

实验七 Python面向对象编程

班级： 21计科2

学号： B20210202314

姓名： 朱华畅

Github地址： https://github.com/Lakzhu/python_project

CodeWars地址： <https://www.codewars.com/users/Lak朱>

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>

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>

第三题：分页助手

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>

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>

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）。例外情况是，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:

 程序类图

显示效果如下:

```

---
title: Animal example
---
classDiagram
    note "From Duck till Zebra"
    Animal <|-- Duck
    note for Duck "can fly\ncan swim\ncan dive\ncan help in debugging"
    Animal <|-- Fish
    Animal <|-- Zebra
    Animal : +int age
    Animal : +String gender
    Animal: +isMammal()
    Animal: +mate()
    class Duck{
        +String beakColor
        +swim()
        +quack()
    }
    class Fish{
        -int sizeInFeet
        -canEat()
    }
    class Zebra{
        +bool is_wild

```

```
+run()  
}
```

查看Mermaid类图的语法-->[点击这里](#)

使用Markdown编辑器（例如VScode）编写本次实验的实验报告，包括[实验过程与结果](#)、[实验考查](#)和[实验总结](#)，并将其导出为 **PDF格式** 来提交。

实验过程与结果

请将实验过程与结果放在这里，包括：

- [第一部分 Python面向对象编程](#)
- [第二部分 Codewars Kata挑战](#)
- [第三部分 使用Mermaid绘制程序流程图](#)

第一题

```
class Ship:  
    def __init__(self, draft, crew):  
        self.draft = draft  
        self.crew = crew  
    def is_worth_it(self):  
        return self.draft - self.crew * 1.5 > 20
```

The screenshot shows a Codewars kata interface for 'OOP: Object Oriented Piracy' (8 kyu). The problem description includes a difficulty level of 8 kyu, 280 stars, 97 likes, and 88% of 1,844 users solved it. The solution is written in Python 3.11. The test results show 103 tests passed and 0 failed. The sample tests provided are:

```
1 EmptyShip = Ship(0, 0)  
2 test.assertEqual(EmptyShip.is_worth_it(), False)
```

第二题

```
class Block:  
    def __init__(self, block):
```

```

    self.w = block[0]
    self.l = block[1]
    self.h = block[2]
def get_width(self):
    return self.w
def get_length(self):
    return self.l
def get_height(self):
    return self.h
def get_volume(self):
    return self.w * self.h * self.l
def get_surface_area(self):
    return (self.w * self.l + self.w * self.h + self.h * self.l) * 2

```

Heads Up: Unlock the full potential of your account - confirm your email to enable the full doc

7 kyu Building blocks

☆ 154 57 93% of 1,254 2,753 of 10,219 NaMe613

3 Issues Reported

Instructions Output

Block with dimensions [9, 9, 9] (5 of 5 Assertions)

Block with dimensions [10, 10, 10] (5 of 5 Assertions)

Random Tests

- Block with dimensions [1, 7, 96] (5 of 5 Assertions)
- Block with dimensions [21, 99, 15] (5 of 5 Assertions)
- Block with dimensions [95, 44, 11] (5 of 5 Assertions)
- Block with dimensions [57, 46, 13] (5 of 5 Assertions)
- Block with dimensions [85, 61, 54] (5 of 5 Assertions)
- Block with dimensions [15, 70, 19] (5 of 5 Assertions)
- Block with dimensions [17, 78, 80] (5 of 5 Assertions)
- Block with dimensions [10, 35, 15] (5 of 5 Assertions)
- Block with dimensions [13, 46, 8] (5 of 5 Assertions)
- Block with dimensions [49, 40, 25] (5 of 5 Assertions)

You have passed all of the tests! :)

Solution

```

1 class Block:
2     def __init__(self, block):
3         self.w = block[0]
4         self.l = block[1]
5         self.h = block[2]
6     def get_width(self):
7         return self.w
8     def get_length(self):
9         return self.l
10    def get_height(self):
11        return self.h
12    def get_volume(self):
13        return self.w * self.h * self.l
14    def get_surface_area(self):
15        return (self.w * self.l + self.w * self.h + self.h * self.l) * 2

```

Impressive! You may take your time to refactor/comment your solution. Submit when ready.

Sample Tests

```

1 block1 = Block([2,2,2])
2 test.assertEqual(block1.get_width(), 2)
3 test.assertEqual(block1.get_length(), 2)
4 test.assertEqual(block1.get_height(), 2)
5 test.assertEqual(block1.get_volume(), 8)
6 test.assertEqual(block1.get_surface_area(), 24)

```

SKIP UNLOCK SOLUTIONS DISCUSS (22) RESET TEST SUBMIT

第三题

```

import math

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 math.ceil(self.item_count() / 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:

            last_page = self.item_count() % self.items_per_page

            return self.items_per_page if last_page == 0 else last_page

        else:
            return self.items_per_page

    def page_index(self, item_index):
        if item_index < 0 or item_index >= self.item_count():
            return -1
        else:
            return item_index // self.items_per_page

```

第四题

```

from math import sqrt

class Vector:

    def __init__(self, iterable):
        self.v = tuple(x for x in iterable)

    def __str__(self):
        return str(self.v).replace(' ', '')

    def check(self, other):
        if not len(self.v) == len(other.v):
            raise ValueError('Vectors of different length')

    def add(self, other):
        self.check(other)
        return Vector(s + o for s, o in zip(self.v, other.v))

    def subtract(self, other):
        self.check(other)
        return Vector(s - o for s, o in zip(self.v, other.v))

    def dot(self, other):
        self.check(other)
        return sum(s * o for s, o in zip(self.v, other.v))

    def norm(self):

```



```

        return sqrt(sum(x**2 for x in self.v))

    def equals(self, other):
        return self.v == other.v

```

5 kyu Vector class

☆ 565 🌟 122 🔄 88% of 824 📊 3,950 of 7,370 👤 eugene-bulkin

⚠️ 8 Issues Reported

Instructions Output

Time: 471ms Passed: 15 Failed: 0

Test Results:

- Testing arithmetic
 - Addition
 - Subtraction
 - Dot Product (2 of 2 Assertions)
 - Norms (3 of 3 Assertions)
 - Equality (5 of 5 Assertions)
 - Strings (3 of 3 Assertions)

Completed in 0.32ms

You have passed all of the tests! :)

Solution

```

1 from math import sqrt
2
3 class Vector:
4
5     def __init__(self, iterable):
6         self.v = tuple(x for x in iterable)
7
8     def __str__(self):
9         return str(self.v).replace(' ', '')
10
11     def check(self, other):
12         if not len(self.v) == len(other.v):
13             raise ValueError('Vectors of different length')
14
15     def add(self, other):
16         self.check(other)

```

Good Job! You may take your time to refactor/comment your solution. Submit when ready.

Sample Tests

```

4 @test.describe("Vector tests")
5 def vector_tests():
6
7     @test.it("Example tests")
8     def example_tests():
9
10         a = Vector([1, 2])
11         b = Vector([3, 4])
12
13         test.expect(a.add(b).equals(Vector([4, 6])))

```

SKIP UNLOCK SOLUTIONS DISCUSS (98) RESET TEST SUBMIT

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

2. Python语言中如何继承父类和改写（override）父类的方法。

在Python中，类的继承和方法的重写（override）都是非常直观和简单的。下面是一些基本示例：

定义父类（基类）

```

class Parent:
    def __init__(self):
        print("Parent's __init__")

    def foo(self):
        print("Parent's foo")

    def bar(self):
        print("Parent's bar")

```

```
# 定义子类, 继承自Parent
class Child(Parent):
    def __init__(self):
        super().__init__() # 调用父类的初始化方法
        print("Child's __init__")

    def foo(self): # 重写父类的foo方法
        super().foo() # 可选: 调用父类的foo方法
        print("Child's foo")

# 测试代码
c = Child() # 创建Child类的实例
c.foo() # 调用重写后的foo方法
c.bar() # 调用继承自父类的bar方法
```

在这个例子中, `Child` 类继承自 `Parent` 类, 这意味着 `Child` 类会自动获得 `Parent` 类的所有方法。我们可以在 `Child` 类中重写这些方法, 以改变它们的行为。

在 `Child` 的 `__init__` 和 `foo` 方法中, 我们使用了 `super()` 函数来调用父类的对应方法。这是一种常见的模式, 可以让我们在子类中添加新的行为, 同时保留父类的行为。当然, 如果我们想完全替换父类的行为, 也可以不调用 `super()`。

注意: 在 Python 3 中, 我们可以直接使用 `super()` 而不需要参数。但在 Python 2 中, 我们需要将当前类和实例作为参数传给 `super()`, 例如 `super(Child, self)`。

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。

Python中的类有许多特殊的方法, 这些方法在特定的情况下会被自动调用。这些方法通常由两个下划线开始和结束, 例如 `__init__`, `__str__` 等。以下是一些常见的特殊方法及其用途:

1. `__init__(self, ...)` : 构造函数, 在创建新实例时被调用。
2. `__str__(self)` : 当实例被转换为字符串时 (例如通过 `str()` 函数或 `print` 语句), 返回一个代表该实例的可读字符串。
3. `__add__(self, other)` : 定义 `+` 操作符的行为。

以下是一些简单的代码示例:

```
class Complex:
    def __init__(self, real, imag):
        self.real = real
        self.imag = imag

    def __str__(self):
        return f"{self.real} + {self.imag}j"

    def __add__(self, other):
        if isinstance(other, Complex):
            return Complex(self.real + other.real, self.imag + other.imag)
```

```
        else:
            return Complex(self.real + other, self.imag)

# 测试代码
c1 = Complex(1, 2)
c2 = Complex(2, 3)
print(c1) # 输出: 1 + 2j
print(c2) # 输出: 2 + 3j
c3 = c1 + c2
print(c3) # 输出: 3 + 5j
```

在这个例子中，我们定义了一个表示复数的类`Complex`。`__init__`方法用于初始化复数的实部和虚部。`__str__`方法返回一个表示复数的字符串。`__add__`方法定义了两个复数相加的行为，也可以与一个实数相加。

实验总结

本次的Python面向对象编程实验，我完成了教材第九章的类章节，编写CodeWars中的编程题，使用Mermaid绘制程序流程图并对本实验的思考题进行解答。通过这次实验，我学习了Python面向对象编程的基本概念，包括类、对象、属性、方法、继承等，编写了Python类、继承和重写等方面的代码，并尝试导入模块和库。