

# 实验七 Python面向对象编程

---

班级： 21计科02班

学号： B20230302210

姓名： 陶鑫

Github地址： <https://github.com/Muyu-ikun>

CodeWars地址： <https://www.codewars.com/users/Muyu-ikun>

---

## 实验目的

1. 学习Python类和继承的基础知识
2. 学习namedtuple和DataClass的使用

## 实验环境

1. Git
2. Python 3.10
3. VSCode
4. VSCode插件

## 实验内容和步骤

### 第一部分

Python面向对象编程

完成教材《Python编程从入门到实践》下列章节的练习：

- 第9章 类
- 

### 第二部分

在[Codewars网站](#)注册账号，完成下列Kata挑战：

---

#### 第一题：面向对象的海盗

难度： 8kyu

啊哈，伙计！

你是一个小海盗团的首领。而且你有一个计划。在OOP的帮助下，你希望建立一个相当有效的系统来识别船上有大量战利品的船只。对你来说，不幸的是，现在的人很重，那么你怎么知道一艘船上装的是黄金而不是人呢？

你首先要写一个通用的船舶类。

```
class Ship:
    def __init__(self, draft, crew):
        self.draft = draft
        self.crew = crew
```

每当你的间谍看到一艘新船进入码头，他们将根据观察结果创建一个新的船舶对象。

- **draft**吃水 - 根据船在水中的高度来估计它的重量
- **crew**船员 - 船上船员的数量

```
Titanic = Ship(15, 10)
```

## 任务

你可以访问船舶的 "draft(吃水)" 和 "crew(船员)"。"draft(吃水)" 是船的总重量，"船员" 是船上的人数。每个船员都会给船的吃水增加1.5个单位。如果除去船员的重量后，吃水仍然超过20，那么这艘船就值得掠夺。任何有这么重的船一定有很多战利品! 添加方法 **is\_worth\_it** 来决定这艘船是否值得掠夺。

例如：

```
Titanic.is_worth_it()
False
```

祝你好运，愿你能找到金子!

代码提交地址：<https://www.codewars.com/kata/54fe05c4762e2e3047000add>

```
class Ship:
    def __init__(self, draft, crew):
        self.draft = draft
        self.crew = crew

    def is_worth_it(self):
        # 计算除去船员重量后的吃水深度
        effective_draft = self.draft - (self.crew * 1.5)

        # 判断船是否值得掠夺
        return effective_draft > 20
```

---

## 第二题：搭建积木

难度：7kyu

写一个创建Block的类（Duh.） 构造函数应该接受一个数组作为参数，这个数组将包含3个整数，其形式为 `[width, length, height]`，Block应该由这些整数创建。

定义这些方法:

- `get_width()` return the width of the `Block`
- `get_length()` return the length of the `Block`
- `get_height()` return the height of the `Block`
- `get_volume()` return the volume of the `Block`
- `get_surface_area()` return the surface area of the `Block`

例子:

```
b = Block([2,4,6]) # create a `Block` object with a width of `2` a length of `4`
and a height of `6`
b.get_width() # return 2
b.get_length() # return 4
b.get_height() # return 6
b.get_volume() # return 48
b.get_surface_area() # return 88
```

注意： 不需要检查错误的参数。

代码提交地址: <https://www.codewars.com/kata/55b75fcf67e558d3750000a3>

```
class Block:
    def __init__(self, dimensions):
        self.width, self.length, self.height = dimensions

    def get_width(self):
        return self.width

    def get_length(self):
        return self.length

    def get_height(self):
        return self.height

    def get_volume(self):
        return self.width * self.length * self.height

    def get_surface_area(self):
        return 2 * (self.width * self.length + self.length * self.height +
self.height * self.width)
```

难度: 5kyu

在这个练习中，你将加强对分页的掌握。你将完成PaginationHelper类，这是一个实用类，有助于查询与数组有关的分页信息。该类被设计成接收一个值的数组和一个整数，表示每页允许多少个项目。集合/数组中包含的值的类型并不相关。

下面是一些关于如何使用这个类的例子：

```
helper = PaginationHelper(['a','b','c','d','e','f'], 4)
helper.page_count() # should == 2
helper.item_count() # should == 6
helper.page_item_count(0) # should == 4
helper.page_item_count(1) # last page - should == 2
helper.page_item_count(2) # should == -1 since the page is invalid

# page_index takes an item index and returns the page that it belongs on
helper.page_index(5) # should == 1 (zero based index)
helper.page_index(2) # should == 0
helper.page_index(20) # should == -1
helper.page_index(-10) # should == -1 because negative indexes are invalid
```

代码提交地址: <https://www.codewars.com/kata/515bb423de843ea99400000a>

```
class PaginationHelper:
    def __init__(self, collection, items_per_page):
        self.collection = collection
        self.items_per_page = items_per_page

    def item_count(self):
        return len(self.collection)

    def page_count(self):
        return (len(self.collection) + self.items_per_page - 1) // self.items_per_page

    def page_item_count(self, page_index):
        if page_index < 0 or page_index >= self.page_count():
            return -1
        elif page_index == self.page_count() - 1:
            # 如果是6%4, 那么最后一页就是2
            # 如果是8%4, 那么最后一页就是0, 说明最后一页是满的, 应该返回4
            last_page = self.item_count() % self.items_per_page

            return self.items_per_page if last_page == 0 else last_page

        # 其他页
        else:
            return self.items_per_page

    def page_index(self, item_index):
```

```
if item_index < 0 or item_index >= len(self.collection):  
    return -1  
return item_index // self.items_per_page
```

---

## 第四题：向量 (Vector) 类

难度：5kyu

创建一个支持加法、减法、点积和向量长度的向量 (Vector) 类。

举例来说：

```
a = Vector([1, 2, 3])  
b = Vector([3, 4, 5])  
c = Vector([5, 6, 7, 8])  
  
a.add(b)      # should return a new Vector([4, 6, 8])  
a.subtract(b) # should return a new Vector([-2, -2, -2])  
a.dot(b)      # should return 1*3 + 2*4 + 3*5 = 26  
a.norm()      # should return sqrt(1^2 + 2^2 + 3^2) = sqrt(14)  
a.add(c)      # raises an exception
```

如果你试图对两个不同长度的向量进行加减或点积，你必须抛出一个错误。向量类还应该提供：

- 一个 `__str__` 方法，这样 `str(a) === '(1,2,3)'`
- 一个 `equals` 方法，用来检查两个具有相同成分的向量是否相等。

注意：测试案例将利用用户提供的 `equals` 方法。

代码提交地址： <https://www.codewars.com/kata/526dad7f8c0eb5c4640000a4>

```
import math  
  
class Vector:  
    def __init__(self, components):  
        self.components = components  
  
    def __str__(self):  
        return f"({'',}'.join(map(str, self.components))})"  
  
    def equals(self, other):  
        return self.components == other.components  
  
    def add(self, other):  
        if len(self.components) != len(other.components):  
            raise ValueError("Vector dimensions must match for addition")
```

```
        result = [x + y for x, y in zip(self.components, other.components)]
        return Vector(result)

    def subtract(self, other):
        if len(self.components) != len(other.components):
            raise ValueError("Vector dimensions must match for subtraction")
        result = [x - y for x, y in zip(self.components, other.components)]
        return Vector(result)

    def dot(self, other):
        if len(self.components) != len(other.components):
            raise ValueError("Vector dimensions must match for dot product")
        result = sum(x * y for x, y in zip(self.components, other.components))
        return result

    def norm(self):
        return math.sqrt(sum(x ** 2 for x in self.components))
```

---

## 第五题：Codewars风格的等级系统

难度：4kyu

编写一个名为User的类，用于计算用户在类似于Codewars使用的排名系统中的进步量。

业务规则：

- 一个用户从等级-8开始，可以一直进步到8。
- 没有0（零）等级。在-1之后的下一个等级是1。
- 用户将完成活动。这些活动也有等级。
- 每当用户完成一个有等级的活动，用户的等级进度就会根据活动的等级进行更新。
- 完成活动获得的进度是相对于用户当前的等级与活动的等级而言的。
- 用户的等级进度从零开始，每当进度达到100时，用户的等级就会升级到下一个等级。
- 在上一等级时获得的任何剩余进度都将被应用于下一等级的进度（我们不会丢弃任何进度）。例外的情況是，如果没有其他等级的进展（一旦你达到8级，就没有更多的进展了）。
- 一个用户不能超过8级。
- 唯一可接受的等级值范围是-8,-7,-6,-5,-4,-3,-2,-1,1,2,3,4,5,6,7,8。任何其他值都应该引起错误。

逻辑案例：

- 如果一个排名为-8的用户完成了一个排名为-7的活动，他们将获得10的进度。
- 如果一个排名为-8的用户完成了排名为-6的活动，他们将获得40的进展。
- 如果一个排名为-8的用户完成了排名为-5的活动，他们将获得90的进展。
- 如果一个排名-8的用户完成了排名-4的活动，他们将获得160个进度，从而使该用户升级到排名-7，并获得60个进度以获得下一个排名。
- 如果一个等级为-1的用户完成了一个等级为1的活动，他们将获得10个进度（记住，零等级会被忽略）。

代码案例：

```
user = User()
user.rank # => -8
user.progress # => 0
user.inc_progress(-7)
user.progress # => 10
user.inc_progress(-5) # will add 90 progress
user.progress # => 0 # progress is now zero
user.rank # => -7 # rank was upgraded to -7
```

代码提交地址: <https://www.codewars.com/kata/51fda2d95d6efda45e00004e>

---

## 第三部分

使用Mermaid绘制程序的类图

安装VSCode插件:

- Markdown Preview Mermaid Support
- Mermaid Markdown Syntax Highlighting

使用Markdown语法绘制你的程序绘制程序类图（至少一个），Markdown代码如下:

显示效果如下:

```
classDiagram
    class Ship {
        + draft: float
        + crew: int
        --
        + __init__(draft: float, crew: int)
        + is_worth_it(): bool
    }
```

查看Mermaid类图的语法-->[点击这里](#)

## 实验考查

请使用自己的语言并使用尽量简短代码示例回答下面的问题，这些问题将在实验检查时用于提问和答辩以及实际的操作。

1. Python的类中\_\_init\_\_方法起什么作用？

初始化对象的属性：**init** 方法允许您在创建类的实例时设置对象的属性。您可以在\_\_init\_\_方法中定义类的属性，并为这些属性赋予初始值。

执行初始化操作：您可以在\_\_init\_\_方法中执行与对象相关的初始化操作。这些操作可以包括计算、配置、打开文件等。

接受参数：**init** 方法可以接受参数，这些参数可以在创建对象时传递给构造方法，以便在初始化对象时使用。

## 2. Python语言中如何继承父类和改写（override）父类的方法。

创建子类：首先，定义一个子类，它继承自父类。子类的定义应包括子类的名称和要继承的父类名称，例如：

定义方法：在子类中定义一个与父类方法名称相同的方法。这将是改写（override）的方法。方法的定义可以包含子类特定的逻辑。

使用 `super()`：在子类方法中，如果希望保留父类的功能并扩展它，可以使用 `super()` 函数来调用父类的方法。这允许子类在执行父类方法的基础上添加额外的功能。

## 3. Python类有那些特殊的方法？它们的作用是什么？请举三个例子并编写简单的代码说明。

在Python中，类可以定义一些特殊的方法（也称为魔术方法或魔法方法），它们具有特定的名称和功能，用于控制类的行为。以下是一些常见的特殊方法以及它们的作用：

1. `__init__(self, ...)`: 这是类的构造方法，用于初始化对象的属性。它在创建对象时自动调用，并允许您设置对象的初始状态。

```
class MyClass:
    def __init__(self, name):
        self.name = name

obj = MyClass("Alice")
print(obj.name) # 输出 "Alice"
```

2. `__str__(self)`: 这个方法返回对象的字符串表示形式，通常用于自定义对象的可读性。

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return f"Person: {self.name}, {self.age} years old"

person = Person("Alice", 30)
print(person) # 输出 "Person: Alice, 30 years old"
```

3. `__add__(self, other)`: 这个方法允许您定义对象的加法操作。它允许您在自定义类中定义加法操作的行为。

```
class Vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y
```



```
def __add__(self, other):
    new_x = self.x + other.x
    new_y = self.y + other.y
    return Vector(new_x, new_y)

v1 = Vector(1, 2)
v2 = Vector(3, 4)
result = v1 + v2
print(result.x, result.y) # 输出 "4 6"
```

## 实验总结

总结一下这次实验你学习和使用到的知识，例如：编程工具的使用、数据结构、程序语言的语法、算法、编程技巧、编程思想。Python类和对象：我学会了如何创建Python类和实例化对象，以及如何定义类的属性和方法。

构造方法（**init**）：我了解了构造方法的作用，它用于初始化对象的属性，并在创建对象时自动调用。