

实验七 Python面向对象编程

班级：21计科1

学号：B20210302110

姓名：刘湘怡

Github地址：https://github.com/righting1/python_Experiments

CodeWars地址：<https://www.codewars.com/users/righting1>

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>

The screenshot shows the Codewars interface for the kata "OOP: Object Oriented Piracy" (8 kyu). The page includes a header with the kata title, difficulty, and a progress bar. Below the header, there are tabs for "Instructions" and "Output". The "Output" tab is active, showing a list of test results, all of which are "Passed". To the right of the test results, there is a "Solution" section with a code editor showing the implementation of the `Ship` class. The code defines the `__init__` method and the `is_worth_it` method. The `is_worth_it` method calculates the draft minus 1.5 times the crew and returns `True` if the result is greater than 20. Below the solution, there is a "Sample Tests" section with a code editor showing a test case: `EmptyShip = Ship(0, 0)` and `test.assertEqual(EmptyShip.is_worth_it(), False)`. A message at the bottom of the solution section says: "Excellent! You may take your time to refactor/comment your solution. Submit when ready."

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

```
# Your code here
def is_worth_it(self):
    return self.draft-self.crew*1.5>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>

The screenshot shows the Codewars interface for the 'Building blocks' kata (7kyu). The solution is written in Python 3.11. The code defines a `Block` class with methods `get_width`, `get_length`, `get_height`, `get_volume`, and `get_surface_area`. The test results show that all 10 basic tests passed. The sample tests at the bottom show the expected behavior of the `Block` class.

```
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):
        ans = self.width * self.length * self.height
        return ans

    def get_surface_area(self):
        ans1 = self.width * self.length
        ans2 = self.width * self.height
        ans3 = self.length * self.height
        ans = 2 * (ans1 + ans2 + ans3)
        return ans
```

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)

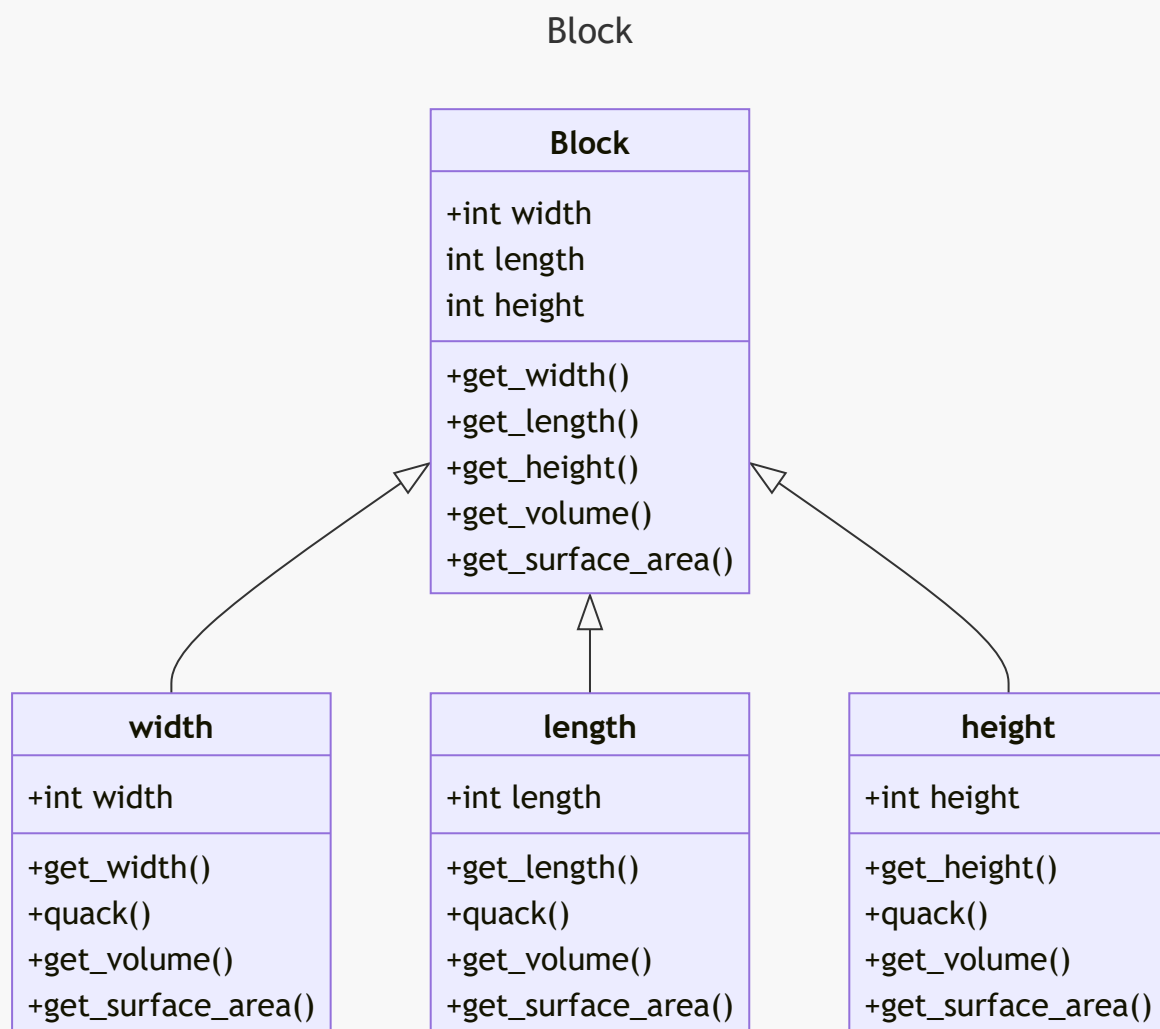
Sample Tests:

```
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)
```

```

class Block:
    # Good Luck!
    def __init__(self, args):
        self.width=args[0]
        self.length=args[1]
        self.height=args[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):
        ans=self.width*self.length*self.height
        return ans
    def get_surface_area(self):
        ans1=self.width*self.length
        ans2=self.width*self.height
        ans3=self.length*self.height
        ans=2*(ans1+ans2+ans3)
        return ans

```



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>

The screenshot shows the Codewars interface for the 'PaginationHelper' kata (5 kyu). The top bar includes the kata title, difficulty (5 kyu), and statistics (1662 stars, 314 solutions, 81% of 3,159 users solved it). The 'Solution' tab is active, displaying a Python solution for the 'page_index' method. The 'Test Results' tab shows that all tests passed (216 passed, 0 failed) in 658ms. The 'Sample Tests' section shows a list of test cases for the 'empty' instance of the class.

Test Results:

- Fixed tests (9 of 9 Assertions)
 - 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 (4 of 4 Assertions)
 - List with 9 items and 8 items per page (4 of 4 Assertions)
 - List with 24 items and 26 items per page (4 of 4 Assertions)
 - List with 31 items and 15 items per page (4 of 4 Assertions)

Solution:

```
41
42
43
44 # determines what page an item at the given index is on. Zero based indexes.
45 # this method should return -1 for item_index values that are out of range
46 def page_index(self, item_index):
47     if item_index < 0:
48         return -1
49     if item_index >= len(self.collection):
50         return -1
51     ans = (item_index) / (self.items_per_page)
52     ans = int(ans)
53     return ans
54
55 Good Job! You may take your time to refactor/comment your solution. Submit when ready.
```

Sample Tests:

```
31 empty = PaginationHelper([], 10)
32 test.assert_equals(empty.item_count(), 0, "item_count is returning incorrect value")
33 test.assert_equals(empty.page_count(), 0, "page_count is returning incorrect value")
34 test.assert_equals(empty.page_index(0), -1, "page_index(0) called when there was an empty")
35 test.assert_equals(empty.page_index(1), -1, "page_index(1) called when there was an empty")
36 test.assert_equals(empty.page_index(-1), -1, "page_index(-1) called when there was an empty")
37 test.assert_equals(empty.item_count(0), -1, "page_item_count is returning incorrect value")
38 test.assert_equals(empty.page_item_count(1), -1, "page_item_count is returning incorrect value")
39 test.assert_equals(empty.page_item_count(-1), -1, "page_item_count is returning incorrect value")
```

TODO: complete this class

class PaginationHelper:

The constructor takes in an array of items and an integer indicating
how many items fit within a single page

```
def __init__(self, collection, items_per_page):
    #pass
    self.collection=collection
    self.items_per_page=items_per_page

# returns the number of items within the entire collection
def item_count(self):
    #pass
    cnt=len(self.collection)
    return cnt

# returns the number of pages
def page_count(self):
    #pass
    cnt1=len(self.collection)
    ans=(cnt1+self.items_per_page-1)/self.items_per_page
    return int(ans)

# returns the number of items on the given page. page_index is zero based
# this method should return -1 for page_index values that are out of range
def page_item_count(self, page_index):
    #pass
    lim1=(len(self.collection))/self.items_per_page
    lim1=int(lim1)
    lim2=(len(self.collection)+self.items_per_page-1)/self.items_per_page
    lim2=int(lim2)
    ans3=(len(self.collection))%(self.items_per_page)
    if page_index>=lim2 :
        return -1
    if page_index<0 :
        return -1
    if page_index<lim1:
        return self.items_per_page
    return ans3

# determines what page an item at the given index is on. Zero based indexes.
# this method should return -1 for item_index values that are out of range
def page_index(self, item_index):
    if item_index<0:
        return -1
    if item_index>=len(self.collection):
        return -1
    ans=(item_index)/(self.items_per_page)
    ans=int(ans)
    return ans
```

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>

The screenshot shows the Codewars interface for the 'Vector class' kata. The top bar indicates it's a 5kyu problem with 564 stars, 122 comments, and 88% of 823 users solved it. The 'Solution' tab is active, displaying a Python solution. The 'Test Results' tab on the left shows that all tests passed, including arithmetic operations and string representation. A message at the bottom of the test results states 'You have passed all of the tests! :)'.

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 0.29ms

You have passed all of the tests! :)

Solution

```
3 class Vector:
4     def __init__(self, args):
5         self.v=tuple(x for x in args)
6
7     def __str__(self):
8         return str(self.v).replace(' ','')
9
10    def check(self, other):
11        if not len(self.v) == len(other.v):
12            raise ValueError('Vectors of different length')
13
14    def add(self, other):
15        self.check(other)
16        return Vector(x+y for x,y in zip(self.v, other.v))
17
18    def subtract(self, other):
19        self.check(other)
20        return Vector(x-y for x,y in zip(self.v, other.v))
21
22    def dot(self, other):
23        self.check(other)
24        return sum(x*y for x,y in zip(self.v, other.v))
25
26    def norm(self):
27        return sqrt(sum(x**2 for x in self.v))
```

Sample Tests

```
14 test.expect(b.add(b).equals(Vector([4, 6])))
15
16 a = Vector([1, 2, 3])
17 b = Vector([3, 4, 5])
18
19 test.expect(a.add(b).equals(Vector([4, 6, 8])))
20 test.expect(a.subtract(b).equals(Vector([-2, -2, -2])))
21 test.assertEqual(a.dot(b), 26)
22 test.assertEqual(a.norm(), sqrt(14))
```

```
from math import sqrt

class Vector:
    def __init__(self, args):
        self.v=tuple(x for x in args)

    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(x+y for x,y in zip(self.v, other.v))

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

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

    def norm(self):
        sum=0
        for x in self.v:
            sum+=x*x
        return sqrt(sum)

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

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：

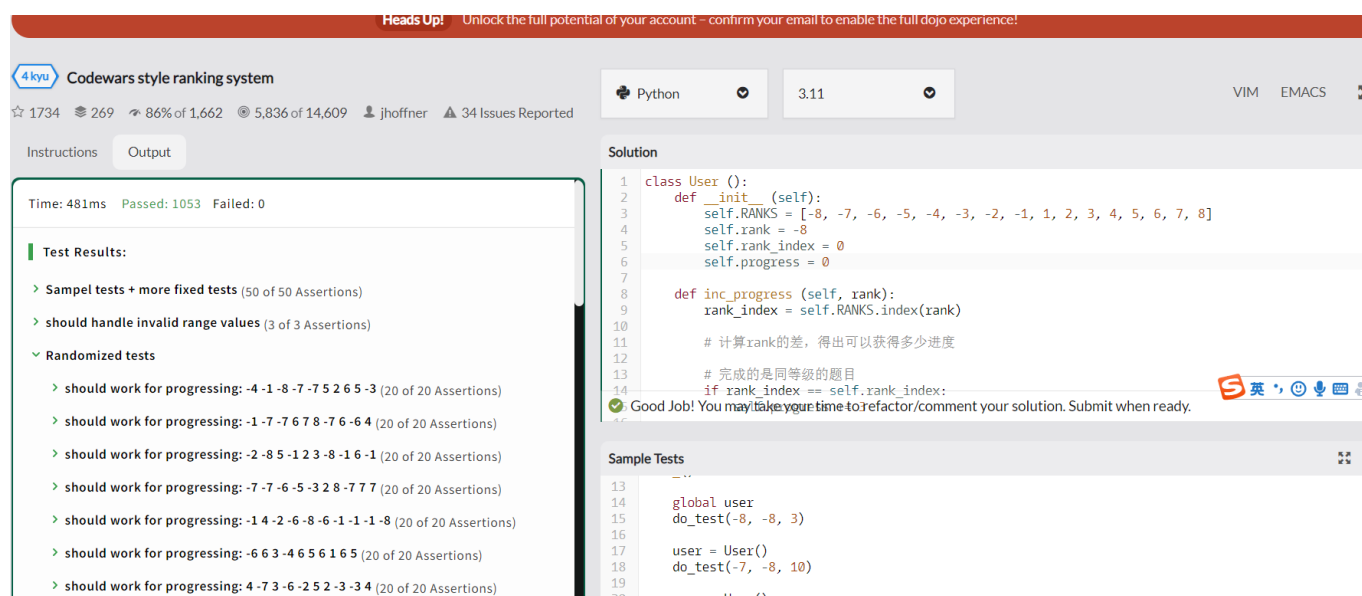
- 如果一个排名为-8的用户完成了一个排名为-7的活动，他们将获得10的进度。
- 如果一个排名为-8的用户完成了排名为-6的活动，他们将获得40的进展。
- 如果一个排名为-8的用户完成了排名为-5的活动，他们将获得90的进展。
- 如果一个排名-8的用户完成了排名-4的活动，他们将获得160个进度，从而使该用户升级到排名-7，并获得60个进度以获得下一个排名。

- 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>



Heads Up! Unlock the full potential of your account - confirm your email to enable the full dojo experience!

4 kyu Codewars style ranking system

Python 3.11

VIM EMACS

1734 269 86% of 1,662 5,836 of 14,609 jhoffner 34 Issues Reported

Instructions Output

Time: 481ms Passed: 1053 Failed: 0

Test Results:

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

Solution

```
1 class User():
2     def __init__(self):
3         self.RANKS = [-8, -7, -6, -5, -4, -3, -2, -1, 1, 2, 3, 4, 5, 6, 7, 8]
4         self.rank = -8
5         self.rank_index = 0
6         self.progress = 0
7
8     def inc_progress(self, rank):
9         rank_index = self.RANKS.index(rank)
10
11         # 计算rank的差，得出可以获得多少进度
12
13         # 完成的是同等级的题目
14         if rank_index == self.rank_index:
15             self.progress += 3
16
17         # 完成的是比当前等级低一级的题目
18         elif rank_index == self.rank_index - 1:
19             self.progress += 1
```

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

Sample Tests

```
13 global user
14 do_test(-8, -8, 3)
15
16 user = User()
17 do_test(-7, -8, 10)
18
19 user = User()
```

```
class User():
    def __init__(self):
        self.RANKS = [-8, -7, -6, -5, -4, -3, -2, -1, 1, 2, 3, 4, 5, 6, 7, 8]
        self.rank = -8
        self.rank_index = 0
        self.progress = 0

    def inc_progress(self, rank):
        rank_index = self.RANKS.index(rank)

        # 计算rank的差，得出可以获得多少进度

        # 完成的是同等级的题目
        if rank_index == self.rank_index:
            self.progress += 3

        # 完成的是比当前等级低一级的题目
        elif rank_index == self.rank_index - 1:
            self.progress += 1
```

```
# 完成的是比当前等级高的题目
elif rank_index > self.rank_index:
    difference = rank_index - self.rank_index
    self.progress += 10 * difference * difference

# 如果进度大于100, 升级, 每减去100进度, 升一级
while self.progress >= 100:
    self.rank_index += 1
    self.rank = self.RANKS[self.rank_index]
    self.progress -= 100

# 如果升到8级 (最高级), 进度被置为0
if self.rank == 8:
    self.progress = 0
    break
if self.rank==8:
    self.progress = 0
```

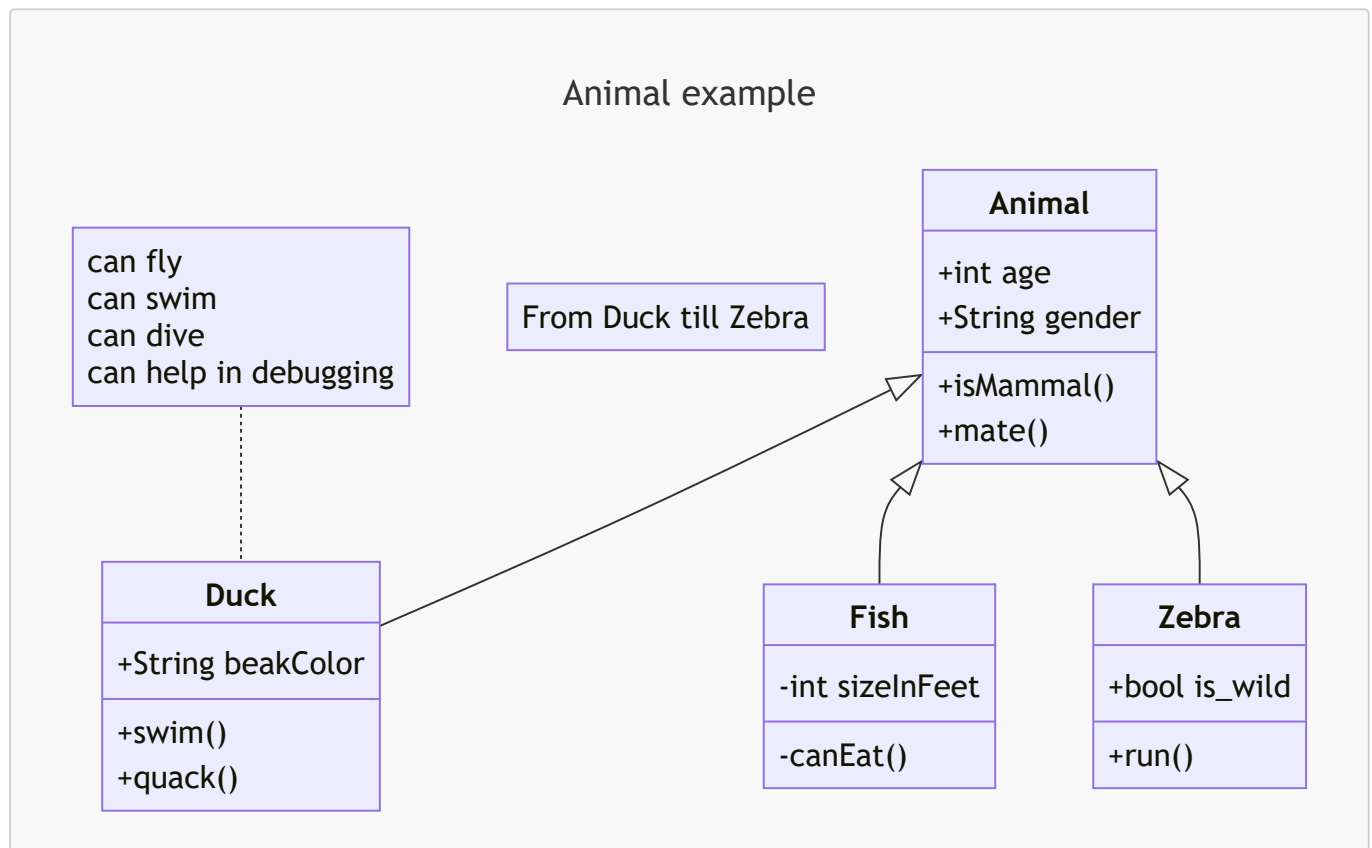
第三部分

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：



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：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：

显示效果如下：

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

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？在Python中，__init__方法是一个特殊的方法，用于在创建类的新实例时进行初始化。当你使用class关键字创建一个新类并使用new_object = ClassName()来创建一个该类的实例时，Python会自动调用__init__方法。

__init__方法允许类接受初始参数，这样我们可以在创建实例时设置属性。例如，如果我们有一个Person类，我们可能希望在创建Person实例时立即设定他们的姓名和年龄。下面是如何使用__init__方法来实现这个目标的代码示例：

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

在这个例子中，`__init__`方法接受两个参数：`name`和`age`。当我们创建新的`Person`实例时，我们需要提供这两个参数，如下所示：

```
bob = Person("Bob", 32)
```

在这个例子中，字符串"Bob"和数字32分别传递给`__init__`方法的`name`和`age`参数，然后这些值被用来初始化新创建的`Person`实例。

2. Python语言中如何继承父类和改写（override）父类的方法。在Python中，可以使用`class`语句来定义一个类，并使用`extends`关键字来继承父类。如果子类需要改写父类的方法，可以在子类中重新定义该方法。

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。Python类有许多特殊的方法，这些方法以双下划线开头和结尾，例如`__init__`、`__str__`、`__repr__`、`__call__`等。这些特殊方法在Python中被称为"魔法方法"或"双下划线方法"。它们可以改变类的行为或提供额外的功能。以下举三个例子来说明：

`__init__`: 这是一个构造器方法，当创建类的新实例时，会自动调用这个方法。它可以用来初始化新创建的对象的状态。

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

person = Person("Alice", 25) # 创建一个Person类的实例
print(person.name) # 输出: Alice
print(person.age) # 输出: 25
```

`__str__`: 当我们将类的实例转换为字符串时，例如使用`print()`函数，或者将实例与字符串进行连接操作时，Python会自动调用这个方法。它可以用来定义实例的字符串表示。

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

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

person = Person("Alice", 25)
print(person) # 输出: Alice, 25 years old
```

`__call__`: 当我们将类的实例像函数一样调用时, 会自动调用这个方法。这可以用来定义类的实例的行为。

```
class Multiplier:
    def __init__(self, factor):
        self.factor = factor
    def __call__(self, num):
        return self.factor * num

multiplier = Multiplier(3) # 创建一个Multiplier类的实例, factor为3
result = multiplier(10) # 调用__call__方法, 相当于multiply(3)(10)
print(result) # 输出: 30
```

以上就是三个例子, 实际上Python类有许多其他的特殊方法, 如`__eq__`, `__ne__`, `__lt__`, `__gt__`, `__le__`, `__ge__`等, 可以用来定义对象之间的比较行为, 还有`__del__`用来定义对象的销毁行为等等。

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。看题要细心, 写上面的一个题题目中的一些要求没有看到导致找了好久的问題