

实验三 Python列表

班级： 21计科02

学号： B20210302221

姓名： 王日晖

Github地址： <https://github.com/wrrh>

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

实验目的

1. 学习Python的简单使用和列表操作
2. 学习Python中的if语句

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列表操作

完成教材《Python编程从入门到实践》下列章节的练习：

- 第3章 列表简介
 - 第4章 操作列表
 - 第5章 if语句
-

第二部分

在[Codewars网站](#)注册账号，完成下列Kata挑战：

第一题：3和5的倍数 (Multiples of 3 or 5)

难度： 6kyu

如果我们列出所有低于 10 的 3 或 5 倍数的自然数，我们得到 3、5、6 和 9。这些数的总和为 23。完成一个函数，使其返回小于某个整数的所有是3 或 5 的倍数的数的总和。此外，如果数字为负数，则返回 0。

注意：如果一个数同时是3和5的倍数，应该只被算一次。

提示：首先使用列表解析得到一个列表，元素全部是3或者5的倍数。使用sum函数可以获取这个列表所有元素的和。

代码提交地址：<https://www.codewars.com/kata/514b92a657cdc65150000006>

第二题：重复字符的编码器 (Duplicate Encoder)

难度：6kyu

本练习的目的是将一个字符串转换为一个新的字符串，如果新字符串中的每个字符在原字符串中只出现一次，则为"("，如果该字符在原字符串中出现多次，则为")"。在判断一个字符是否是重复的时候，请忽略大写字母。

例如：

```
"din"      => "((("
"recede"   => "()()())"
"Success"  => ")()())()"
"(( @"     => "))((("
```

代码提交地址：<https://www.codewars.com/kata/54b42f9314d9229fd6000d9c>

第三题：括号匹配 (Valid Braces)

难度：6kyu

写一个函数，接收一串括号，并确定括号的顺序是否有效。如果字符串是有效的，它应该返回True，如果是无效的，它应该返回False。例如：

```
"(){}[]" => True
"([{}])" => True
"{}"     => False
"[]"     => False
"[({})]" => False
```

提示：python中没有内置堆栈数据结构，可以直接使用list来作为堆栈，其中append方法用于入栈，pop方法可以出栈。

代码提交地址：<https://www.codewars.com/kata/5277c8a221e209d3f6000b56>

第四题：从随机三元组中恢复秘密字符串(Recover a secret string from random triplets)

难度：4kyu

有一个不为你所知的秘密字符串。给出一个随机三个字母的组合的集合，恢复原来的字符串。

这里的三个字母的组合被定义为三个字母的序列，每个字母在给定的字符串中出现在下一个字母之前。"whi"是字符串 "whatisup" 的一个三个字母的组合。

作为一种简化，你可以假设没有一个字母在秘密字符串中出现超过一次。

对于给你的三个字母的组合，除了它们是有效的三个字母的组合以及它们包含足够的信息来推导出原始字符串之外，你可以不做任何假设。特别是，这意味着秘密字符串永远不会包含不出现在给你的三个字母的组合中的字母。

测试用例：

```
secret = "whatisup"
triplets = [
    ['t', 'u', 'p'],
    ['w', 'h', 'i'],
    ['t', 's', 'u'],
    ['a', 't', 's'],
    ['h', 'a', 'p'],
    ['t', 'i', 's'],
    ['w', 'h', 's']
]
test.assert_equals(recoverSecret(triplets), secret)
```

代码提交地址：<https://www.codewars.com/kata/53f40dff5f9d31b813000774/train/python>

提示：

- 利用集合去掉 `triplets` 中的重复字母，得到字母集合 `letters`，最后的 `secret` 应该由集合中的字母组成，`secret` 长度也等于该集合。

```
letters = {letter for triplet in triplets for letter in triplet}
length = len(letters)
```

- 创建函数 `check_first_letter(triplets, first_letter)`，检测一个字母是不是 `secret` 的首字母，返回 `True` 或者 `False`。
- 创建函数 `remove_first_letter(triplets, first_letter)`，从三元组中去掉首字母，返回新的三元组。
- 遍历字母集合 `letters`，利用上面2个函数得到最后的结果 `secret`。

第五题：去掉喷子的元音 (Disemvowel Trolls)

难度：7kyu

喷子正在攻击你的评论区！处理这种情况的一个常见方法是删除喷子评论中的所有元音(字母：a,e,i,o,u)，以消除威胁。你的任务是写一个函数，接收一个字符串并返回一个去除所有元音的新字符串。例如，字符串 "This website is for losers LOL!" 将变成 "Ths wbst s fr lsrs LL!"。

注意：对于这个Kata来说，y不被认为是元音。代码提交地址：

<https://www.codewars.com/kata/52fba66badcd10859f00097e>

提示：

- 首先使用列表解析得到一个列表，列表中所有不是元音的字母。
- 使用字符串的join方法连结列表中所有的字母，例如：

```
last_name = "lovelace"
letters = [letter for letter in last_name ]
print(letters) # ['l', 'o', 'v', 'e', 'l', 'a', 'c', 'e']
name = ''.join(letters) # name = "lovelace"
```


第三部分

使用Mermaid绘制程序流程图

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流程图（至少一个），Markdown代码如下：

 程序流程图

显示效果如下：

```
graph LR
    A[Start] --> B{Is it?}
    B -->|Yes| C[OK]
    C --> D[Rethink]
    D --> B
    B -.->|No| E[End]
```

查看Mermaid流程图语法-->[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列表操作和if语句](#)
- [第二部分 Codewars Kata挑战](#)
- [第三部分 使用Mermaid绘制程序流程图](#)

如果是Python代码，应该使用下面代码块格式，例如：

第一题：3和5的倍数（Multiples of 3 or 5）

显示效果如下：

```
def solution(number):
    if number < 0:
        return 0

    list = []

    for i in range(1,number):
        if i % 3 == 0 or i % 5 == 0:
            list.append(i)

    return sum(list)
```

第一题：3和5的倍数（Multiples of 3 or 5） [程序流程图]

显示效果如下：

```
flowchart LR
    A[Start] --> B{将单词转为小写}
    B --> C[初始化新单词字符串]
    C --> D[遍历单词中的每个字符]
    D --> E[判断字符在单词中出现的次数]
    E ---->|出现次数为1| F[将新单词字符串添加 ( ]
    E ---->|出现次数大于1| G[将新单词字符串添加 ) ]
    F --> D
    G --> D
    D --> H[判断是否遍历完所有字符]
    H ---->|是| I[返回新单词字符串]
    H ---->|否| D
    B ---->|No| E[End]
```

第二题：重复字符的编码器（Duplicate Encoder）

显示效果如下：

```
def duplicate_encode(word):
    word = word.lower()
    new_word = ""
    for char in word:
        if word.count(char) == 1:
            new_word += "("
        else:
```

```
        new_word += ")"
    return new_word
```

第三题：括号匹配 (Valid Braces)

显示效果如下：

```
def validBraces(braces):
    stack = []
    for char in braces:
        if char == "(" or char == "[" or char == "{":
            stack.append(char)
        elif char == ")" and stack and stack[-1] == "(":
            stack.pop()
        elif char == "]" and stack and stack[-1] == "[":
            stack.pop()
        elif char == "}" and stack and stack[-1] == "{":
            stack.pop()
        else:
            return False
    return len(stack) == 0
```

第四题：从随机三元组中恢复秘密字符串(Recover a secret string from random triplets)

显示效果如下：

第五题：去掉喷子的元音 (Disemvowel Trolls)

显示效果如下：

```
def disemvowel(string):
    vowels = "aeiouAEIOU"
    new_string = "".join([char for char in string if char not in vowels])
    return new_string
```

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中的列表可以进行哪些操作？

Python中的列表可以进行以下操作：

添加元素：使用`append()`方法将元素添加到列表的末尾，使用`insert()`方法在指定位置插入元素。

删除元素：使用`remove()`方法通过元素的值删除列表中的元素，使用`pop()`方法通过索引删除列表中的元素。

访问元素：使用索引来访问列表中的元素，索引从0开始。

切片操作：使用切片操作来获取列表中的子列表。

修改元素：使用索引来修改列表中的元素。

长度和计数：使用`len()`函数获取列表的长度，使用`count()`方法计算列表中某个元素的出现次数。

迭代和循环：使用`for`循环遍历列表中的元素。

列表合并：使用`+`操作符或`extend()`方法将两个列表合并成一个列表。

列表排序：使用`sort()`方法对列表进行排序。

列表反转：使用`reverse()`方法将列表中的元素反转。

2. 哪两种方法可以用来对Python的列表排序？这两种方法有和区别？

两种对Python列表进行排序的方法是使用`sort()`方法和`sorted()`函数。它们的区别在于：

`sort()`方法是对原列表进行排序，改变了列表本身，不返回新的列表。

`sorted()`函数是返回一个新的排序后的列表，不改变原列表。

3. 如何将Python列表逆序打印？

可以使用切片操作`[::-1]`来将Python列表逆序打印。

4. Python中的列表执行哪些操作时效率比较高？哪些操作效率比较低？是否有类似的数据结构可以用来替代列表？

Python中列表的操作效率比较高的包括：

访问元素：由于列表使用了连续的内存空间，通过索引访问元素的时间复杂度为 $O(1)$ 。

添加元素：在列表末尾添加元素的时间复杂度为 $O(1)$ ，在指定位置插入元素的时间复杂度为 $O(n)$ 。

列表的操作效率比较低的包括：

删除元素：通过值删除元素的时间复杂度为 $O(n)$ ，通过索引删除元素的时间复杂度为 $O(n)$ 。

切片操作：切片操作会创建一个新的列表，如果原列表很大，切片操作的时间和空间复杂度都较高。

Python中可以用数组（array）来替代列表，数组的元素类型必须相同，数组的访问速度更快，但是数组的长度固定，不支持动态增加和删除元素。

5. 阅读《Fluent Python》Chapter 2. An Array of Sequence - Tuples Are Not Just Immutable Lists小节（p30-p35）。总结该小节的主要内容。

Fluent Python》Chapter 2. An Array of Sequence - Tuples Are Not Just Immutable Lists小节主要介绍了元组的特点和用法。元组是不可变的序列，可以包含任意类型的元素，并且可以

通过索引访问元素。元组的优点是占用的内存较小，访问速度较快。元组还可以用作字典的键、集合的元素或其他数据结构的元素。元组可以通过拆包和命名元组来优雅地处理数据。拆包是将元组的元素赋值给多个变量的操作，可以简化代码。命名元组是一个具有字段名的元组，可以通过字段名访问元组的元素，提高了代码的可读性。

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。

通过这次实验，我学习了Markdown语法的基本用法。同时，我也巩固了Python编程语言的知识，包括简单数据类型的操作和变量的特点。此外，我还学习了提高代码可读性的一些方法，这对于编写清晰、易懂的代码是非常重要的。

在今后的学习和工作中，我将继续运用Markdown语法来编写文档和笔记，以提高自己的文档整理和表达能力。同时，我也会继续深入学习Python编程语言，不断提升自己的编程技巧和解决问题的能力。

在编程实践中，我会更加注重代码的可读性和可维护性，尽量遵循良好的编码习惯和规范，使代码更易于理解和修改。我也会持续学习和掌握更多的数据结构、算法和编程技巧，以应对各种实际问题 and 挑战。

总之，通过这次实验，我不仅学到了Markdown语法的使用和Python编程的知识，还培养了自己的思考和解决问题的能力。我会将这些知识和技能应用到实际中，不断进步和成长。