的另一个好处是可以给`Student`类增加新的方法，比如`get_grade`：

```
class Student(object):
    ...

    def get_grade(self):
        if self.score >= 90:
            return 'A'
        elif self.score >= 60:
            return 'B'
        else:
            return 'C'
```

```
    return 'C'
```

同样的，`get_grade`方法可以直接在实例变量上调用，不需要知道内部实现细节：

```
>>> bart.get_grade()
'C'
```

小结

类是创建实例的模板，而实例则是一个一个具体的对象，各个实例拥有的数据都互相独立，互不影响；

方法就是与实例绑定的函数，和普通函数不同，方法可以直接访问实例的数据；

通过在实例上调用方法，我们就直接操作了对象内部的数据，但无需知道方法内部的实现细节。

和静态语言不同，Python允许对实例变量绑定任何数据，也就是说，对于两个实例变量，虽然它们都是同一个类的不同实例，但拥有的变量名称都可能不同：

```
>>> bart = Student('Bart Simpson', 59)
>>> lisa = Student('Lisa Simpson', 87)
>>> bart.age = 8
>>> bart.age
8
>>> lisa.age
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Student' object has no attribute 'age'
```

参考源码

student.py

感觉本站内容不错，读后有收获？

¥ 我要小额赞助，鼓励作者写出更好的教程

还可以分享给朋友

分享 赶快成为第一个分享的人吧

面向对象编程

访问限制

评论

我想知道这个类的成员变量都是后来自己给定的吗？

董DHH董 created at 11-29 13:35, Last updated at 4天前

比如这个Student你没有定义它的成员变量，而是直接在init函数参数中传入name和score，python中就是这样的语法?

廖雪峰

Created at 11-30 9:19, Last updated at 11-30 9:19

动态语言就是这样，成员变量，包括方法都是动态绑定的

你在class定义的def xxx方法都是解释器动态绑定到class对象上的

Jaraxuss

Created at 4天前, Last updated at 4天前

不明白这样设计的意义。

廖雪峰

Created at 4天前, Last updated at 4天前

多写java代码

View Full Discuss　Reply This Topic

没整明白，为啥我的报错？

秀才的救赎 created at 10-25 15:06, Last updated at 11-25 15:39

```
在此插入代码
>>> class Student:
    def get_grade(self):
        self.score='54'
        if self.score>=90:
            return 'A'
        elif self.score>=60:
            return 'B'
        else:
            return 'C'


>>> a=Student
>>> a.get_grade()
Traceback (most recent call last):
  File "<pyshell#54>", line 1, in <module>
    a.get_grade()
TypeError: get_grade() missing 1 required positional argument: 'self'
```

Lam-友
Created at 11-9 15:35, Last updated at 11-9 15:35

self.score='54' 不需要引号 self.score=54

String 和 int 不能做比较

胡焕只喜欢静妹
Created at 11-16 21:50, Last updated at 11-16 21:50

```
a=Student
a

<class '__main__.Student'>
a=Student()
a

<__main__.Student object at 0x101a2ca90>
```

不要随便就少敲两个字符啊！！括号是极为重要的语法关键字，看见括号就把肾上腺素升起来，不然就全是坑！
不用括号，赋值给a的是Student这个类本身
用了括号就变成了调用，先调用Student的构造函数，再将函数调用的结果（即返回的对象实例）赋值给a

秀才的救赎
Created at 11-25 15:39, Last updated at 11-25 15:39

谢谢两位大只差，我犯了个低级错误

谢谢两位大兄弟，我犯了个低级错误

View Full Discuss | Reply This Topic

小作业

Young先森森 created at 11-22 10:14, Last updated at 11-22 10:14

class Student(object):
"""docstring for student"""
def **init**(self, name,score):
super(Student, self).**init**()
self.name = name
self.score = score

```
def printScore(self):
    print('name:%s score:%s' %(self.name,self.score))

def get_grade(self):
    if self.score >= 90:
        print('The score is: 优秀')
    elif self.score >= 80:
        print('The score is: 良好')
    elif self.score >= 70:
        print('The score is: 中等')
    elif self.score >= 60:
        print('The score is: 及格')
    else:
        print('The score is: 差')
```

Read More

View Full Discuss | Reply This Topic

报错了半天

老乡快开下门 created at 4-28 13:45, Last updated at 11-21 17:19

```
__init__
```

原来这里init前后各有两个下划线。

碉你妹的堡

Created at 8-1 0:22, Last updated at 8-1 0:22

兄弟要是没看到这评论，我今晚估计要郁闷死！！！

用户n7mfhsodoh

Created at 8-2 9:02, Last updated at 8-2 9:02

我也是

高室长
Created at 8-24 15:24, Last updated at 8-24 15:24

谢谢，谢谢，太感动了，我一个月前用windows，发现这个问题，怎么也调不好。怀疑是win太糟糕。现在花了一个月看鸟哥的liunx教程，上了Ubuntu，发现还有这个问题。
要不是看到你的评论，我都开始怀疑人生了。
感动。

耶路撒冷无信仰
Created at 10-4 18:00, Last updated at 10-4 18:00

同感，好痛苦，刚刚

来旭光
Created at 10-15 13:25, Last updated at 10-15 13:25

看评论想笑哈哈

大好青年陆小司
Created at 10-20 15:29, Last updated at 10-20 15:29

评论很精彩

用户5642643636
Created at 10-21 20:31, Last updated at 10-21 20:31

原来你们都没看到哈　哈哈哈哈哈哈

沉默的跳蛋
Created at 10-23 5:26, Last updated at 10-23 5:26

幸亏我看了评论，谢谢，哈啊哈哈哈哈啊哈哈

挺侹葛格
Created at 10-26 11:32, Last updated at 10-26 11:32

我是在前边章节func.**name**那里发现的，当时也整了半个上午。。。

姜丝号st
Created at 11-2 22:02, Last updated at 11-2 22:02

哈哈，这个下划线，明显过长，前面章节就确认过一遍了

View Full Discuss　Reply This Topic

Python里是不是不能有多个构造函数？

Moooove created at 9-21 10:12, Last updated at 9-21 11:58

今天突然返回来看了一下这节，`__init()__`函数其实就是构造函数吧，然后就想起来java里面是可以有多个构造函数的，我就想试试看，于是我写了

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

class Screen(object):

    def __init__(self, width, height):
        self.width = width
        self.height = height

    def __init__(self):
        self.width = 2880
        self.height = 1800

screen_custom = Screen(2880, 1800)
screen_default = Screen()

print('screen_default: %d * %d' % (screen_default.width, screen_default.height))
print('screen_custom: %d * %d' % (screen_custom.width, screen_custom.height))
```

Read More

风飞浅停

Created at 9-21 11:50, Last updated at 9-21 11:50

python可以一个构造函数顶四个

```
class Screen(object):

    def __init__(self, width=2880, height=1800):
        self.width = width
        self.height = height

screen1 = Screen()
screen2 = Screen(1080)
screen3 = Screen(1400, 900)
screen4 = Screen(height=2100)
```

灰_手

Created at 9-21 11:58, Last updated at 9-21 11:58

python的构造方法灵活到只有你想不到的，没有它做不到的。

```
class Screen(object):

    def __init__(self, **kw):
        self.width = kw.get('width', 2880)
        self.height = kw.get('height', 1800)
        # 你还可以在下面为任何属性设置初默认值
```

```
screen1 = Screen()
screen2 = Screen(width=1080)
screen3 = Screen(width=1400, height=900)
screen4 = Screen(height=2100)
```

View Full Discuss Reply This Topic

不同学生获取分数等级

谁动了我的Code created at 9-8 17:55, Last updated at 9-8 17:55

```
class Student(object):
    def __init__(self,name,age,score):
        self.name = name
        self.age = age
        self.score = score
    def get_level(self):
        if self.score >= 90:
            return self.name +":A"
        elif self.score<90 and self.score>70:
            return  self.name +":B"
        else:
            return self.name +":C"


lisa= Student('lisa',18,91)
jerry = Student('jerry',18,85)
tom = Student('tom',18,45)
print(lisa.get_level())
print(jerry.get_level())
print(tom.get_level())
```

View Full Discuss Reply This Topic

小练习

_Left created at 9-6 21:43, Last updated at 9-6 21:43

```
>>> class man(object):

    def __init__(self,height,weight):
        self.height = height
        self.weight = weight
    def printBMI(self):
        BMI = self.weight / self.height ** 2
        if BMI < 18.5:
            print('体重过低')
        elif 24 > BMI >= 18.5:
            print('正常')
        elif 28 > BMI >= 24:
            print('超重')
        else:
            print('肥胖')
```

```
>>> GIN = man(1.73,56)
>>> GIN.printBMI()
正常
```

View Full Discuss | Reply This Topic

是实例的方法还是类的方法？

破晓江旭 created at 6-8 9:23, Last updated at 9-4 10:26

这些封装数据的函数是和Student类本身是关联起来的，我们称之为类的方法：

```
class Student(object):

    def __init__(self, name, score):
        self.name = name
        self.score = score

    def print_score(self):
        print('%s: %s' % (self.name, self.score))
```

因为学过C#，对比C#, print_score 是实例的方法，并不是类的方法？是吗？

yikedaxibuojjy
Created at 9-3 20:08, Last updated at 9-3 20:08

更像是一种语法糖，你可以这样调用print_socre

```
s = Student('Bob', 99)
Student.print_score(s)
```

通过类名来调用，显式传入了self

```
s.print_socre()
```

直接通过实例来调用,隐式的将s传入到self

yikedaxibuojjy
Created at 9-4 10:26, Last updated at 9-4 10:26

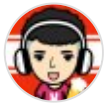
由于python完全没有类型检查，甚至可以这么写

```
class Student(object):
    def __init__(self, name, score):
        self.name = name
        self.score = score

    def print_score(self):
        print(self.score)
```

```
s = Student('Bob', 12)

s.print_score()

Student.print_score(s)
class A_class_have_score_attr:
    def __init__(self, score):
        self.score = score


Student.print_score(A_class_have_score_attr(9999))
```

同样能运行0 0

View Full Discuss | Reply This Topic



一个实操

慢热程某某 created at 7-14 13:54, Last updated at 8-24 23:36

```
class Com(object):

    def __init__(self,num1,num2):
        self.n1=num1
        self.n2=num2
    def compa(co):
        if co.n1>co.n2:
            return str(co.n1)+'大于'+str(co.n2)
        elif co.n1==co.n2:
            return str(co.n1)+'等于'+str(co.n2)
        else:
            return str(co.n1)+'小于'+str(co.n2)

def collect():
    M = []
    M0 = float(input('输入一个数字：'))
    M.append(M0)
    M1 = float(input('再输入一个数字：'))
    M.append(M1)
    return M
```

Read More

早起的达先生

Created at 8-6 15:48, Last updated at 8-6 15:48

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

class Com(object):

    def __init__(self):
        self.n1=float(input('输入一个数字：'))
```

```
        self.n2=float(input('再输入一个数字：'))

    def compa(self):
        if self.n1>self.n2:
            return str(self.n1)+'大于'+str(self.n2)
        elif self.n1==self.n2:
            return str(self.n1)+'等于'+str(self.n2)
        else:
            return str(self.n1)+'小于'+str(self.n2)


x=Com()
print('第一个数字:''%s' % (x.n1)+'\n''第二个数字：''%s' % (x.n2))
print(x.compa())
```

Al0an
Created at 8-6 16:44, Last updated at 8-6 16:44

```
# ! /usr/bin/env python
# -*- coding:utf-8 -*-

class Com:
    def __init__(self,num1,num2):
        self.num1=num1
        self.num2=num2

    def compa(self):
        if self.num1 > self.num2:
            return str(self.num1)+'>'+str(self.num2)
        elif self.num1 == self.num2:
            return str(self.num1)+'='+str(self.num2)
        else:
            return str(self.num1)+'<'+str(self.num2)

M0=input('input a num:')
M1=input('input a num again:')

print('first num:%s' % M0+'\n'+'two num:%s' % M1)
x=Com(M0,M1)
print(x.compa())
```

机修贾森
Created at 8-24 23:36, Last updated at 8-24 23:36

Com类定义部分不变，外部方法：

```
def collect():
    M = []
    M0 = input('first one: ')
    M1 = input('second one: ')
    M.append(M0)
    M.append(M1)
    return M
def compare():
      M = collect()
```

```
        x = Com(M[0],M[1])
        x.compa()

if __name__ == '__main__':
        compare()
```

按照你之前的写法，collect（）就是用来获得用户输入的，如果想写到Com类定义中去，可以参考1楼~

View Full Discuss | Reply This Topic

对象类和实例

珐瑞斯特 created at 8-2 9:47, Last updated at 8-2 9:47

> 类：用户定义的原型对象，它定义了一套描述类的任何对象的属性。属性是数据成员(类变量和实例变量)和方法，通过点符号访问。

类变量：这是一个类的所有实例共享的变量。类变量在类，但外面的任何类的方法定义。类变量不被用作经常作为实例变量。

> 数据成员：保存与类和对象关联的数据的类变量或实例变量。

函数重载：一个以上的行为特定功能的分配。执行的操作所涉及的对象(自变量)的类型不同而不同。

> 实例变量：所定义的方法内，只属于一个类的当前实例的变量。

继承：类的特点，即都是由它派生其他类的转移。

> 实例：某一类的一个单独对象。属于类Circle一个obj对象，例如，是类Circle的一个实例。

实例化：创建一个类的实例。

Read More

View Full Discuss | Reply This Topic

发表评论

Sign In to Make a Comment