**长沙学院计算机科学与工程学院**

**实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 课程名称 | Python程序开发基础 | 班级 | 20软件01 | 实验日期 | 2023.4 |
| 姓名 | 胡锦华 | 学号 | B20200304125 | 指导教师 | 周景 |
| 成绩 |  | | | | |

### 一. 实验名称

Python高级实验

### 二. 实验目的及要求

第一部分（2学时）：

掌握Python函数，Lambda函数和gen函数的使用

第二部分（2学时）：

掌握Python面向对象编程，创建类和对象，编写类的构造函数和方法等等。

第三部分（2学时）：

掌握Python文件和异常，读取和写入文件，处理程序的异常，读取和写入json格式的文件。

### 三. 实验环境

Python 3.6+

### 四. 实验练习

**练习一：**

练习8-1：消息 编写一个名为display\_message() 的函数，它打印一个句子，指出你在本章学的是什么。调用这个函数，确认显示的消息正确无误。

练习8-3：T恤 编写一个名为make\_shirt() 的函数，它接受一个尺码以及要印到T恤上的字样。这个函数应打印一个句子，概要地说明T恤的尺码和字样。使用位置实参调用该函数来制作一件T恤，再使用关键字实参来调用这个函数。

练习8-4：大号T恤 修改函数make\_shirt() ，使其在默认情况下制作一件印有“I love Python”字样的大号T恤。调用这个函数来制作：一件印有默认字样的大号T恤，一件印有默认字样的中号T恤，以及一件印有其他字样的T恤（尺码无关紧要）。

练习8-7：专辑 编写一个名为make\_album() 的函数，它创建一个描述音乐专辑的字典。这个函数应接受歌手的名字和专辑名，并返回一个包含这两项信息的字典。使用这个函数创建三个表示不同专辑的字典，并打印每个返回的值，以核实字典正确地存储了专辑的信息。

给函数make\_album() 添加一个默认值为None 的可选形参，以便存储专辑包含的歌曲数。如果调用这个函数时指定了歌曲数，就将该值添加到表示专辑的字典中。调用这个函数，并至少在一次调用中指定专辑包含的歌曲数。

练习8-8：用户的专辑 在为完成练习8-7编写的程序中，编写一个while 循环，让用户输入专辑的歌手和名称。获取这些信息后，使用它们来调用函数make\_album() 并将创建的字典打印出来。在这个while 循环中，务必提供退出途径。

练习8-9：消息 创建一个列表，其中包含一系列简短的文本消息。将该列表传递给一个名为show\_messages() 的函数，这个函数会打印列表中的每条文本消息。

练习8-10：发送消息 在你为完成练习8-9而编写的程序中，编写一个名为send\_messages() 的函数，将每条消息都打印出来并移到一个名为sent\_messages 的列表中。调用函数send\_messages() ，再将两个列表都打印出来，确认正确地移动了消息。

练习8-11：消息归档 修改你为完成练习8-10而编写的程序，在调用函数

send\_messages() 时，向它传递消息列表的副本。调用函数send\_messages() 后，将两个列表都打印出来，确认保留了原始列表中的消息。

练习8-14：汽车 编写一个函数，将一辆汽车的信息存储在字典中。这个函数总是接受制造商和型号，还接受任意数量的关键字实参。这样调用该函数：提供必不可少的信息，以及两个名称值对，如颜色和选装配件。这个函数必须能够像下面这样进行调用：

car = make\_car('subaru', 'outback', color='blue', tow\_package=True)

打印返回的字典，确认正确地处理了所有的信息。

练习8-16：导入 选择一个你编写的且只包含一个函数的程序，将该函数放在另一个文件中。在主程序文件中，使用下述各种方法导入这个函数，再调用它：

import module\_name

from module\_name import function\_name

from module\_name import function\_name as fn

import module\_name as mn

from module\_name import \*

**练习二：**

练习9-1：餐馆 创建一个名为Restaurant 的类，为其方法\_\_init\_\_()设置属性restaurant\_name 和cuisine\_type 。创建一个名为describe\_restaurant() 的方法和一个名为open\_restaurant() 的方法，前者打印前述两项信息，而后者打印一条消息，指出餐馆正在营业。根据这个类创建一个名为restaurant 的实例，分别打印其两个属性，再调用前述两个方法。

练习9-4：就餐人数 在为完成练习9-1而编写的程序中，添加一个名为number\_served 的属性，并将其默认值设置为0。根据这个类创建一个名为restaurant 的实例。打印有多少人在这家餐馆就餐过，然后修改这个值并再次打印它。

添加一个名为set\_number\_served() 的方法，让你能够设置就餐人数。调用这个方法并向它传递一个值，然后再次打印这个值。

添加一个名为increment\_number\_served() 的方法，让你能够将就餐人数递增。调用这个方法并向它传递一个这样的值：你认为这家餐馆每天可能接待的就餐人数。

练习9-6：冰激凌小店 冰激凌小店是一种特殊的餐馆。编写一个名为IceCreamStand 的类，让它继承为完成练习9-1或练习9-4而编写的Restaurant 类。这两个版本的Restaurant 类都可以，挑选你更喜欢的那个即可。添加一个名为flavors 的属性，用于存储一个由各种口味的冰激凌组成的列表。编写一个显示这些冰激凌的方法。创建一个IceCreamStand 实例，并调用这个方法。

练习9-10：导入**Restaurant** 类 将最新的Restaurant 类存储在一个模块中。在另一个文件中，导入Restaurant 类，创建一个Restaurant 实例并调用Restaurant 的一个方法，以确认import 语句正确无误。

练习9-13：骰子 创建一个Die 类，它包含一个名为sides 的属性，该属性的默认值为6。编写一个名为roll\_die() 的方法，它打印位于1和骰子面数之间的随机数。创建一个6面的骰子再掷10次。

创建一个10面的骰子和一个20面的骰子，再分别掷10次。

练习9-14：彩票 创建一个列表或元组，其中包含10个数和5个字母。从这个列表或元组中随机选择4个数或字母，并打印一条消息，指出只要彩票上是这4个数或字母，就中大奖了。

练习9-15：彩票分析 可以使用一个循环来明白前述彩票大奖有多难中奖。为此，创建一个名为my\_ticket 的列表或元组，再编写一个循环，不断地随机选择数或字母，直到中大奖为止。请打印一条消息，报告执行循环多少次才中了大奖。

**练习三：**

练习10-1：Python学习笔记 在文本编辑器中新建一个文件，写几句话来总结一下你至此学到的Python知识，其中每一行都以“In Python you can”打头。将这个文件命名为learning\_python.txt，并存储到为完成本章练习而编写的程序所在的目录中。编写一个程序，它读取这个文件，并将你所写的内容打印三次：第一次打印时读取整个文件；第二次打印时遍历文件对象；第三次打印时将各行存储在一个列表中，再在with 代码块外打印它们。

练习10-3：访客 编写一个程序，提示用户输入名字。用户做出响应后，将其名字写入文件guest.txt中。

练习10-4：访客名单 编写一个while 循环，提示用户输入名字。用户输入名字后，在屏幕上打印一句问候语，并将一条到访记录添加到文件guest\_book.txt中。确保这个文件中的每条记录都独占一行。

练习10-6：加法运算 提示用户提供数值输入时，常出现的一个问题是，用户提供的是文本而不是数。在此情况下，当你尝试将输入转换为整数时，将引发ValueError 异常。编写一个程序，提示用户输入两个数，再将其相加并打印结果。在用户输入的任何一个值不是数时都捕获ValueError 异常，并打印一条友好的错误消息。对你编写的程序进行测试：先输入两个数，再输入一些文本而不是数。

练习10-7：加法计算器 将为完成练习10-6而编写的代码放在一个while 循环中，让用户犯错（输入的是文本而不是数）后能够继续输入数。

练习10-11：喜欢的数 编写一个程序，提示用户输入喜欢的数，并使用json.dump() 将这个数存储到文件中。再编写一个程序，从文件中读取这个值，并打印如下所示的消息。

I know your favorite number! It's \_\_\_\_\_.

练习10-12：记住喜欢的数 将练习10-11中的程序合二为一。如果存储了用户喜欢的数，就向用户显示它，否则提示用户输入喜欢的数并将其存储到文件中。运行这个程序两次，看看它能否像预期的那样工作。

### 五. 实验内容

前往[codewars.com](http://www.codewars.com)注册账号，完成下面的习题(或者自选题目)并且在题目的页面提交，通过题目的测试用例来获得分数，并将完成情况在表格中登记。

20软件1班：【腾讯文档】20软件1班Codewars登记

[20软件1班Codewars账号登记](https://docs.qq.com/sheet/DWXhEV2lNZVVhVUFP?tab=BB08J2)

20软件2班：【腾讯文档】20软件2班Codewars登记

[20软件2班Codewars账号登记](https://docs.qq.com/sheet/DWVJVV3Z2YXlnWWxR?tab=BB08J2)

1. 编码聚会 8kyu

你将得到一个字典数组，代表关于首次报名参加你所组织的编码聚会的开发者的数据。

你的任务是返回来自欧洲的JavaScript开发者的数量。

例如，给定以下列表：

lst1 = [

{ 'firstName': 'Noah', 'lastName': 'M.', 'country': 'Switzerland', 'continent': 'Europe', 'age': 19, 'language': 'JavaScript' },

{ 'firstName': 'Maia', 'lastName': 'S.', 'country': 'Tahiti', 'continent': 'Oceania', 'age': 28, 'language': 'JavaScript' },

{ 'firstName': 'Shufen', 'lastName': 'L.', 'country': 'Taiwan', 'continent': 'Asia', 'age': 35, 'language': 'HTML' },

{ 'firstName': 'Sumayah', 'lastName': 'M.', 'country': 'Tajikistan', 'continent': 'Asia', 'age': 30, 'language': 'CSS' }

]

你的函数应该返回数字1。

如果，没有来自欧洲的JavaScript开发人员，那么你的函数应该返回0。

注意：

字符串的格式将总是"Europe"和"JavaScript"。

所有的数据将始终是有效的和统一的，如上面的例子。

这个卡塔是Coding Meetup系列的一部分，其中包括一些简短易行的卡塔，这些卡塔是为了让人们掌握高阶函数的使用。在Python中，这些方法包括：filter, map, reduce。当然也可以采用其他方法来解决这些卡塔。

代码提交地址：

<https://www.codewars.com/kata/coding-meetup-number-1-higher-order-functions-series-count-the-number-of-javascript-developers-coming-from-europe>

1. 使用函数进行计算 5kyu

这次我们想用函数来写计算，并得到结果。让我们看一下一些例子：

seven(times(five())) # must return 35

four(plus(nine())) # must return 13

eight(minus(three())) # must return 5

six(divided\_by(two())) # must return 3

要求：

* 从0（"零"）到9（"九"）的每个数字都必须有一个函数。
* 必须有一个函数用于以下数学运算：加、减、乘、除。
* 每个计算都由一个操作和两个数字组成。
* 最外面的函数代表左边的操作数，最里面的函数代表右边的操作数。
* 除法应该是整数除法。例如，下面的计算应该返回2，而不是2.666666...。

eight(divided\_by(three()))

代码提交地址：

<https://www.codewars.com/kata/525f3eda17c7cd9f9e000b39>

1. Number Shortening Filter 6kyu

在这个kata中，我们将创建一个函数，它返回另一个缩短长数字的函数。给定一个初始值数组替换给定基数的 X 次方。如果返回函数的输入不是数字字符串，则应将输入本身作为字符串返回。

例子：

filter1 = shorten\_number(['','k','m'],1000)

filter1('234324') == '234k'

filter1('98234324') == '98m'

filter1([1,2,3]) == '[1,2,3]'

filter2 = shorten\_number(['B','KB','MB','GB'],1024)

filter2('32') == '32B'

filter2('2100') == '2KB';

filter2('pippi') == 'pippi'

代码提交地址：

<https://www.codewars.com/kata/56b4af8ac6167012ec00006f>

1. Hamming Numbers 4kyu

[Hamming number](https://en.wikipedia.org/wiki/Regular_number)是一个正整数，它的形式如2*i*3*j*5*k*, i、j、k都是非负整数。

具体来说：

* 最小的Hamming number 是 1 = 203050
* 第二小的Hamming number 是 2 = 213050
* 第三小的Hamming number 是 3 = 203150
* 第四小的Hamming number 是 4 = 223050
* 第五小的Hamming number 是 5 = 203051

测试用例中给出了20个最小的Hamming number。

你的代码应该能够计算出前5 000个的Hamming number而不超时。

代码提交地址：

<https://www.codewars.com/kata/526d84b98f428f14a60008da>

1. 面向对象编程：面向对象的海盗 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>

1. 搭建积木 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>

1. 分页助手 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>

1. 向量（Vector）类 5kyu

创建一个支持加法、减法、点积和向量长度的向量（Vector）类。因此，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>

1. Codewars风格的等级系统 **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。任何其他的值都应该引起错误。

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>

1. 淘气还是乖孩子 **7kyu**

圣诞老人要来镇上了，他需要你帮助找出谁是淘气的或善良的。你将会得到一整年的JSON数据，按照这个格式：

{

January: {

'1': 'Naughty','2': 'Naughty', ..., '31': 'Nice'

},

February: {

'1': 'Nice','2': 'Naughty', ..., '28': 'Nice'

},

...

December: {

'1': 'Nice','2': 'Nice', ..., '31': 'Naughty'

}

}

你的函数应该返回 "Naughty!"或 "Nice!"，这取决于在某一年发生的总次数（以较大者为准）。如果两者相等，则返回 "Nice！"。

代码提交地址：

<https://www.codewars.com/kata/5662b14e0a1fb8320a00005c>

1. 具有相同数字的下一个更大的数 **4kyu**

创建一个函数，接收一个正整数并返回通过重新排列其数字而形成的下一个更大的数字。比如说:

12 ==> 21

513 ==> 531

2017 ==> 2071

如果数字不能重新排列形成一个更大的数，则返回-1

9 ==> -1

111 ==> -1

531 ==> -1

代码提交地址：

<https://www.codewars.com/kata/55983863da40caa2c900004e>

1. RGB转16进制 **5kyu**

rgb函数是不完整的。完善它，使传入RGB十进制值的结果是返回一个十六进制的表示。RGB的有效十进制值是0-255。任何超出这个范围的数值必须四舍五入到最接近的有效数值。

注意：你的答案应该总是6个字符的长度，用3的速记法在这里不起作用。

以下是预期输出值的例子：

rgb(255, 255, 255) # returns FFFFFF

rgb(255, 255, 300) # returns FFFFFF

rgb(0,0,0) # returns 000000

rgb(148, 0, 211) # returns 9400D3

代码提交地址：

<https://www.codewars.com/kata/513e08acc600c94f01000001>

1. 有多少元素比我小II **3kyu**

给定一个数组arr，你必须返回一个数组，对于每个索引i处应该是数组的右边比arr[i]小的数量。例如：

smaller([5, 4, 3, 2, 1]) === [4, 3, 2, 1, 0]

smaller([1, 2, 0]) === [1, 1, 0]

代码提交地址：

<https://www.codewars.com/kata/56a1c63f3bc6827e13000006>

提示：

* 算法的时间复杂度应该是O(nlogn)
* 可以考虑改写归并排序或者二叉树（binary tree）的数据结构达到要求。

1. 观察到的PIN **4kyu**

好了，侦探，我们的一个同事成功地观察到了我们的目标人物，抢劫犯罗比。我们跟踪他到了一个秘密仓库，我们认为在那里可以找到所有被盗的东西。这个仓库的门被一个电子密码锁所保护。不幸的是，我们的间谍不确定他看到的密码，当罗比进入它时。

键盘的布局如下：

┌───┬───┬───┐

│ 1 │ 2 │ 3 │

├───┼───┼───┤

│ 4 │ 5 │ 6 │

├───┼───┼───┤

│ 7 │ 8 │ 9 │

└───┼───┼───┘

│ 0 │

└───┘

他注意到密码1357，但他也说，他看到的每个数字都有可能是另一个相邻的数字（水平或垂直，但不是对角线）。例如，代替1的也可能是2或4。而不是5，也可能是2、4、6或8。

他还提到，他知道这种锁。你可以无限制地输入错误的密码，但它们最终不会锁定系统或发出警报。这就是为什么我们可以尝试所有可能的（\*）变化。

\*可能的意义是：观察到的PIN码本身和考虑到相邻数字的所有变化。

你能帮助我们找到所有这些变化吗？如果有一个函数，能够返回一个列表，其中包含一个长度为1到8位的观察到的PIN的所有变化，那就更好了。我们可以把这个函数命名为getPINs（在python中为get\_pins，在C#中为GetPINs）。但请注意，所有的PINs，包括观察到的PINs和结果，都必须是字符串，因为有可能会有领先的 "0"。我们已经为你准备了一些测试案例。

侦探，我们就靠你了!

代码提交地址：

<https://www.codewars.com/kata/5263c6999e0f40dee200059d>

1. 疯狂的彩色三角形 **2kyu**

一个彩色的三角形是由一排颜色组成的，每一排都是红色、绿色或蓝色。连续的几行，每一行都比上一行少一种颜色，是通过考虑前一行中的两个相接触的颜色而产生的。如果这些颜色是相同的，那么新的一行就使用相同的颜色。如果它们不同，则在新的一行中使用缺失的颜色。这个过程一直持续到最后一行，只有一种颜色被生成。

例如，不同的可能性是：

Colour here: G G B G R G B R

Becomes colour here: G R B G

一个更大的三角形例子：

R R G B R G B B

R B R G B R B

G G B R G G

G R G B G

B B R R

B G R

R B

G

你将得到三角形的第一行字符串，你的工作是返回最后的颜色，这将出现在最下面一行的字符串。在上面的例子中，你将得到 "RRGBRGBB"，你应该返回 "G"。

限制条件： 1 <= length(row) <= 10 \*\* 5

输入的字符串将只包含大写字母'B'、'G'或'R'。

例子：

triangle('B') == 'B'

triangle('GB') == 'R'

triangle('RRR') == 'R'

triangle('RGBG') == 'B'

triangle('RBRGBRB') == 'G'

triangle('RBRGBRBGGRRRBGBBBGG') == 'G'

代码提交地址：

<https://www.codewars.com/kata/5a331ea7ee1aae8f24000175>

提示：请参考下面的链接，利用三进制的特点来进行计算。

<https://stackoverflow.com/questions/53585022/three-colors-triangles>

### 六. 实验过程和结果

1. 编码聚会 8kyu

代码：

|  |
| --- |
| def count\_developers(lst):      sums = 0      for l in lst:          if l['continent'] == 'Europe' and l['language'] == 'JavaScript':              sums += 1      return sums |

输出：

list1 = [

          { 'firstName': 'Noah', 'lastName': 'M.', 'country': 'Switzerland', 'continent': 'Europe', 'age': 19, 'language': 'JavaScript' },

          { 'firstName': 'Maia', 'lastName': 'S.', 'country': 'Tahiti', 'continent': 'Oceania', 'age': 28, 'language': 'JavaScript' },

          { 'firstName': 'Shufen', 'lastName': 'L.', 'country': 'Taiwan', 'continent': 'Asia', 'age': 35, 'language': 'HTML' },

          { 'firstName': 'Sumayah', 'lastName': 'M.', 'country': 'Tajikistan', 'continent': 'Asia', 'age': 30, 'language': 'CSS' }

        ]

print(count\_developers(list1)) # 1

1. 使用函数进行计算 5kyu

代码：

|  |
| --- |
| def zero(f = None):      if f:          return f(0)      else:          return 0  def one(f = None):      if f:          return f(1)      else:          return 1  def two(f = None):      if f:          return f(2)      else:          return 2  def three(f = None):      if f:          return f(3)      else:          return 3  def four(f = None):      if f:          return f(4)      else:          return 4  def five(f = None):      if f:          return f(5)      else:          return 5  def six(f = None):      if f:          return f(6)      else:          return 6  def seven(f = None):      if f:          return f(7)      else:          return 7  def eight(f = None):      if f:          return f(8)      else:          return 8  def nine(f = None):      if f:          return f(9)      else:          return 9  def plus(y):      return lambda x: x+y  def minus(y):      return lambda x: x-y  def times(y):      return lambda x: x\*y  def divided\_by(y):      return lambda x: int(x/y) |

输出：print(seven(times(five()))) # 35

1. Number Shortening Filter 6kyu

代码：

|  |
| --- |
| def shorten\_number(suffixes, base):      def shorten(suffix):          if not isinstance(suffix, str) or not suffix.isdigit():              return str(suffix)          l = len(suffixes)          x = int(suffix)          for i in range(l, 0, -1):              y = (base \*\* (i-1))              if x >= y:                  z = int(x / y)                  return f"{z}{suffixes[i-1]}"          return str(suffix)      return shorten |

输出：

filter1 = shorten\_number(['','k','m'],1000)

filter2 = shorten\_number(['','KB','MB','GB'],1024)

print((filter1('98234324'))) # 98m

print(filter2('1073741823')) # 1023MB

1. Hamming Numbers 4kyu

代码：

|  |
| --- |
| def hamming(n):      a,b,c = 0,0,0      r = [1]      for i in range(1, n):          x = r[a] \* 2          y = r[b] \* 3          z = r[c] \* 5          m = min(x, y, z)          r.append(m)          if m == x:              a += 1          if m == y:              b += 1          if m == z:              c += 1      return r[-1] |

输出：print(hamming(15)) # 24

1. 面向对象编程：面向对象的海盗 8kyu

代码：

|  |
| --- |
| 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 |

输出：

em = Ship(15, 10)

print(em.is\_worth\_it()) # False

1. 搭建积木 7kyu

代码：

|  |
| --- |
| class Block:      def \_\_init\_\_(self, q):          self.width = q[0]          self.length = q[1]          self.height = q[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.height \* self.length        def get\_surface\_area(self):          return (self.width \* self.height + self.height \* self.length + self.width \* self.length) \* 2 |

输出：

b = Block([2,4,6])

print(b.get\_width()) # 2

print(b.get\_length()) # 4

print(b.get\_height()) # 6

print(b.get\_volume()) # 48

print(b.get\_surface\_area()) # 88

1. 分页助手 5kyu

代码：

|  |
| --- |
| class PaginationHelper:        def \_\_init\_\_(self, collection, items\_per\_page):          self.items\_per\_page = items\_per\_page          self.collection = collection      def item\_count(self):          return len(self.collection)      def page\_count(self):          i = int(PaginationHelper.item\_count(self) / self.items\_per\_page)          j = int(PaginationHelper.item\_count(self) % self.items\_per\_page)          if j == 0:              return i          return i + 1      def page\_item\_count(self, page\_index):          if page\_index < 0 or page\_index >= PaginationHelper.page\_count(self):              return -1          if page\_index == PaginationHelper.page\_count(self) - 1:              return PaginationHelper.item\_count(self) - page\_index \* self.items\_per\_page          return self.items\_per\_page      def page\_index(self, item\_index):          if item\_index < 0 or item\_index >= PaginationHelper.item\_count(self):              return -1          return int(item\_index / self.items\_per\_page) |

输出：

helper = PaginationHelper(['a','b','c','d','e','f'], 4)

print(helper.page\_count()) # 2

print(helper.item\_count()) # 6

print(helper.page\_item\_count(0)) # 4

print(helper.page\_item\_count(1)) # 2

print(helper.page\_item\_count(2)) # -1

print(helper.page\_index(5)) # 1

print(helper.page\_index(2)) # 0

print(helper.page\_index(20)) # -1

print(helper.page\_index(-10)) # -1

1. 向量（Vector）类 5kyu

代码：

|  |
| --- |
| import operator  class Vector:      def \_\_init\_\_(self, collection):          self.collection = collection        def \_\_str\_\_(self):          return '(' + ','.join(str(i) for i in self.collection) + ')'        def equals(self, other):          return operator.eq(self.collection, other.collection)        def add(self, f = None):          if len(self.collection) == len(f.collection):              i = []              for j in self.collection:                  i.append(j)              for l in range(len(self.collection)):                  i[l] = self.collection[l] + f.collection[l]              return Vector(i)          raise IndexError()        def subtract(self, f):          if len(self.collection) == len(f.collection):              i = []              for j in self.collection:                  i.append(j)              for l in range(len(self.collection)):                  i[l] = self.collection[l] - f.collection[l]              return Vector(i)          raise IndexError()        def dot(self, f):          if len(self.collection) == len(f.collection):              sums = 0              i = []              for j in self.collection:                  i.append(j)              for l in range(len(self.collection)):                  i[l] = self.collection[l] \* f.collection[l]                  sums += i[l]              return sums          raise IndexError()        def norm(self):          sums = 0          i = []          for j in self.collection:              i.append(j)          for l in range(len(self.collection)):              i[l] \*\*= 2              sums += i[l]          return sums \*\* 0.5 |

输出：

a = Vector([1, 2, 3])

b = Vector([3, 4, 5])

print(a.add(b)) # (4,6,8)

print(a.subtract(b)) # (-2,-2,-2)

print(a.dot(b)) # 26

print(a.norm()) # 3.7416573867739413

1. Codewars风格的等级系统 4kyu

代码：

|  |
| --- |
| class User:      def \_\_init\_\_(self):          self.rank = -8          self.progress = 0        def inc\_progress(self, activity\_rank):          if activity\_rank not in range(-8, 0) and activity\_rank not in range(1, 9):              raise ValueError("Invalid rank value")            rank\_diff = activity\_rank - self.rank            if rank\_diff == 0:              progress = 3          elif rank\_diff == -1 or (activity\_rank == -1 and self.rank == 1):              progress = 1          elif rank\_diff >= 1:              if activity\_rank \* self.rank <= 0:                  progress = ((activity\_rank - self.rank - 1) \*\* 2) \* 10              else:                  progress = ((activity\_rank - self.rank) \*\* 2) \* 10          else:              progress = 0            self.progress += progress            pre\_rank = self.rank          while self.progress >= 100 and self.rank < 8:              self.progress -= 100              self.rank += 1          if self.rank \* pre\_rank <= 0:              self.rank += 1            if self.rank >= 8:              self.rank = 8              self.progress = 0 |

输出：

user = User()

user.inc\_progress(5)

print(user.progress) # 40

print(user.rank) # 7

1. 淘气还是乖孩子 7kyu

代码：

|  |
| --- |
| def naughty\_or\_nice(data):      i,j = 0,0      for item,a in data.items():          for key, value in a.items():              if value == 'Naughty':                  i += 1              if value == 'Nice':                  j += 1      if i <= j:          return 'Nice!'      return 'Naughty!' |

输出：

n = {"January": {"1": "Naughty", "2": "Nice", "3": "Naughty", "4": "Nice", "5": "Nice", "6": "Nice", "7": "Naughty", "8": "Nice", "9": "Nice", "10": "Naughty", "11": "Nice", "12": "Nice", "13": "Nice", "14": "Naughty", "15": "Naughty", "16": "Naughty", "17": "Nice", "18": "Nice", "19": "Naughty", "20": "Nice", "21": "Naughty", "22": "Nice", "23": "Naughty", "24": "Nice", "25": "Naughty", "26": "Nice", "27": "Nice", "28": "Naughty", "29": "Nice", "30": "Nice", "31": "Nice"},

     "February": {"1": "Nice", "2": "Naughty", "3": "Nice", "4": "Nice", "5": "Nice", "6": "Nice", "7": "Nice", "8": "Nice", "9": "Naughty", "10": "Naughty", "11": "Naughty", "12": "Nice", "13": "Nice", "14": "Naughty", "15": "Naughty", "16": "Nice", "17": "Nice", "18": "Naughty", "19": "Nice", "20": "Nice", "21": "Nice", "22": "Nice", "23": "Nice", "24": "Naughty", "25": "Naughty", "26": "Nice", "27": "Naughty", "28": "Nice"},

     "March": {"1": "Nice", "2": "Naughty", "3": "Nice", "4": "Nice", "5": "Nice", "6": "Naughty", "7": "Nice", "8": "Nice", "9": "Nice", "10": "Naughty", "11": "Naughty", "12": "Nice", "13": "Naughty", "14": "Naughty", "15": "Naughty", "16": "Nice", "17": "Nice", "18": "Nice", "19": "Naughty", "20": "Nice", "21": "Naughty", "22": "Naughty", "23": "Nice", "24": "Nice", "25": "Nice", "26": "Nice", "27": "Nice", "28": "Naughty", "29": "Nice", "30": "Nice", "31": "Naughty"},

     "April": {"1": "Naughty", "2": "Naughty", "3": "Nice", "4": "Nice", "5": "Nice", "6": "Naughty", "7": "Naughty", "8": "Nice", "9": "Nice", "10": "Nice", "11": "Nice", "12": "Nice", "13": "Naughty", "14": "Nice", "15": "Naughty", "16": "Naughty", "17": "Nice", "18": "Naughty", "19": "Nice", "20": "Naughty", "21": "Naughty", "22": "Nice", "23": "Nice", "24": "Naughty", "25": "Nice", "26": "Naughty", "27": "Naughty", "28": "Nice", "29": "Nice", "30": "Nice"},

     "May": {"1": "Nice", "2": "Naughty", "3": "Naughty", "4": "Nice", "5": "Nice", "6": "Nice", "7": "Naughty", "8": "Nice", "9": "Nice", "10": "Nice", "11": "Naughty", "12": "Naughty", "13": "Naughty", "14": "Naughty", "15": "Nice", "16": "Naughty", "17": "Naughty", "18": "Nice", "19": "Nice", "20": "Nice", "21": "Nice", "22": "Nice", "23": "Naughty", "24": "Naughty", "25": "Nice", "26": "Nice", "27": "Nice", "28": "Naughty", "29": "Naughty", "30": "Naughty", "31": "Nice"},

     "June": {"1": "Naughty", "2": "Nice", "3": "Naughty", "4": "Nice", "5": "Naughty", "6": "Nice", "7": "Nice", "8": "Nice", "9": "Nice", "10": "Naughty", "11": "Naughty", "12": "Nice", "13": "Nice", "14": "Naughty", "15": "Nice", "16": "Naughty", "17": "Naughty", "18": "Naughty", "19": "Nice", "20": "Nice", "21": "Nice", "22": "Nice", "23": "Nice", "24": "Nice", "25": "Nice", "26": "Nice", "27": "Nice", "28": "Nice", "29": "Naughty", "30": "Nice"},

     "July": {"1": "Nice", "2": "Nice", "3": "Nice", "4": "Naughty", "5": "Nice", "6": "Nice", "7": "Nice", "8": "Nice", "9": "Naughty", "10": "Nice", "11": "Nice", "12": "Naughty", "13": "Nice", "14": "Naughty", "15": "Nice", "16": "Nice", "17": "Naughty", "18": "Nice", "19": "Naughty", "20": "Nice", "21": "Nice", "22": "Nice", "23": "Nice", "24": "Naughty", "25": "Naughty", "26": "Nice", "27": "Naughty", "28": "Naughty", "29": "Nice", "30": "Nice", "31": "Nice"},

     "August": {"1": "Naughty", "2": "Nice", "3": "Naughty", "4": "Nice", "5": "Nice", "6": "Nice", "7": "Nice", "8": "Nice", "9": "Naughty", "10": "Naughty", "11": "Nice", "12": "Naughty", "13": "Nice", "14": "Naughty", "15": "Nice", "16": "Nice", "17": "Naughty", "18": "Nice", "19": "Naughty", "20": "Nice", "21": "Nice", "22": "Naughty", "23": "Naughty", "24": "Naughty", "25": "Naughty", "26": "Nice", "27": "Nice", "28": "Nice", "29": "Naughty", "30": "Naughty", "31": "Nice"},

     "September": {"1": "Naughty", "2": "Nice", "3": "Naughty", "4": "Nice", "5": "Nice", "6": "Nice", "7": "Nice", "8": "Naughty", "9": "Naughty", "10": "Nice", "11": "Naughty", "12": "Naughty", "13": "Nice", "14": "Naughty", "15": "Nice", "16": "Nice", "17": "Nice", "18": "Nice", "19": "Nice", "20": "Naughty", "21": "Nice", "22": "Nice", "23": "Nice", "24": "Nice", "25": "Nice", "26": "Naughty", "27": "Nice", "28": "Nice", "29": "Naughty", "30": "Nice"},

     "October": {"1": "Nice", "2": "Naughty", "3": "Naughty", "4": "Naughty", "5": "Naughty", "6": "Nice", "7": "Nice", "8": "Naughty", "9": "Nice", "10": "Nice", "11": "Naughty", "12": "Nice", "13": "Nice", "14": "Nice", "15": "Nice", "16": "Nice", "17": "Naughty", "18": "Naughty", "19": "Nice", "20": "Nice", "21": "Naughty", "22": "Nice", "23": "Nice", "24": "Naughty", "25": "Nice", "26": "Nice", "27": "Nice", "28": "Naughty", "29": "Naughty", "30": "Nice", "31": "Nice"},

     "November": {"1": "Naughty", "2": "Nice", "3": "Naughty", "4": "Nice", "5": "Nice", "6": "Nice", "7": "Nice", "8": "Nice", "9": "Nice", "10": "Nice", "11": "Nice", "12": "Naughty", "13": "Naughty", "14": "Naughty", "15": "Naughty", "16": "Nice", "17": "Naughty", "18": "Nice", "19": "Nice", "20": "Nice", "21": "Naughty", "22": "Naughty", "23": "Nice", "24": "Naughty", "25": "Naughty", "26": "Nice", "27": "Nice", "28": "Nice", "29": "Nice", "30": "Naughty"},

     "December": {"1": "Nice", "2": "Nice", "3": "Nice", "4": "Naughty", "5": "Nice", "6": "Naughty", "7": "Nice", "8": "Naughty", "9": "Nice", "10": "Naughty", "11": "Naughty", "12": "Naughty", "13": "Naughty", "14": "Naughty", "15": "Naughty", "16": "Nice", "17": "Nice", "18": "Nice", "19": "Naughty", "20": "Nice", "21": "Naughty", "22": "Naughty", "23": "Nice", "24": "Nice", "25": "Naughty", "26": "Nice", "27": "Nice", "28": "Nice", "29": "Nice", "30": "Nice", "31": "Nice"}}

print(naughty\_or\_nice(n)) # Nice!

1. 具有相同数字的下一个更小的数 4kyu

代码：

|  |
| --- |
| import itertools  def next\_smaller(n):      # l = [int(x) for x in str(n)]      # ls = sorted(l)      # if l == ls:      #     return -1      # all\_l = list(itertools.permutations(ls))      # j = all\_l.index(tuple(l))      # if all\_l[j - 1][0] == 0:      #     return -1      # return int(''.join(str(x) for x in all\_l[j - 1]))        digits = list(str(n))      i = len(digits) - 2      while i >= 0 and digits[i] <= digits[i+1]:          i -= 1      if i < 0:          return -1      j = i + 1      while j < len(digits) and digits[j] < digits[i]:          j += 1      j -= 1        print(digits)      digits[i], digits[j] = digits[j], digits[i]      digits[i+1:] = reversed(digits[i+1:])      result = int(''.join(digits))      if len(str(result)) != len(str(n)):          return -1      return result |

输出：

print(next\_smaller(17128)) # 12871

1. RGB转16进制 5kyu

代码：

|  |
| --- |
| def rgb(r, g, b):      dic = {0:'0', 1:'1', 2:'2', 3:'3', 4:'4', 5:'5', 6:'6', 7:'7', 8:'8', 9:'9',             10:'A', 11:'B', 12:'C', 13:'D', 14:'E', 15:'F'}      l = []      for i in (r, g, b):          if i < 0:              i = 0          if i > 255:              i = 255          l.append(dic[int(i / 16)])          i = i % 16          l.append(dic[i])      return ''.join(l) |

输出：print(rgb(234,255,0)) # EAFF00

### 七. 实验小结

经过这次实验，我学习了很多有用的知识和技能。在这个实验中，我掌握了如何使用类和对象、继承和多态、异常处理、文件操作、模块和包等Python的高级特性。我也发现了Python的优势和局限，以及如何在不同的场景中选择合适的工具和方法。我觉得这个实验对我的编程能力和思维方式有很大的提升，然后就是说多思考怎么用合理的算法去编写、优化代码，暴力枚举很容易超时。