

Objects and Classes

巧克力商店的例子见上课内容. 对于一种巧克力, 我们可以用函数做数据抽象.

之前的多个作业就是这么干的 (天文台的行星模型)

当然, 我们可以从 ^{objects} 对象的角度考虑问题

对象由数据和操作组成

对于巧克力 数据可以是 库存、名称、价格...

操作可以是 更改价格、增加库存...

classes

类 是定义新数据类型的模板

instance

类的实例称为对象.

每个对象都有 1) 数据属性, 称为描述其状态的 ^{instance variables} 实例变量

2) 函数属性, 称为方法.

现在让我们着手创建一个类. 以巧克力为例.

```
class Chocolate:  
    # Set the initial values  
    def __init__(self, name, price, nutrition):  
        self._name=name  
        self._price=price  
        self._nutrition=nutrition  
    # Define methods
```

pina_bar=Chocolate('Pina Chocolate', 7.99, ['24 g sugar'])

调用 Chocolate() 时自动调用 __init__ 函数进行赋值.

创建方法与定义函数类似. 大部分时候变量都包含 self.

执行完 __init__ 后

所有数据均保存于全局帧中

Class instantiation

上面标紫的一段代码即为 ^{constructor} 类实例化

Chocolate(*args) 通常被称为构造函数, 因为它构造了一个新对象.

调用类的数据或方法的语句是 . (dot notation)

例如 pina_bar._name

pina_bar.increase_inventory(2)

```
class Chocolate:  
    #Set the initial values  
    def __init__(self, name, price, nutrition):  
        self._name=name  
        self._price=price  
        self._nutrition=nutrition  
        self._inventory=0  
    #Define methods  
    def increase_inventory(self, amount):  
        self._inventory+=amount  
  
pina_bar=Chocolate('Pina Chocolate', 7.99, ['24 g sugar'])  
pina_bar.increase_inventory(2)
```

pina_bar.increase_inventory 是一个 ^{bound method} 绑定方法
调用时, 第一个参数 self ^{pre-bound} 预先绑定了 pina_bar, 所以看起来只传递了一个参数
在实例上调用方法都会将该方法与实例预先绑定

^{class variables}
在类中可以定义类变量, 它们对于属于该类的所有对象都适用。
可以通过 <class name>.<class variable> = ... 对其赋值。