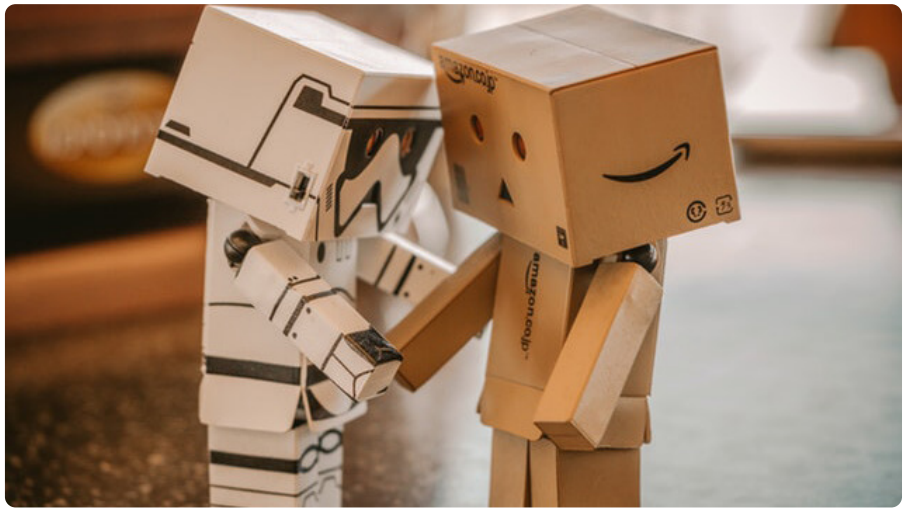
目录

10 定制一个模子—类

更新时间：2019-09-04 10:35:40



> 时间像海绵里的水，只要你愿意挤，总还是有的。
>
> ——鲁迅

查看数据类型

Python 中内置有这么一个函数，通过它可以查看变量或值的数据类型，它就是 type() 。像这样来使用：

```
type(变量或值)
```

执行几个例子看看：

```
>>> type(100)
<class 'int'>
```

```
>>> type(3.14)
<class 'float'>
```

```
>>> type('words')
<class 'str'>
```

目录


```
>>> type(None)
<class ‘NoneType’ >
```

```
>>> type([1, 2, 3])
<class ‘list’ >
```

执行的结果是 `<class '类型'>` 形式，其中类型的含义是：

类型	含义
int	整数型
float	浮点型
str	字符串类型
bool	布尔型
NoneType	None 类型
list	列表类型

> 上表中的这些数据类型，都内置在 Python 中。

那 `<class '类型'>` 中的 `class` 是指什么呢？

类

`class` 是指面向对象编程范式中的一个概念——**类**。Python 中的数据类型就是类，一个类对应一种数据类型。类的具体对象中可以保存若干数据，以及用于操作这些数据的若干函数。

我们来看一个例子：

我们常用的字符串类型，就是名为 `str` 的类。一个 `str` 中可以保存若干字符，并且针对这些字符提供了一系列的操作函数。

如 `'hello'` 就是一个 `str` 对象，我们可以把这个对象赋值给变量：

```
>>> words =  ‘hello’
>>> words
’ hello’
```

`str` 对象自带的 `find()` 函数，可用于获取字符的索引：

```
1
```

str 对象自带的 upper() 函数，可用于获取英文字符的大写形式：

```
>>> words.upper()
 'HELLO'
```

除此 str 之外，前面列表中的那些数据类型也都是类。

类的定义

像 str 、 int 、 list 这样的类，是被预先定义好并且内置在 Python 中的。

当然，我们也可以自己来定义类。

类的定义方法是：

```
class 类名:
    代码块
```

如：

```
class A:
    pass
```

这里定义了一个非常简单的类，名为 A 。 pass 是占位符，表示什么都不做或什么都没有。

类的实例化

我们把类看作是自定义的数据类型，既然是类型，那么它只能用来表示数据的种类，不能直接用于保存数据。想要保存数据，就需要先创建一个属于这种类型的类似于容器的东西，这种容器就叫做**对象**（或称实例）。通过类产生对象的过程叫**实例化**。

打个比方，类就相当于图纸，对象就相当于按照图纸所生产出来的产品。图纸能决定产品的内部构造以及所具有的功能，但图纸不能替代产品被直接使用。类能决定对象能保存什么样的数据，以及能拥有什么样的函数，但类不直接用来保存数据。

定义好类以后，可以像这样实例化对象：

```
变量 = 类名()
```

通过 类名() 这样类似函数调用的方式生成出对象，并将对象赋值给 变量 。

如实例化之前的类 A 并将对象赋值为 a ：

```
...     pass
...
>>> a = A()
```

查看变量 a 的类型：

```
>>> type(a)
<class '__main__.A'>
```

可以看到类型是 __main__.A ，表示模块 __main__ 下的 A 类。模块的概念后续章节中介绍，现在只需关注类即可。

可以看看 a 是什么：

```
>>> a
<__main__.A object at 0x103d8e940>
```

a 是 A 的对象，位于内存的 0x103d8e940 地址。

对象属性

之前定义的 A 类是一个空的类，像一个空壳子，它的对象 a 并没有保存任何数据。

想要在对象中保存数据该怎么做呢？

可以像这样来定义类，实例化的时候就可以用参数的形式将数据传入，并保存在对象中：

```
class 类名:
    def __init__(self, 数据1, 数据2, ...):
        self.数据1 = 数据1
        self.数据2 = 数据2
        ...
```

和之前相比类的内部多了一个函数 __init__() ， __init__() 函数一方面可以接收要保存在对象中的数据，另一方面也可以在实例化类的时候做一些初始化工作。

我们通过实际例子来学习。之前介绍的类（数据类型）要么保存一个数据，要么保存多个数据，假如现在想要一个不多不少只保存两个数据的类，这就需要我们自己来定义了。如下：

```
class Pair:
    def __init__(self, first, second):
        self.first = first
        self.second = second
```

我们将这个类命名为 Pair ，即表示数据对。

1. 第一个是 self ，它是类中函数默认的第一个参数，表示对象自身。我们可以将数据赋值给 self.数据名 ，这样数据就保存在对象中了
2. first 是数据对的第一个值
3. second 是数据对的第二个值

实例化的时候像这样传入数据：

```
pair = Pair(10, 20)
```

这个过程中会自动调用 __init__() 函数，并将 10 传给了 first 参数，将 20 传给了 second 参数，而 __init__() 的第一个参数 self 是不需要传值的，Python 会自动填充这个参数。

实例化之后我们可以通过 pair 对象来获取数据对中的数据，像这样：

```
pair.first
pair.second
```

```
>>> pair = Pair(10, 20)

>>> pair.first
10
>>> pair.second
20
```

通过 pair = Pair(10, 20) 来实例化 Pair 类，得到对象的变量 pair ，使用 pair.first 、 pair.second 就可以获得对象中保存的数据了。

first 和 second 叫做 Pair 类的**对象属性**，一般也可以直接叫作属性。

我们不仅可以通过对象获取对象属性的值，也能修改对象属性值。如：

```
>>> pair = Pair(10, 20)

>>> pair.first = 1000
1000
>>> pair.first
1000
```

对象方法

刚才在类中定义了对象属性，也可以在类中定义一些函数。这样的函数可直接由对象调用，例如我们之前学过的 list.append() 。

定义在类中，供对象调用的函数称为**对象方法**，一般也可以直接叫作方法。定义方式如下：

```
    ...
```

定义对象方法时第一个参数默认使用 self ，定义时必须有这个参数，但是调用时不必传递。之前介绍过的 __init__() 就是一个对象方法，不过是个特殊的对象方法。

我们在之前 Pair 类的基础上定义一个方法，功能是交换对象的 first 和 second 属性的值。来实现一下：

```
class Pair:
    def __init__(self, first, second):
        self.first = first
        self.second = second

    def swap(self):
        self.first, self.second = self.second, self.first
```

这个方法被命名为 swap ，无需传递参数，内部通过

```
self.first, self.second = self.second, self.first
```

实现了 self.first 和 self.second 两个值的交换。

执行下看看：

```
>>> pair = Pair(10, 20)
>>> pair.first
10
>>> pair.second
20

>>> pair.swap()

>>> pair.first
20
>>> pair.second
10
```

总结

定义类的方式是：

```
class 类名:
    代码块
```

在类中定义方法：

```
class 类名:
    def 方法(self, 参数1, ...):
```

可以在 __init__ 方法中定义对象属性，之后在实例化类的时候传入数据。如：

```
class Pair:
    def __init__(self, first, second):
        self.first = first
        self.second = second

pair = Pair(10, 20)
```

← 09 知错能改—错误处理、异常机制

11 更大的代码盒子—模块和包 →

精选留言 3

欢迎在这里发表留言，作者筛选后可公开显示

慕娘0558415

定义类时，代码块中定义了两个函数，在实例化时传入的数据（10，20），程序如何判断是传入第一个函数，而不是传入第二个函数？另外程序执行第二个函数时是用pair.函数2，那如果类中还定义了其它函数，执行其它函数时是不是应该时pair.函数3、pair.函数4...

0 回复 2019-12-01

瘦削的猪

老师，在使用pycharm运行示例时，调用swap方法时，提示缺少self参数，在括号中填写实例化的对象后，才正常运行，请问这个是什么原因呢？

0 回复 2019-11-04

黄浮云 回复 **瘦削的猪**

我猜你可能是把 swap(self) 定义到了类外面，导致 swap(self) 变成了普通函数，而不是对象方法

回复 2019-11-06 22:40:26

WangFitZ

__init__()的对象方法有什么特殊之处？

0 回复 2019-09-30

黄浮云 回复 **WangFitZ**

这个方法主要用作对象的初始化，在执行「对象 = 类名()」的时候，Python 会在创建对象的过程中自动调用 __init__() 方法

回复 2019-11-06 22:43:43

目录