

Object-Oriented Programming in Python

Python 面向对象编程

OOP 听起来很玄乎。其实就像个筐,什么都能装



机会总是青睐做好准备的人。

Chance favors the prepared mind.

—— 路易·巴斯德 (Louis Pasteur) | 法国微生物学家、化学家 | 1822 ~ 1895



- ◆ class 定义一个类, 类是一种数据结构, 包含属性和方法, 用于创建实例对象
- ◀ def __init__() 用于初始化对象的属性,在对象创建时自动调用
- ◀ self 表示当前对象的引用,用于访问对象的属性和调用对象的方法
- ◀ @property 装饰器,将方法转换为属性,使得方法像属性一样访问
- ◀ @classmethod 装饰器,将方法定义为类方法,而不是实例方法
- ◀ cls 用于访问类的属性和调用类的方法
- ◀ super().__init__() 调用父类的构造方法,用于在子类的构造方法中初始化父类的属性



9.1 什么是面向对象编程?

本章蜻蜓点水介绍面向对象编程基本用法。对于大部分读者来说,本章可以跳过不读。如果对面向 对象编程感兴趣的话,请继续阅读本章。

面向对象编程 (Object-Oriented Programming, OOP) 是一种编程范式,它将数据和操作数据的方法组合在一起,形成一个对象。在面向对象编程中,一个对象拥有一组属性 (用来描述对象的特征) 和方法 (用来设定对象的行为)。对象可以与其他对象互动,实现特定的功能。面向对象编程强调封装、继承和多态等概念,使程序更易于维护和扩展。

在 Python 中,一切皆为对象,可以通过 class 关键字来定义一个类,类中可以包含属性和方法,然后通过实例化对象来使用类中的属性和方法。

打个比方, OOP 中的类 (class) 就好比图 1 中的成套餐具, 相当一种模板。盘子好比属性 (attribute), 用来装各种食物 (数据); 刀叉好比方法 (method), 用来用餐 (操作)。而实例 (instance) 则相当于一个个具体的套餐, 盘中餐可以是凉菜、炒饭、炒面等等。

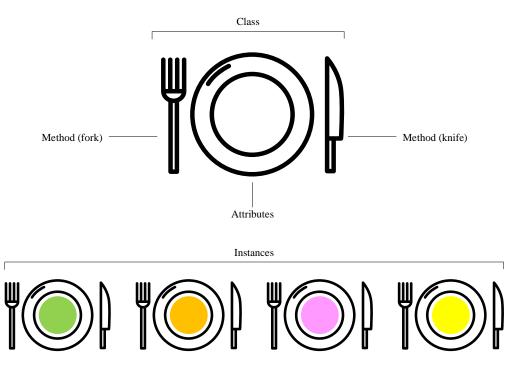


图 1. 面向对象编程中的属性、方法

图 2 这段代码定义了一个名为 Rectangle 的类,它具有构造函数来初始化矩形的宽度和高度,并提供了两个方法来计算矩形的周长和面积。



图 2. 定义、使用"矩形"类

下面详细介绍图2代码。

- ②定义了一个矩形类,名称为 Rectangle。Rectangle 有两个属性 width 和 height。类是一个代码模板,用于创建具有相似属性和行为的对象。Rectangle 有两个方法: circumference (计算周长)、area (计算面积)。
- ¹ 中关键字 class 是用来创建对象的模板,它是面向对象编程的基础。关键词 class 把数据 (属性) 和操作 (方法) 封装起来,这样便于代码模块化,方便维护。此外,类之间可以通过继承机制建立关系,本章后面将介绍。
- ©中__init__(self,...) 方法是 Python 中的一个特殊构造方法,用于在创建类的实例时进行初始化操作。

在 init 方法的参数列表中,第一个参数通常被命名为 self, 它指向类的实例对象。

注意, __中有两个半角下划线_ (underscore); init 四个字母均为小写字母; self 四个字母也均为小写字母。

self 参数在调用类的其他方法时自动传递,可以通过 self 访问类的属性和其他方法。在 __init__方法内部,可以定义初始化对象时需要执行的逻辑,例如设置对象的初始状态,为对象设置 属性的初始值等。

- ¹ 用 def 定义了 circumference()这个方法,用来计算矩形周长,并用 return 返回计算结果。
 - ◉用 def 定义了 area()这个方法,用来计算矩形面积,并用 return 返回计算结果。

本 PDF 文件为作者草稿,发布目的为方便读者在移动终端学习,终稿内容以清华大学出版社纸质出版物为准。版权归清华大学出版社所有,请勿商用,引用请注明出处。 代码及 PDF 文件下载: https://github.com/Visualize-ML 本书配套微课视频均发布在 B 站——生姜 DrGinger: https://space.bilibili.com/513194466 欢迎大家批评指教,本书专属邮箱: jiang.visualize.ml@gmail.com

● 调用了自定义的 Rectangle 对象,将其命名为 rect_width_5_height_10。输入的参数为:矩形宽度 5、矩形高度 10。

大家练习时,利用 rect width 5 height 10.width 打印矩形宽度。

注意, 调用属性时不加圆括号 ()。

然后, rect_width_5_height_10.circumference() 调用矩形对象的 circumference() 方法计算这个矩形的周长。rect_width_5_height_10.area() 调用矩形对象的 area() 方法计算面积。

注意, 使用方法时需要圆括号 ()。

请大家自行练习图2代码,使用 Rectangle 定义宽度为6、高度为8的矩形对象,并计算矩形的周长、面积。

9.2 定义属性

在图 3 代码 ³ 中,我们定义一个叫 Chicken 的类,这个类有以下属性: (1) name (名字); (2) age (鸡龄); (3) color (毛色); (4) weight (体重)。

图 3 代码中 [©] 使用__init__方法来初始化 Chicken 这个类的属性。

接下来,图3创建一只名为"小红"的黄色小鸡,命名为 chicken_01;然后,创建了一只名为"小黄"的红色色小鸡,命名为 chicken_02。请大家在练习的时候,也打印 chicken_02 的属性。

此外,在后续代码中还可以覆盖对象属性。比如,如果对象 chicken_01 的年龄写错,也可以用 chicken_01.age = 5 覆盖。

图 4 中也定了 Chicken 类,图 4 和图 3 的代码的最大不同的是图 4 中在定义 Chicken 类时给 color、weight 两个属性默认值。

图 4 代码 ^② 调用 Chicken 类时,覆盖了默认毛色,但是保留体重默认值。

注意,图 3 中定义的 Chicken类,不能通过 chicken_01 = Chicken() 直接定义一个实例。会产生如下错误。

TypeError: Chicken.__init__() missing 4 required positional arguments:
'name', 'age', 'color', and 'weight'

将图 3 改成图 5 后,在 ⁶ 中利用 Chicken 类创建实例 chicken 01 时不需要赋值。

然后,如[●]所示,再对 chicken 01 的每个属性分别赋值。

```
# 创建了一个名为 "Chicken" 的类
class Chicken:
    def __init__(self, name, age, color, weight):
    # 初始化对象的属性
       # 设置实例变量self.name来存储小鸡名字
       self.name = name
       # 设置实例变量self.age来存储小鸡年龄
       self.age = age
                                                       Attributes
       # 设置实例变量self.color来存储小鸡体色

    self name

                                                       • self.age
       self.color = color

    self.color

       # 设置实例变量self.weight来存储小鸡体重
                                                       • self.weight
       self.weight = weight
# 调用Chicken类
chicken_01 = Chicken("小红", 1, "黄色", 1.5)
chicken_02 = Chicken("小黄", 1.2, "红色", 2)
print('==小鸡的名字=='); print(chicken_01.name)
print('==小鸡的年龄, yr=='); print(chicken_01.age)
print('==小鸡的颜色=='); print(chicken_01.color)
print('==小鸡的体重, kg=='); print(chicken_01.weight)
```

图 3. 定义、使用"鸡"类



图 4. 定义、使用"鸡"类,设置默认参数变量 (color, weight)

```
# 创建了一个名为 "Chicken" 的类
 class Chicken:
    def __init__(self):
    # 初始化对象的属性
        # 设置实例变量self.name来存储小鸡名字的参数
        self.name = ''
        # 设置实例变量self.age来存储小鸡名字的年龄
        self.age = ''
                                                       Attributes
        # 设置实例变量self.color来存储小鸡名字的颜色
                                                       • self.name
                                                       • self.age
        self.color = ''
                                                       self.color
        # 设置实例变量self.weight来存储小鸡名字的体重
                                                       • self.weight
        self.weight = ''
 # 调用Chicken类, 然后赋值
chicken_01 = Chicken()
 chicken_01.name = '小红'
 chicken_01.age = 1
 chicken_01.color = '黄色'
 chicken_01.weight = 1.5
```

图 5. 定义、使用"鸡"类,创建实例时不需要参数

9.3 定义方法

图 6 给出一个例子,代码 ^② 定义一个 ListStatistics 类来计算一个浮点数列表的长度、和、平均值、 方差。

- 仓定义的 list_mean() 方法计算平均值时用到了 list_length() 方法。
- ②定义的 list_variance() 方法还有一个输入 ddof, ddof 默认值为 1。
- 调用 ListStatistics 类创建对象。
- ①计算两个方差;第一个方差相当于粽子方差。第二个方差相当于样本无偏方差。

此外,我们在第4章介绍过,Python 变量名一般采用蛇形命名法,比如 list_mean(); Python 面向对象编程中的类定义一般采用驼峰命名法,比如 ListStatistics。

```
# 创建 ListStatistics 类
b
   class ListStatistics:
       # 构造函数,用于初始化属性
       def __init__(self, data):
           # ListStatistics包含一个data属性来存储浮点数列表
           self.data = data
                                                          Attribute:
       # 下面定义了4个方法

    self data

       # 方法1: 计算列表的长度, 即元素的数量
       def list_length(self):
                                                          Methods:
           return len(self.data)
                                                          • self.list_length()
                                                          self.list_sum()
       # 方法2: 计算列表元素之和
       def list_sum(self):
                                                          self.list_mean()
                                                          • self.list_variance()
           return sum(self.data)
       # 方法3: 计算列表元素平均值
       def list_mean(self):
    return sum(self.data)/self.list_length()
       # 方法4: 计算列表元素方差
       def list_variance(self, ddof = 1):
# Delta自由度 ddof 默认为 1; 无偏样本方差
           return sum_squares/(self.list_length() - ddof)
   # 创建一个浮点数列表
   data = [8.8, 1.8, 7.8, 3.8, 2.8, 5.6, 3.9, 6.9]
   # 创建ListStatistics对象实例
float_list = ListStatistics(data)
   # 使用float_list对象计算列表长度
   print("列表长度: ", float_list.list_length())
   # 使用float_list对象计算列表和
   print("列表和: ", float_list.list_sum())
   # 使用float_list对象计算列表平均值
   print("列表平均值: ", float_list.list_mean())
# 使用float_list对象计算列表方差
   print("列表方差: ", float_list.list_variance())
print("列表方差 (ddof = 0): ",
         float_list.list_variance(0))
```

图 6. 定义、使用"列表统计量"类

9.4 装饰器

在 Python 中,装饰器 (decorator) 是一种特殊的语法,用于在不修改函数代码的情况下,为函数添加额外的功能或修改函数的行为。

如图 7 所示, ¹ 中<mark>装饰器 @property 用于将一个方法转换为只读属性</mark>,可以像访问属性一样访问该方法,而无需使用括号调用它。

①中装饰器 @data.setter 装饰器用于在 @property 装饰的方法后定义一个 setter 方法,这样可以在设置属性时执行一些逻辑或验证,对属性的赋值进行控制。

```
本 PDF 文件为作者草稿,发布目的为方便读者在移动终端学习,终稿内容以清华大学出版社纸质出版物为准。版权归清华大学出版社所有,请勿商用,引用请注明出处。
代码及 PDF 文件下载: https://github.com/Visualize-ML
本书配套微课视频均发布在 B 站——生姜 DrGinger: https://space.bilibili.com/513194466
欢迎大家批评指教,本书专属邮箱: jiang.visualize.ml@gmail.com
```

- 中装饰器 @classmethod 装饰器用于定义类方法。类方法是在类上而不是在实例上调用的方法。不同于 self,类方法的第一个参数通常被命名为 cls,它表示类本身而不是实例简单来说,cls 是一个约定俗成的名字,表示类本身,而不是类的实例。
 - ●用于逐个判断一个列表中的所有元素是否都是数值,比如 float 或 int 类型。
- 创建了 ListStatistics 类的实例,命名为 float_list_obj。由于 data 中有一个非数值元素,在 [™] 赋值 时会报错。

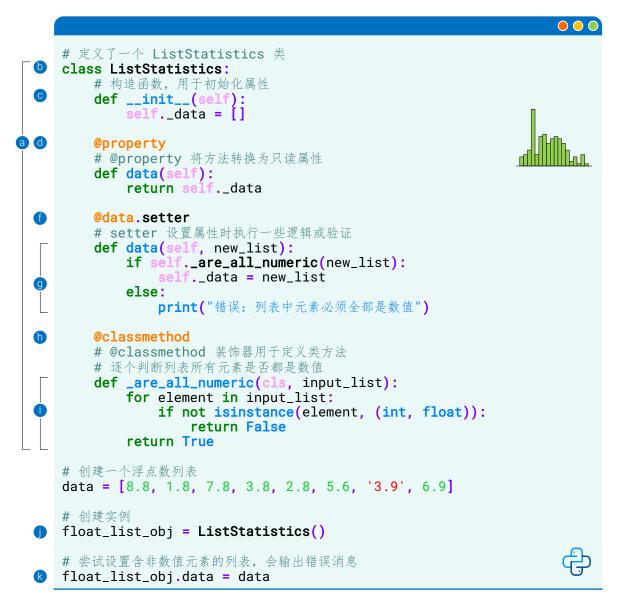


图 7. 定义、使用"列表统计量"类,使用装饰器

9.5 父类、子类

在面向对象编程中,父类 (parent class) 和子类 (child class) 之间是一种继承关系。父类,也称基类、超类,在继承关系中层次更高;子类,也称派生类,可以继承父类的属性和方法,从而实现代码的重用和扩展。子类可以有多个,并且一个子类也可以再被其他类继承,形成继承的层级结构。

简单来说,父类提供了一个通用模板。如图8所示,盘子+刀叉,这个组合就相当于父类。而午餐、晚餐一方面继承了"盘子+刀叉",并在此基础上进行了扩展和订制。

午餐的餐具组合为: 父类(盘子+刀叉)+碗; 晚餐的餐具组合为: 父类(盘子+刀叉)+酒杯。

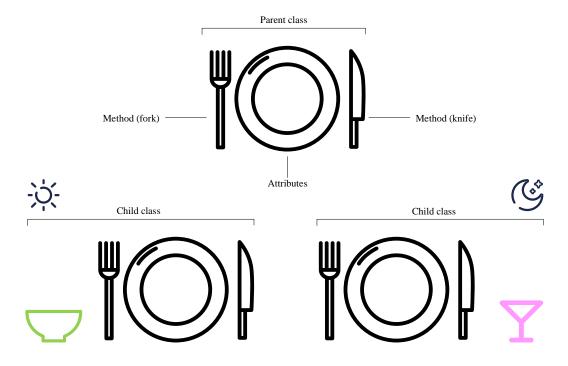


图 8. 面向对象编程中, 父类、子类关系

图 9 代码演示了如何定义父类 Animal 和子类 Chicken、Rabbit、Pig。

首先. ② 定义了一个 Animal 父类。

- ●定义了 Animal 的两个属性——名字、年龄; Animal 有两个方法——吃饭 ()、睡觉 ()。
- **b o d** 分别定义了三个子类 Chicken、Rabbit、Pig。它们分别继承了父类 Animal 的属性和方法,并且分别定义了自己的属性和方法。

当一个类继承自另一个类时,子类可以通过 super().__init__() 来调用父类的构造方法,以便在实例 化子类时,也能初始化从父类继承的属性。

比如, ¹ 定义了 Chicken 类专属属性 color,表示鸡的颜色。

● 定义了 Chicken 类专属方法 lay_egg,表示鸡下蛋。Rabbit、Pig 也有各自的专属属性和方法。

```
# 父类, 动物
class Animal:
              _init__(self, name, age):
        def
             self.name = name
             self.age = age
        def eat(self):
             print(f"{self.name} is eating.")
        def sleep(self);
g
             print(f"{self.name} is sleeping.")
    class Chicken(Animal):
             __init__(self, name, age, color):
super():__init__(name, age)
        def
             self.color = color
        def lay_egg(self):
    print(f"{self.name} is laying an egg.")
    class Rabbit(Animal);
             __init__(self, name, age, speed):
        def
             super().__init__(name, age)
             self.speed = speed
        def jump(self):
             print(f"{self.name} is jumping.")
    # 子类,猪
    class Pig(Animal):
             __init__(self, name, age, weight):
super().__init__(name, age)
        def
             self.weight = weight
        def roll(self):
             print(f"{self.name} is rolling around.")
    chicken1 = Chicken("chicken1", 1,
                                          "white")
    chicken1.eat(); chicken1.lay_egg()
    rabbit1 = Rabbit("rabbit1", 2, 10)
    rabbit1.sleep(); rabbit1.jump()
    pig1 = Pig("pig1", 3, 100)
pig1.eat(); pig1.roll()
```

图 9. 定义、使用父类 (动物)、子类 (鸡、兔、猪)



请大家完成下面2道题目。

- Q1. 参考图 2,写一个名为 Circle 的类,参数为半径,定义两个方法分别计算圆的周长、面积。提示,需要导入 math.pi 圆周率近似值。
- Q2. 请大家在练习图 6代码时,再增加 4 个方法,分别计算最大值、最小值、极差 (最大值 最小值)、标准差。

* 两道题目很简单,本书不提供答案。

本 PDF 文件为作者草稿,发布目的为方便读者在移动终端学习,终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有, 请勿商用, 引用请注明出处。

代码及 PDF 文件下载: https://github.com/Visualize-ML

本书配套微课视频均发布在 B 站——生姜 DrGinger: https://space.bilibili.com/513194466

欢迎大家批评指教,本书专属邮箱: jiang.visualize.ml@gmail.com

本章只是 Python 面向对象编程 OOP 冰山一角,希望大家在需要用到 OOP 时深入学习。