

# 实验七 Python面向对象编程

---

班级： 21计科03

学号： B20210302322

姓名： 董一浩

Github地址： <https://github.com/ByL1eng/xuexi>

CodeWars地址： <https://www.codewars.com/users/ByL1eng>

---

## 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](#)

注意代码需要使用markdown的代码块格式化，例如Git命令行语句应该使用下面的格式：

 Git命令

显示效果如下：

```
git init  
git add .  
git status  
git commit -m "first commit"
```

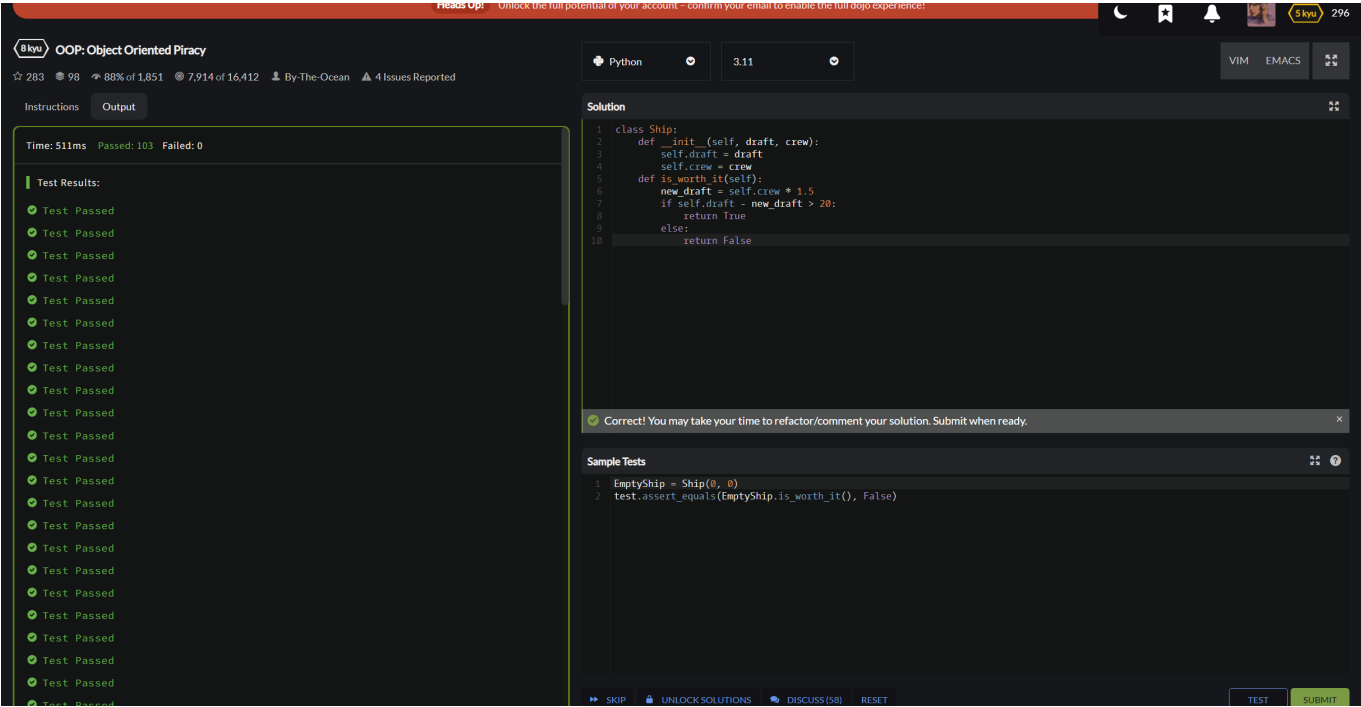
如果是Python代码，应该使用下面代码块格式，例如：

 Python代码

显示效果如下：

```
def add_binary(a,b):  
    return bin(a+b)[2:]
```

第一问：面向对象的海盗



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

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

第二问：搭建积木

7 kyu Building blocks

☆ 154 ● 57 93% of 1,258 ● 2,820 of 10,295 NaMe613 ▲ 3 Issues Reported

Instructions Output

Time: 476ms Passed: 100 Failed: 0

Test Results:

- Basic Tests
  - Block with dimensions [1, 1, 1] (5 of 5 Assertions)
  - Block with dimensions [2, 2, 2] (5 of 5 Assertions)
  - Block with dimensions [3, 3, 3] (5 of 5 Assertions)
  - Block with dimensions [4, 4, 4] (5 of 5 Assertions)
  - Block with dimensions [5, 5, 5] (5 of 5 Assertions)
  - Block with dimensions [6, 6, 6] (5 of 5 Assertions)
  - Block with dimensions [7, 7, 7] (5 of 5 Assertions)
  - Block with dimensions [8, 8, 8] (5 of 5 Assertions)
  - Block with dimensions [9, 9, 9] (5 of 5 Assertions)
  - Block with dimensions [10, 10, 10] (5 of 5 Assertions)
- Random Tests

You have passed all of the tests! :)

Solution

```
1 class Block:
2     def __init__(self, block):
3         self.length = block[1]
4         self.width = block[0]
5         self.height = block[2]
6
7     def get_width(self):
8         return self.width
9
10    def get_length(self):
11        return self.length
12
13    def get_height(self):
14        return self.height
15
16    def get_volume(self):
17        height = self.get_height()
18        length = self.get_length()
19        width = self.get_width()
20
21        return height * width * length
22
23    def get_surface_area(self):
24        height = self.get_height()
25        length = self.get_length()
26        width = self.get_width()
27
28        return (length * width * 2) + (length * height * 2) + (width * height * 2)
```

Outstanding! 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

```
class Block:
    def __init__(self, block):
        self.length = block[1]
        self.width = block[0]
        self.height = block[2]

    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):
        height = self.get_height()
        length = self.get_length()
        width = self.get_width()

        return height * width * length

    def get_surface_area(self):
        height = self.get_height()
        length = self.get_length()
        width = self.get_width()

        return (length * width * 2) + (length * height * 2) + (width * height * 2)
```

代码提交地址: <https://www.codewars.com/kata/55b75fc67e558d3750000a3>



## 第三问：分页助手

难

The screenshot shows a coding challenge titled "PaginationHelper" with a difficulty level of "难" (Hard). The interface includes a sidebar with test results, a main editor for the solution, and a bottom section for sample tests.

**Test Results:**

- Time: 472ms Passed: 216 Failed: 0
- Fixed tests:
  - Sample tests from description (9 of 9 Assertions)
  - Basic tests (20 of 20 Assertions)
  - Edge case: List [1,2,3,4] with 4 items per page (7 of 7 Assertions)
  - Edge case: Empty list (8 of 8 Assertions)
  - Edge case: List [1,2,3,4] with 1 item per page (12 of 12 Assertions)
- Random tests:
  - List with 28 items and 40 items per page (4 of 4 Assertions)
  - List with 35 items and 46 items per page (4 of 4 Assertions)
  - List with 45 items and 64 items per page (4 of 4 Assertions)
  - List with 20 items and 4 items per page (4 of 4 Assertions)
  - List with 23 items and 26 items per page (4 of 4 Assertions)
  - List with 0 items and 1 item per page (4 of 4 Assertions)
  - List with 36 items and 19 items per page (4 of 4 Assertions)
  - List with 0 items and 1 item per page (4 of 4 Assertions)
  - List with 23 items and 9 items per page (4 of 4 Assertions)
  - List with 6 items and 3 items per page (4 of 4 Assertions)
  - List with 12 items and 9 items per page (4 of 4 Assertions)
  - List with 41 items and 60 items per page (4 of 4 Assertions)

**Solution:**

```

1 # TODO: complete this class
2
3 class PaginationHelper:
4
5     def __init__(self, collection, items_per_page):
6         self.collection = collection
7         self.items_per_page = items_per_page
8
9     def item_count(self):
10        return len(self.collection)
11
12    def page_count(self):
13        count = len(self.collection) // self.items_per_page
14        if len(self.collection) % self.items_per_page == 0:
15            return count
16        else:
17            return count + 1
18
19    def page_item_count(self, page_index):
20        count = self.page_count()
21        item_count = [[]] * count
22        if page_index >= count or page_index < 0:
23            return -1

```

**Sample Tests:**

```

24 def __init__(self):
25     test.assert_equals(helper.page_index( 5), 1, 'page_index returned incorrect value for item_index 5')
26     test.assert_equals(helper.page_index( 2), 0, 'page_index returned incorrect value for item_index 2')
27     test.assert_equals(helper.page_index(20), -1, 'page_index returned incorrect value for item_index 20')
28     test.assert_equals(helper.page_index(-10), -1, 'page_index returned incorrect value for item_index -10')
29
30 @test.it("Empty List")
31 def __init__(self):
32     empty = PaginationHelper([], 10)
33     test.assert_equals(empty.item_count(), 0, "item_count is returning incorrect value")
34     test.assert_equals(empty.page_count(), 0, "page_count is returning incorrect value")
35     test.assert_equals(empty.page_index( 0), -1, "page_index(0) called when there was an empty collection")
36     test.assert_equals(empty.page_index( 1), -1, "page_index(1) called when there was an empty collection")
37     test.assert_equals(empty.page_index(-1), -1, "page_index(-1) called when there was an empty collection")
38     test.assert_equals(empty.page_item_count(0), -1, "page_item_count is returning incorrect value for page_index 0")
39     test.assert_equals(empty.page_item_count(1), -1, "page_item_count is returning incorrect value for page_index 1")

```

```
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):
        count = len(self.collection) // self.items_per_page
        if len(self.collection) % self.items_per_page == 0:
            return count
        else:
            return count + 1
```

```
    def page_item_count(self, page_index):
        count = self.page_count()
        item_count = [[]] * count
        if page_index >= count or page_index < 0:
            return -1
        else:
            for i in range(count):
                item_count[i] = self.collection[i * self.items_per_page:(i + 1) *
self.items_per_page]
            return len(item_count[page_index])
```

```
    def page_index(self, item_index):
        if item_index >= len(self.collection):
            return -1
```

```

else:
    empty = []
    if self.collection == empty:
        return -1
    else:
        n, m = divmod(item_index, self.items_per_page)
        count = self.page_count()
        if n >= count or item_index < 0:
            return -1
        else:
            return n

```

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

## 第四问: 向量 (Vector) 类

The screenshot shows the Codewars interface for the 'Vector class' kata. The solution code is written in Python and includes methods for initialization, string representation, equality comparison, norm calculation, and addition. The test results show that all tests passed.

**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 0.34ms

**Solution Code:**

```

class Vector:
    def __init__(self, vector):
        self.new_list = vector

    def __str__(self):
        s = ",".join([str(element) for element in self.new_list])
        return '(%s)' % s

    def equals(self, vector):
        if self.__str__() == vector.__str__():
            return True
        else:
            return False

    def compare(self, vector):
        if len(self.new_list) == len(vector.new_list):
            return True
        else:
            return False

    def norm(self):
        return sqrt(self.new_list[0]**2 + self.new_list[1]**2 + self.new_list[2]**2)

    def add(self, object):

```

**Sample Tests:**

```

@unittest.TestCase
def example_tests():
    a = Vector([1, 2])
    b = Vector([3, 4])
    test.expect(a.add(b).equals(Vector([4, 6])))

    a = Vector([1, 2, 3])
    b = Vector([3, 4, 5])
    test.expect(a.add(b).equals(Vector([4, 6, 8])))
    test.expect(a.subtract(b).equals(Vector([-2, -2, -2])))

```

```

from math import sqrt

class Vector:
    def __init__(self, vector):
        self.new_list = vector

    def __str__(self):
        s = ",".join([str(element) for element in self.new_list])
        return '(%s)' % s

    def equals(self, vector):
        if self.__str__() == vector.__str__():
            return True
        else:

```

```
        return False

    def compare(self, vector):
        if len(self.new_list) == len(vector.new_list):
            return True
        else:
            return False

    def norm(self):
        return sqrt(self.new_list[0] ** 2 + self.new_list[1] ** 2 +
self.new_list[2] ** 2)

    def add(self, Object):
        if self.compare(Object):
            new_vector = Vector
            new_list = []
            for i in range(len(self.new_list)):
                new_list.append(self.new_list[i] + Object.new_list[i])
            return new_vector(new_list)
        else:
            return 'raises an exception'

    def subtract(self, vector):
        if self.compare(vector):
            new_vector = Vector
            new_list = []
            for i in range(len(self.new_list)):
                new_list.append(self.new_list[i] - vector.new_list[i])
            return new_vector(new_list)
        else:
            return 'raises an exception'

    def dot(self, vector):
        if self.compare(vector):
            sum_number = 0
            for i in range(len(self.new_list)):
                sum_number += (self.new_list[i] * vector.new_list[i])

            return sum_number
        else:
            return 'raises an exception'
```

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

---

第五问: Codewars风格的等级系统

**Codewars style ranking system**

1742 ● 270 86% of 1,667 ● 5,895 of 14,713 jhoffner ▲ 36 Issues Reported

Instructions Output

Time: 539ms Passed: 1053 Failed: 0

**Test Results:**

- Sample tests + more fixed tests (50 of 50 Assertions)
- should handle invalid range values (3 of 3 Assertions)
- Randomized tests
  - should work for progressing: -5 -5 3 8 -2 6 -7 8 -3 -8 (20 of 20 Assertions)
  - should work for progressing: -4 7 -5 7 5 5 -6 1 8 -7 (20 of 20 Assertions)
  - should work for progressing: 6 5 -3 1 8 -2 -8 -1 -8 2 (20 of 20 Assertions)
  - should work for progressing: -1 -3 -5 2 5 -4 8 4 1 -2 (20 of 20 Assertions)
  - should work for progressing: 2 3 -1 -5 -1 7 -4 2 -8 -6 (20 of 20 Assertions)
  - should work for progressing: 1 4 -5 2 1 -1 7 -5 -4 7 (20 of 20 Assertions)
  - should work for progressing: -1 5 -7 1 -3 -6 6 8 -3 -5 (20 of 20 Assertions)
  - should work for progressing: -7 7 1 6 -2 -7 -1 -8 7 -2 (20 of 20 Assertions)
  - should work for progressing: 3 -3 7 -5 4 -6 -4 3 -3 5 (20 of 20 Assertions)
  - should work for progressing: -6 8 3 6 -2 -4 7 2 2 2 (20 of 20 Assertions)
  - should work for progressing: 1 1 3 8 -5 -5 -5 -2 3 -6 (20 of 20 Assertions)
  - should work for progressing: -2 -8 -4 1 -8 2 5 7 2 -8 (20 of 20 Assertions)
  - should work for progressing: 2 -1 1 4 8 -4 6 -1 3 -4 (20 of 20 Assertions)
  - should work for progressing: 8 -1 6 -6 3 -7 -7 1 8 5 (20 of 20 Assertions)
  - should work for progressing: 6 -6 -5 -7 7 8 4 -2 -1 -5 (20 of 20 Assertions)
  - should work for progressing: -8 4 5 -5 4 3 8 8 8 1 (20 of 20 Assertions)

**Solution**

```

1 self.progress = 0
2
3 def inc_progress(self, kata):
4     if kata not in self.rank_vector:
5         raise ValueError("Not in the specified Range of features")
6     if self.rank == 8:
7         progressmeter = 0
8     elif self.rank_vector.index(kata) == self.rank_vector.index(self.rank):
9         progressmeter = self.progress + 3
10    elif self.rank_vector.index(kata) == self.rank_vector.index(self.rank) - 1:
11        progressmeter = self.progress + 1
12    elif self.rank_vector.index(kata) <= self.rank_vector.index(self.rank) - 2:
13        progressmeter = self.progress
14    elif self.rank == -1 and kata == 1:
15        progressmeter = self.progress + 10
16
17    else:
18        progressmeter = self.progress + 10 * pow(
19            abs(self.rank_vector.index(kata) - self.rank_vector.index(self.rank)), 2)
20        progressIndex = list(divmod(progressmeter, 100))
21        self.progress = progressIndex[1]
22        self.rank = self.updaterank(progressIndex[0])
23        if self.rank == 8:
24            self.progress = 0
25
26 Correctamundo! You may take your time to refactor/comment your solution. Submit when ready.

```

**Sample Tests**

```

18 do_test(-7, -8, 10)
19
20 user = User()
21 do_test(-6, -8, 40)
22
23 user = User()
24 do_test(-5, -8, 90)
25
26 user = User()
27 do_test(-4, -7, 60)
28
29 user = User()
30 do_test(1, -2, 40)
31 do_test(1, -2, 80)
32 do_test(1, -1, 80)
33 do_test(1, -1, 30)

```

```

class User:
    rank_vector = [i for i in range(-8, 9, 1) if (i != 0)]

    def __init__(self):
        self.rank = -8
        self.progress = 0

    def inc_progress(self, kata):
        if kata not in self.rank_vector:
            raise ValueError("Not in the specified Range of features")
        if self.rank == 8:
            progressmeter = 0
        elif self.rank_vector.index(kata) == self.rank_vector.index(self.rank):
            progressmeter = self.progress + 3
        elif self.rank_vector.index(kata) == self.rank_vector.index(self.rank) -
1:
            progressmeter = self.progress + 1
        elif self.rank_vector.index(kata) <= self.rank_vector.index(self.rank) -
2:
            progressmeter = self.progress
        elif self.rank == -1 and kata == 1:
            progressmeter = self.progress + 10

        else:
            progressmeter = self.progress + 10 * pow(
                abs(self.rank_vector.index(kata) -
self.rank_vector.index(self.rank)), 2)
            progressIndex = list(divmod(progressmeter, 100))
            self.progress = progressIndex[1]
            self.rank = self.updaterank(progressIndex[0])
            if self.rank == 8:
                self.progress = 0
            return self.progress

```

```
def updatetank(self, level=1):  
  
    if self.rank == 8:  
        return self.rank  
    elif self.rank_vector.index(self.rank) + level >  
self.rank_vector.index(8):  
        self.rank = 8  
    else:  
        self.rank = self.rank_vector[self.rank_vector.index(self.rank) +  
level]  
    return self.rank
```

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  
class Ship{ +int draft  
+int crew  
+is_worth_it() }
```

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。代码运行结果的文本可以直接粘贴在这里。

**注意：不要使用截图，Markdown文档转换为Pdf格式后，截图可能会无法显示。**

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

## 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。