实验三 Python列表

班级： 21计科3

学号： B2021032321

姓名： 向钟源

Github地址：<Ch1rs (github.com)>

CodeWars地址：<Ch1rs_X | Codewars>

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

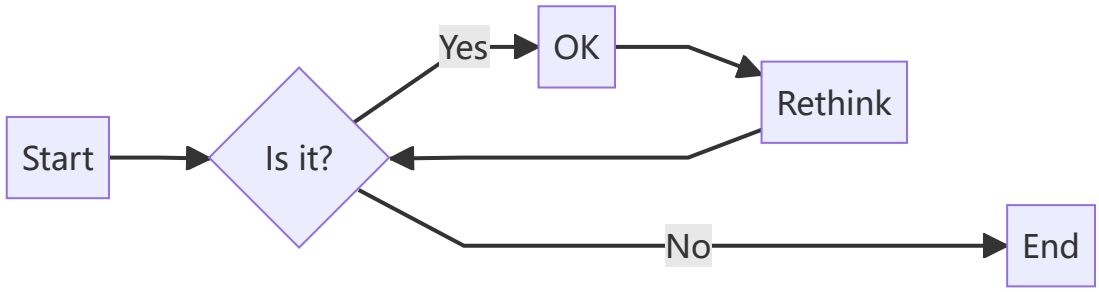
安装VSCode插件：

- 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：

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

显示效果如下：



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

略

- 第二部分 Codewars 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.

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

    list1 = []
    for i in range(1, number):
        if i % 3 == 0 or i % 5 == 0:
            list1.append(i)
    return sum(list1)
```

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"  =>  ")())())"
"(( @"     =>  "))(("
```

```
def duplicate_encode(word):
    #your code here
    word = word.lower()  # 转换为小写，以便忽略大小写

    count = {}
    for char in word:
```

```
        if char in count:
            count[char] += 1
        else:
            count[char] = 1

    result = ''
    for char in word:
        if count[char] > 1:
            result += ')'
        else:
            result += '('

    return result
```

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。

```
def valid_braces(string):
    stack = []
    matching_brackets = {'(': ')', '[': ']', '{': '}'}

    for char in string:
        if char in {'(', '[', '{'}:
            stack.append(char)
        elif char in {')', ']', '}'}:
            if len(stack) == 0 or matching_brackets[stack.pop()] != char:
                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)

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。

```
def recoverSecret(triplets):
    'triplets is a list of triplets from the secrent string. Return the string.'
    lic = []
    for i in range(len(triplets)):
        for j in range(len(triplets[i])):
            lic.append(triplets[i][j])

    lic = set(lic)
    lic = list(lic)
    while True:
        cnt = 0
        for i in range(len(triplets)):
            for j in range(len(triplets[i]) - 1):
                a = triplets[i][j]
                b = triplets[i][j + 1]
                index_a = lic.index(a)
                index_b = lic.index(b)
                if index_a > index_b:
                    temp = lic[index_b]
                    lic[index_b] = lic[index_a]
                    lic[index_a] = temp
                    cnt += 1
        if cnt == 0:
            return ''.join(lic)
```

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。

```
def disemvowel(string_):
    vowels = "aeiouAEIOU"
    return "".join(char for char in string_ if char not in vowels);
```

- 第三部分 使用Mermaid绘制程序流程图

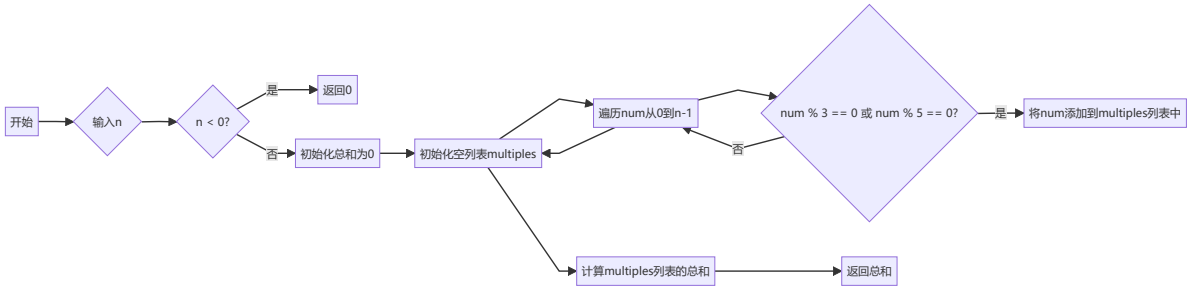
第一题：3和5的倍数（Multiples of 3 or 5）的流程图

难度： 6kyu

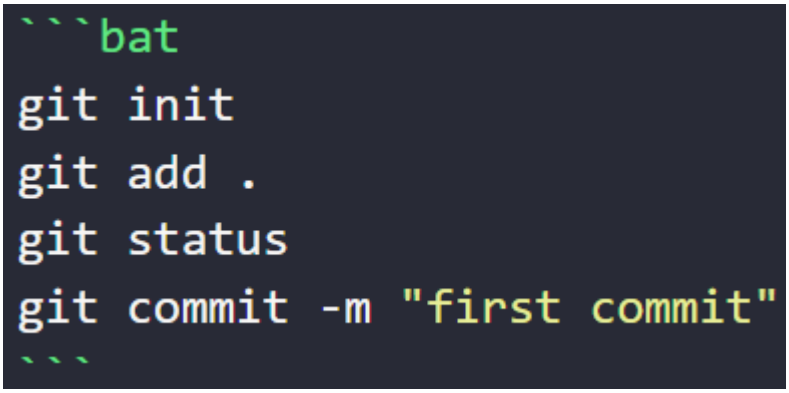
如果我们列出所有低于 10 的 3 或 5 倍数的自然数，我们得到 3、5、6 和 9。这些数的总和为 23. 完成一个函数，使其返回小于某个整数的所有是3 或 5 的倍数的数的总和。此外，如果数字为负数，则返回 0。

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.



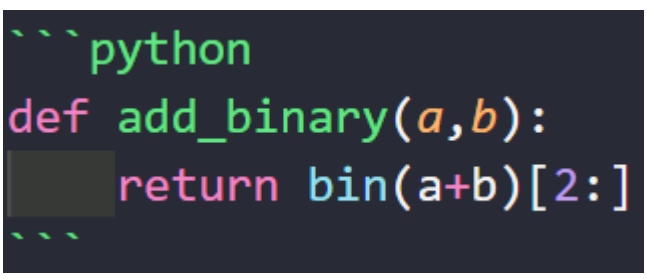
注意代码需要使用markdown的代码块格式化，例如Git命令行语句应该使用下面的格式：



显示效果如下：

```bat
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中的列表可以进行哪些操作?

答：

1. **访问元素：可以使用索引操作符 `[]` 来访问列表中的元素。例如：`my_list[0]` 访问第一个元素。**
2. **切片：可以使用切片操作符 `[:]` 来获取列表的子列表。例如：`my_list[1:4]` 获取索引从1到3的子列表。**
3. **连接：可以使用加号运算符 `+` 将两个列表连接成一个新的列表。例如：`list1 + list2`。**
4. **重复：可以使用乘号运算符 `*` 将列表重复多次。例如：`my_list *3` 会将列表元素重复三次。**
5. **修改元素：可以通过索引对列表中的元素进行修改。例如：`my_list[0] = 10` 将第一个元素修改为10。**
6. **添加元素：可以使用 `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。**
7. **删除元素：可以使用 `del` 语句或 `remove()` 方法删除列表中的元素。**
8. **长度：可以使用 `len()` 函数获取列表的长度（即列表中元素的个数）。**
9. **排序：可以使用 `sort()` 方法对列表进行排序。也可以使用 `sorted()` 函数返回一个排序后的新列表**

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?

答：`sort()` 方法：`sort()` 方法是列表对象的一个方法，可以直接调用。它会在原地修改列表，将列表元素按照升序进行排序。

`sorted()` 函数：`sorted()` 函数是Python的内置函数，可以接受一个可迭代对象作为参数，并返回一个新的已排序的列表。它不会修改原始列表，而是返回一个新的列表。

3. 如何将Python列表逆序打印?

   答：可以使用 `reverse()` 方法对列表进行反转，然后使用循环或 `print()` 函数逐个打印列表元素。

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

   答：在Python中，列表执行以下操作时效率较高：

   1. **访问元素：通过索引直接访问列表中的元素，例如 `my_list[0]`。**
   2. **添加元素：使用 `append()` 方法将元素添加到列表的末尾。**
   3. **扩展列表：使用 `extend()` 方法或 `+` 操作符将另一个列表的元素添加到当前列表的末尾。**
   4. **切片访问：使用切片操作符 `[:]` 访问列表的子列表。**

5. **列表解析：使用列表解析语法创建新的列表。**

这些操作的时间复杂度通常是O(1)或O(n)，其中n是列表的长度。

然而，列表在以下操作上效率较低：

1. **插入和删除元素：在列表中间插入或删除元素，特别是当列表很大时，这些操作的时间复杂度为O(n)。因为在插入或删除元素后，需要移动后续元素来填补空位。**
2. **查找元素：如果需要在列表中查找特定元素，需要遍历整个列表，时间复杂度为O(n)。**

如果对于插入和删除操作的效率要求较高，可以考虑使用其他数据结构来替代列表，例如：

1. **链表（Linked List）：链表是一种动态数据结构，插入和删除操作的时间复杂度为O(1)。但是，访问元素的效率相对较低，需要遍历链表来找到指定位置的元素。**
2. **双向链表（Doubly Linked List）：双向链表在链表的基础上增加了前向指针，使得在插入和删除操作时能够更高效地操作前后元素。**
3. **数组（Array）：数组在插入和删除操作上性能较差，但是在访问元素时效率很高，时间复杂度为O(1)。如果需要频繁地访问元素而不进行插入和删除操作，可以选择使用数组**

1. 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。

   答：

   1. **元组是不可变序列：元组是一个有序的不可变容器，一旦创建就不能修改。元组可以包含任意类型的元素，并通过逗号分隔，通常用圆括号括起来。**
   2. **元组的创建：可以使用圆括号来创建元组，也可以省略圆括号直接用逗号分隔的方式创建元组。例如：`my_tuple = (1, 2, 3)` 或 `my_tuple = 1, 2, 3`。**
   3. **元组的解包：可以通过将元组赋值给多个变量来解包元组，变量的个数必须与元组的元素个数相匹配。例如：`x, y, z = my_tuple`。**
   4. **元组的特性：元组具有不可变性，因此在不需要修改元素的情况下更适合用作多个值的集合。元组还可以作为字典的键值、集合的元素和函数的参数。**
   5. **元组的优势：与列表相比，元组在内存使用和性能方面更加高效。元组占用的内存较小，且不可变性使得元组成为字典的理想键值。**
   6. **元组的应用场景：元组常用于表示一组相关的数据，例如日期和时间、坐标、数据库记录等。它们还可以用于函数的返回值，可以一次返回多个值。**

   总的来说，元组是不可变的有序序列，适用于存储不会变化的一组值，具有较小的内存占用和更高的性能。与列表相比，元组在某些场景下更加适用。

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。

通过实验，你应该对Python的基本语法和列表操作有了一定的了解。我学会了如何运行Python代码、使用变量和基本数据类型、掌握了条件语句和循环语句的使用，以及掌握了列表的创建和常见操作。这些知识是学习和应用Python的基础，为进一步深入学习和实践打下了坚实的基础。