

实验七 Python面向对象编程

班级： 21计科4

学号： 20210301105

姓名： 张湘睿

Github地址： <https://github.com/ttZhang0512/PythonClassTasks.git>

CodeWars地址： <https://www.codewars.com/users/ttZhang0512>

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>

第三部分

使用Mermaid绘制程序的类图

安装VSCode插件：

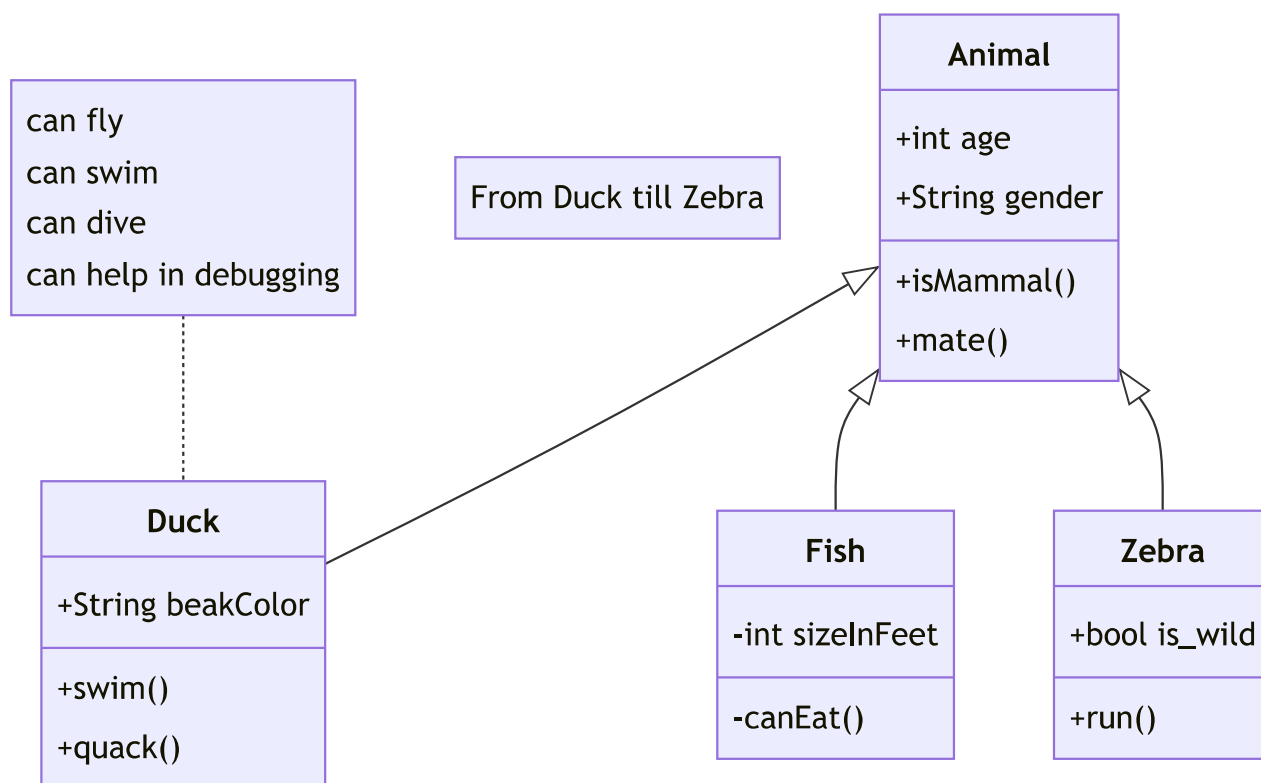
- Markdown Preview Mermaid Support
- Mermaid Markdown Syntax Highlighting

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：

```
---
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()
    }
```

显示效果如下：

Animal example



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](#)

第一题代码

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

第二题代码

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

第三题代码

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


第四题代码

```
from math import sqrt
class Vector:
    def __init__(self, iterable):
        self.examples = tuple(i for i in iterable)
    def __str__(self):
        return str(self.examples).replace(' ', '')
    def check(self, str2):
        if not len(self.examples) == len(str2.examples):
            raise ValueError('Vectors of different length')
    def add(self, str2):
        self.check(str2)
        return Vector(i+j for i, j in zip(self.examples, str2.examples))
    def subtract(self, str2):
        self.check(str2)
        return Vector(i-j for i, j in zip(self.examples, str2.examples))
    def dot(self, str2):
        self.check(str2)
        return sum(i*j for i, j in zip(self.examples, str2.examples))
    def norm(self):
        return sqrt(sum(x**2 for x in self.examples))
    def equals(self, str2):
        return self.examples == str2.examples
```

第五题代码

```
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.index = 0
        self.progress = 0

    def inc_progress(self, rank):
        index = self.ranks.index(rank)
        if index == self.index:
            self.progress+=3
        elif index==self.index-1 or (index==1 and self.index==-1):
            self.progress+=1
        elif index > self.index:
            if(index>0 and self.index<0):
                x = index-self.index+1
            else:
                x = index-self.index
            self.progress +=10*x*x
        while self.progress >= 100:
            self.index += 1
            self.rank = self.ranks[self.index]
            self.progress -=100
        if self.rank ==8 :
            self.progress=0
        return
```

- [第三部分 使用Mermaid绘制类图](#)

第一题类图

Ship

Ship
-draft: int -crew: int
+__init__(self,draft: int, crew: int) +is_worth_it()

第二题类图

Block

Block
<div>-width: float</div> <div>-length: float</div> <div>-height: float</div>
<div>+__init__(self,args: list)</div> <div>+get_width()</div> <div>+get_length()</div> <div>+get_height()</div> <div>+get_volume()</div> <div>+get_surface_area()</div>

第三题类图

PaginationHelper

PaginationHelper
<div>-collection: list</div> <div>-items_per_page: int</div>
<div>+__init__(self,collection: list, items_per_page: int)</div> <div>+item_count()</div> <div>+page_count()</div> <div>+page_item_count(page_index: int)</div> <div>+page_index(item_index: int)</div>

第四题类图

Vector

Vector
-examples: tuple
+__init__(self,iterable) +__str__() +check(str2: Vector) +add(str2: Vector) +subtract(str2: Vector) +dot(str2: Vector) +norm() +equals(str2: Vector)

第五题类图

User

User
-ranks: list -rank: int -index: int -progress: int
+__init__() +inc_progress(rank: int) : : None

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辯以及实际的操作。

1. Python的类中__init__方法起什么作用？
- Python中的__init__方法是一个特殊的方法， 用于在创建一个对象时进行初始化操作。它在对象被

创建时自动调用，用于设置对象的初始状态，也可以接收参数来对对象进行初始化。

2. Python语言中如何继承父类和改写（override）父类的方法。

要继承父类并改写（override）父类的方法，可以在子类中定义一个同名的方法，就可以覆盖父类中的方法。例如：

```
class ParentClass:
    def some_method(self):
        print("父类")

class ChildClass(ParentClass):
    def some_method(self):
        print("子类覆盖父类的同名方法")
```

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。

特殊方法允许自定义类的行为从而实现特定的功能，例如创建可迭代对象、支持比较操作、实现上下文管理等。1、init() 方法：该方法在类实例化时被调用，用于初始化类的属性。

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

my_object = example('John', 30)
print(my_object.name) # Output: 'John'
print(my_object.age) # Output: 30
```

2、str() 方法：该方法返回类的字符串表示形式。

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

    def __str__(self):
        return f'My name is {self.name} and I am {self.age} years old.'

my_object = example('John', 30)
print(my_object) # Output: 'My name is John and I am 30 years old.'
```

3、add() 方法：该方法实现类的加法运算。

```
class example:
    def __init__(self, value):
        self.value = value

    def __add__(self, other):
        return example(self.value + other.value)

my_object1 = example(10)
my_object2 = example(20)
my_object3 = my_object1 + my_object2
print(my_object3.value)  # Output: 30
```

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。

在这次实验中，我学习了Python中的namedtuple和dataclass,用于创建具有命名字段的数据结构，能更好地提高代码的可读性和可维护性。除此之外，通过对Python的类与继承的进一步学习，加深了我对Python类和继承的理解。我学会了如何使用这些工具来定义和操作数据结构，提高了我的编程技巧，也转变了我的思维方式，使我能够更好地应用这些工具来解决实际的编程问题。