

# 实验七 Python面向对象编程

---

班级： 21计科1

学号： B20210302101

姓名： 金库

Github地址： <https://gitee.com/jinku111/jinku111.git>

CodeWars地址： <https://www.codewars.com/users/Jinku123>

---

## 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，"`crew`" 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缀，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

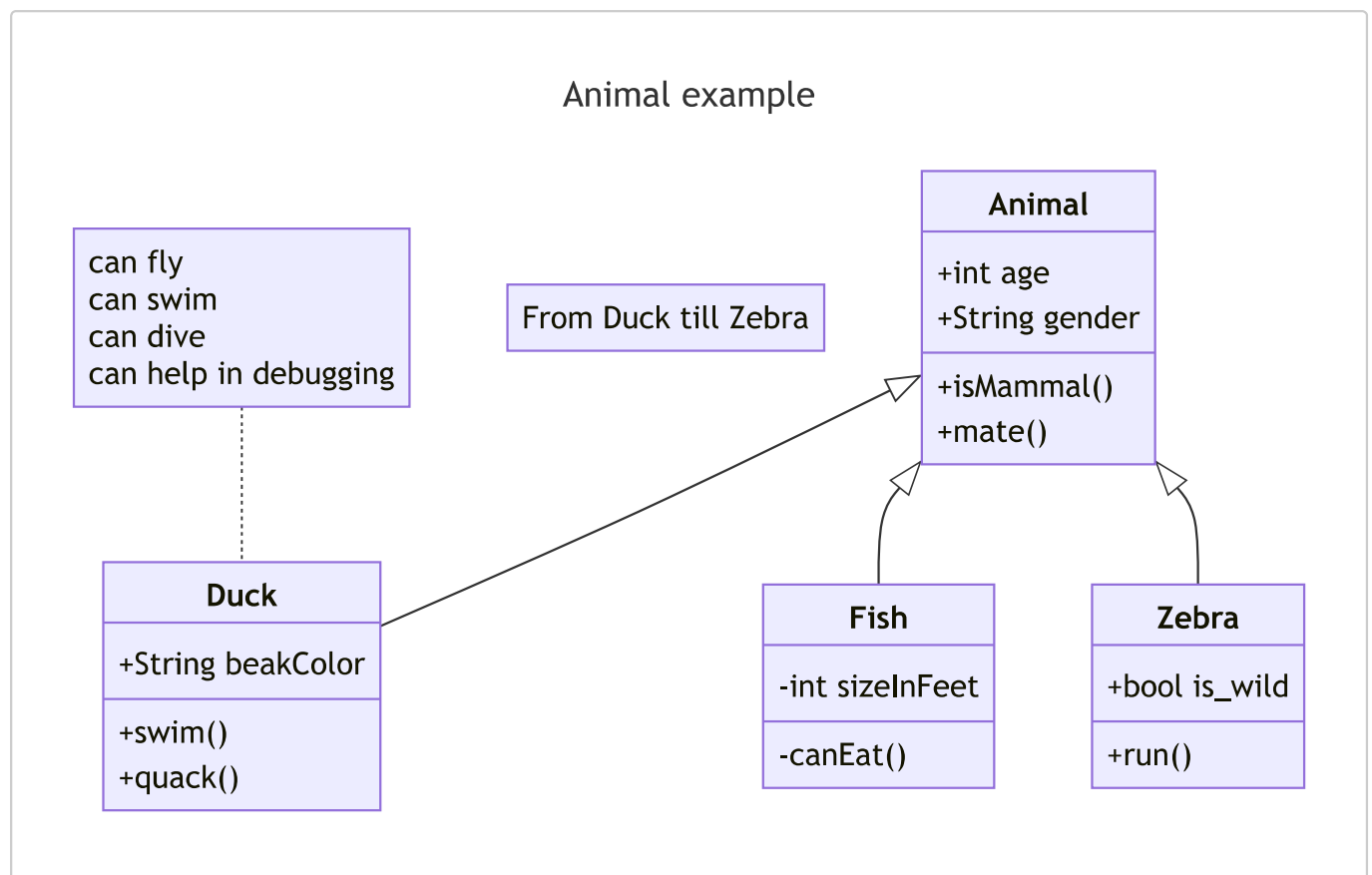
安装VSCode插件:

- 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:

 程序类图

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

```
class Ship:    # 第一题

    def __init__(self, draft, crew):
        self.draft = draft
        self.crew = crew

    def is_worth_it(self):
        return self.draft - self.crew * 1.5 > 20
、

```python
class Block:    # 第二题

    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)
```

```
import math    # 第三题

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)

    # 总页数
    def page_count(self):
```

```

# 总条目数 / 每页条目数, 然后向上取整
return math.ceil(self.item_count() / self.items_per_page)

def page_item_count(self, page_index):

    # 页数为负数或者页数超过总页数
    if page_index < 0 or page_index >= self.page_count():
        return -1

    # 最后一页
    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):
    # 非法的情况
    if item_index < 0 or item_index >= self.item_count():
        return -1
    else:
        return item_index // self.items_per_page

```

```

from math import sqrt      # 第四题
class Vector:

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

    # 把打印元组时的空格去掉
    def __str__(self):
        return str(self._v).replace(' ', '')

    # 检查两个向量是否长度相等
    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

```

```

class User ():
    # 第五题
    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

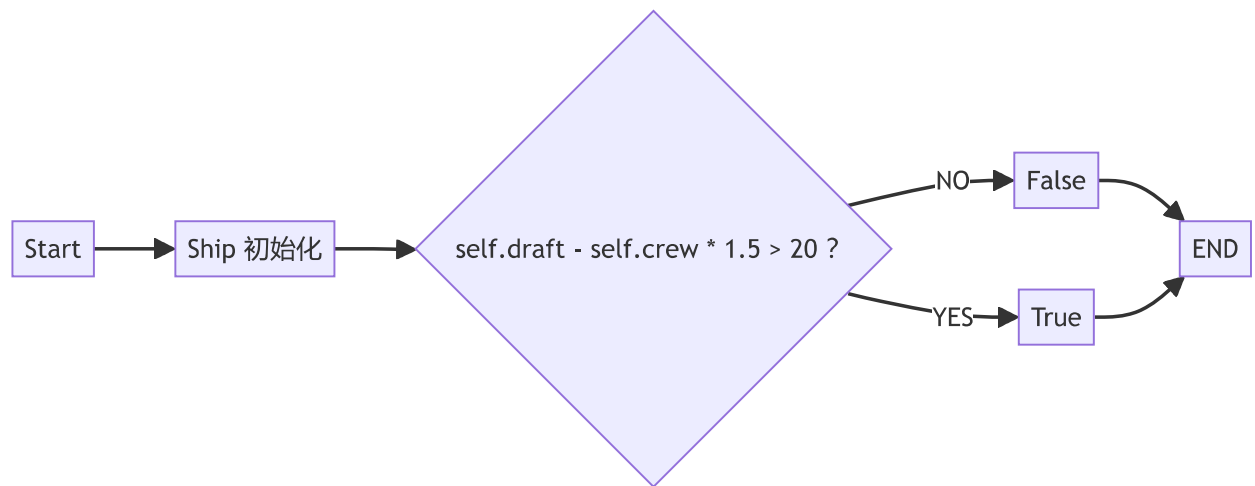
        # 如果进度大于100，升级，每减去100进度，升一级
        while self.progress >= 100:
            self.rank_index += 1
            self.rank = self.RANKS[self.rank_index]
            self.progress -= 100

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

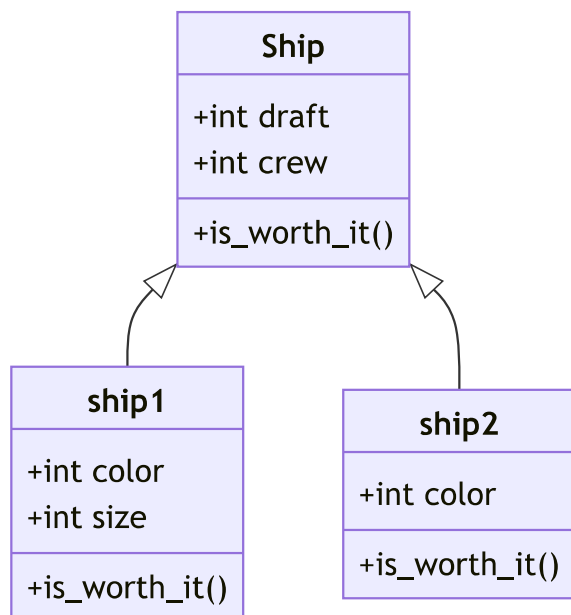
```

- [第三部分 使用Mermaid绘制程序流程图](#)

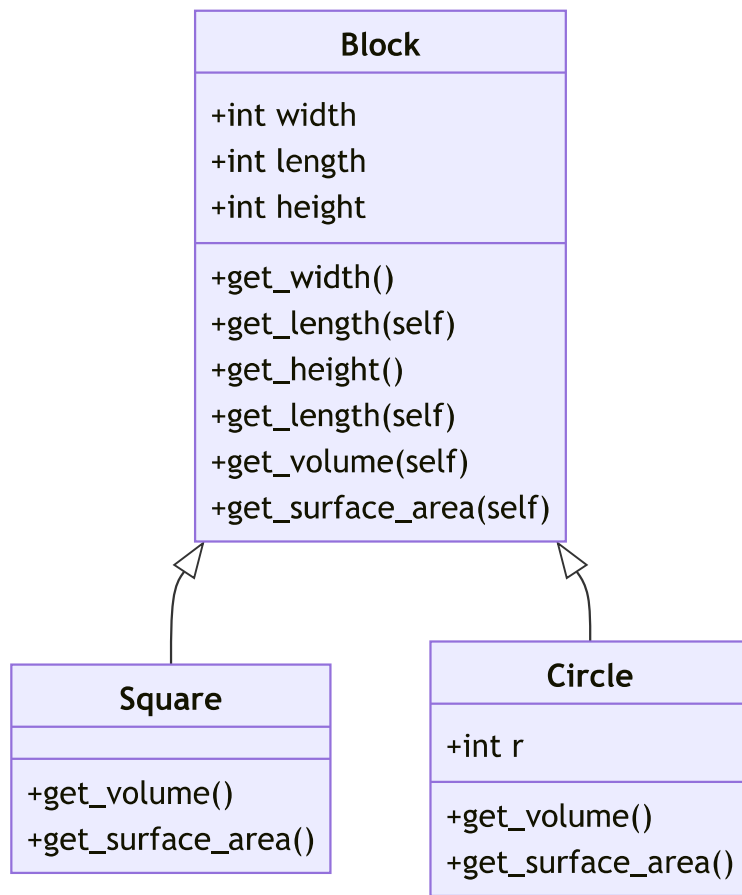




### Ship example



## Block example



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:]
```

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，用于在创建类的实例时进行初始化操作。它是类的构造函数，会在创建对象时自动调用。\_\_init\_\_方法允许我们在实例化对象时传入参数，并在对象创建时执行一些初始化操作，例如设置对象的属性或执行其他必要的操作。下面是一个简单的示例，演示了\_\_init\_\_方法的作用：

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
        print("A new person object has been created")
# 创建一个Person对象
person1 = Person("Alice", 25)
print(person1.name) # 输出: Alice
print(person1.age)  # 输出: 25
```

在这个例子中，\_\_init\_\_方法接受name和age两个参数，并将它们分别赋值给对象的属性。当我们创建Person对象时，\_\_init\_\_方法会自动调用，并执行初始化操作。这样，我们就可以在创建对象时传入必要的参数，并在对象创建时进行一些初始化设置

2. Python语言中如何继承父类和改写（override）父类的方法。

在Python中，要继承父类并改写（override）父类的方法，可以通过创建子类并在子类中重新定义父类的方法来实现。

下面是一个简单的示例，演示了如何继承父类并改写父类的方法：

```
# 定义父类
class Animal:
    def make_sound(self):
        print("Some generic sound")

# 定义子类，并继承父类
class Dog(Animal):
    # 改写（override）父类的方法
    def make_sound(self):
        print("Woof woof")

# 创建父类对象并调用方法
animal = Animal()
animal.make_sound() # 输出: Some generic sound

# 创建子类对象并调用方法
```

```
dog = Dog()
dog.make_sound() # 输出: Woof woof
```

在这个例子中，我们首先定义了一个父类Animal，其中包含一个make\_sound方法。然后我们定义了一个子类Dog，并在子类中重新定义了make\_sound方法，改写了父类中的方法。当我们创建父类对象和子类对象并调用make\_sound方法时，可以看到子类对象调用的是自己定义的方法，而不是父类的方法。

通过继承和改写父类的方法，我们可以在子类中定制特定的行为，实现多态性，并且保持代码的可维护性和复用性。

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，类有许多特殊的方法，也称为魔术方法或魔法方法。这些方法以双下划线（\_\_）开头和结尾，例如\_\_init\_\_、\_\_str\_\_、\_\_eq\_\_等。

下面是三个特殊方法的例子，以及它们的作用和简单的代码说明：

1. \_\_init\_\_方法：这是一个构造函数，用于在创建对象时初始化对象的属性。当我们创建一个新的对象时，Python会自动调用\_\_init\_\_方法，并传入对象本身和任何传递给构造函数的参数。这个方法通常用来初始化对象的属性。

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

    person = Person("Alice", 25)
    print(person.name) # 输出: Alice
    print(person.age) # 输出: 25
```

2. \_\_str\_\_方法：这个方法用于将对象转换为字符串。当我们使用print函数打印对象时，Python会自动调用\_\_str\_\_方法，并将其返回值打印出来。这个方法通常用来定义对象的字符串表示形式。

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

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

    person = Person("Alice", 25)
    print(person) # 输出: Person(name=Alice, age=25)
```

3. \_\_eq\_\_方法：这个方法用于比较两个对象是否相等。当我们使用==运算符比较两个对象时，Python会自动调用\_\_eq\_\_方法，并将另一个对象作为参数传递给它。这个方法通常用来定义对象的相等性。

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

    def __eq__(self, other):
        return self.name == other.name and self.age == other.age

person1 = Person("Alice", 25)
person2 = Person("Alice", 25)
print(person1 == person2)  # 输出: True
```

这些特殊方法可以让我们在类中定义更加灵活和强大的行为，使得我们的代码更加可读、可维护和易于扩展。

## 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，我学习和使用了以下知识：编程工具的使用：使用了Codewars平台进行编程练习和提交代码。数据结构：学习了Python中的类和继承的基础知识，以及如何使用namedtuple和DataClass来创建具有命名字段的数据结构。程序语言的语法：深入了解了Python中类和继承的语法和用法。算法：解决了关于数据结构和类的编程问题，包括如何使用类来组织和操作数据。编程技巧：学习了如何利用Python中的命名元组（namedtuple）和数据类（DataClass）来简化数据结构的定义和操作。编程思想：通过学习类和继承的知识，加深了对面向对象编程思想的理解。总的来说，这次实验让我掌握了Python类和继承的基础知识，并学会了如何使用命名元组和数据类来简化数据结构的定义和操作。