实验三 Python列表

班级： 21计科3班

学号： B20210302326

姓名： 李俊瑜

Github地址：https://gitee.com/Yukilm/python_exp

CodeWars地址：https://www.codewars.com/users/Yukilim

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

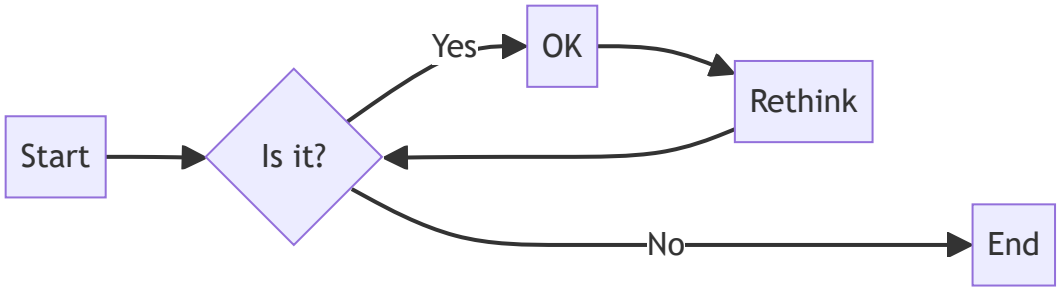
安装VSCode插件：

- 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：



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

```
# 练习 3-6: 添加嘉宾 你刚找到了一个更大的餐桌 可容纳更多的嘉宾 请想想你还想邀请哪三位嘉宾
# 以完成练习 3-4或练习 3-5时编写的程序为基础 在程序末尾添加一条print语句指出你找到了一个更大的餐桌
# 使用insert() 将一位新嘉宾添加到名单开头
# 使用insert() 将另一位新嘉宾添加到名单中间
# 使用append() 将最后一位新嘉宾添加到名单末尾
# 打印一系列消息， 向名单中的每位嘉宾发出邀请
dinner = ['a', 'b', 'c']
dinner.insert(0,'ak')
dinner.insert(1,'bk')
dinner.append('ck')
for i in dinner:
    print('I want to have a dinner with' + ' ' + i)
```

```
PS D:\Python\python_exp> & D:/Python/python.exe d:/Python/python_exp/bk_practice3.py
I want to have a dinner with ak
I want to have a dinner with bk
I want to have a dinner with a
I want to have a dinner with b
I want to have a dinner with c
I want to have a dinner with ck
```

```python
# 练习 3-7: 缩减名单 你刚得知新购买的餐桌无法及时送达 因此只能邀请两位嘉宾
# 以完成练习 3-6时编写的程序为基础 在程序末尾添加一行代码 打印一条你只能邀请两位嘉宾共进晚餐的消息
# 使用pop() 不断地删除名单中的嘉宾 直到只有两位嘉宾为止 每次从名单中弹出一位嘉宾时 都打印一条消息 让该嘉
# 对于余下的两位嘉宾中的每一位 都打印一条消息 指出他依然在受邀人之列
# 使用del 将最后两位嘉宾从名单中删除 让名单变成空的 打印该名单 核实程序结束时名单确实是空的
dinner = ['a', 'b', 'c']
dinner.insert(0, 'ak')
dinner.insert(1, 'bk')
dinner.append('ck')
print('Sorry, I only invite two human to have a dinner')
print(dinner)
for i in range(2):
    name = dinner.pop(0)
    print(name + ' ' + 'Sorry')
n = len(dinner)
for i in range(2, a):
    name = dinner.pop()
    print(name + ' ' + 'Sorry')
del dinner[1]
del dinner[0]
print(dinner)
```

```
PS D:\Python\python_exp> & D:/Python/python.exe d:/Python/python_exp/bk_practice3.py
Sorry, I only invite two human to have a dinner
['ak', 'bk', 'a', 'b', 'c', 'ck']
ak Sorry
bk Sorry
ck Sorry
c Sorry
[]
```

```python
# 练习4-9：立方推导式  使用列表推导式生成一个列表，其中包含前10个整数的立方。
for i in [i**3 for i in range(1, 11)]:
    print(i, end=' ')
```

```
PS D:\Python\python_exp> & D:/Python/python.exe d:/Python/python_exp/bk_practice3.py
1 8 27 64 125 216 343 512 729 1000
```

```python
# 练习4-10：切片  选择你在本章编写的一个程序 在末尾添加几行代码 以完成如下任务
# 打印消息“The first three items in the list are:” 再使用切片来打印列表的前三个元素
# 打印消息“Three items from the middle of the list are:” 再使用切片来打印列表的中间三个元素
# 打印消息“The last three items in the list are:” 再使用切片来打印列表的末尾三个元素
numbers = [i**3 for i in range(1, 11)]
print(numbers)
print(f'The first three items in the list are: {numbers[:3]}')
print(f'Three items from the middle of the list are: {numbers[(len(numbers)//2 -1):(len(numbers)
print(f'The last three items in the list are: {numbers[-3:]}')\
```

```
PS D:\Python\python_exp> & D:/Python/python.exe d:/Python/python_exp/bk_practice3.py
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
The first three items in the list are: [1, 8, 27]
Three items from the middle of the list are: [125, 216, 343]
The last three items in the list are: [512, 729, 1000]
```

```python
# 练习4-11：你的比萨，我的比萨  在你为完成练习4-1而编写的程序中 创建比萨列表的副本
# 并将其赋给变量friend_pizzas 再完成如下任务
# 在原来的比萨列表中添加一种比萨 在列表friend_pizzas中添加另一种比萨
# 核实有两个不同的列表
# 为此 打印消息“My favorite pizzas are:” 再使用一个for循环来打印第一个列表
# 打印消息“My friend's favorite pizzas are:” 再使用一个for循环来打印第二个列表
# 核实新增的比萨被添加到了正确的列表中
pizza = ['apple', 'banana', 'orange']
friend_pizzas = pizza[:]
pizza.append('cherry')
friend_pizzas.append('lemon')
print('My favorite pizzas are:')
for i in pizza:
    print(i, end=' ')
print('')
print("My friend's favorite pizzas are:")
for i in friend_pizzas:
    print(i, end=' ')
```

```
PS D:\Python\python_exp> & D:/Python/python.exe d:/Python/python_exp/bk_practice3.py
My favorite pizzas are:
apple banana orange cherry
My friend's favorite pizzas are:
apple banana orange lemon
```

- 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。

```python
def solution(number):
    list = []
    for value in range(1,number):
        if(value%3 == 0 or value%5 == 0):
            list.append(value)
    a = sum(list)
    return a
```

```
Time: 523ms Passed: 11Failed: 0
Test Results:
Sample tests
You have passed all of the tests! :)
```

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):
    result = []
    word = word.lower()
    for c in word:
        if word.count(c) > 1:
            result.append(")")
        else:
            result.append("(")
    return "".join(result)
```

```
Time: 509ms Passed: 4Failed: 0
Test Results:
Duplicate Encoder
You have passed all of the tests! :)
```

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
```

```
def valid_braces(string):
    braces = []
    for i in string:
        if i in '({[':
            braces.append(i)
        elif len(braces) != 0:
            if i == ')' and braces[-1] == '(':
                list_braces.pop()
            elif (i == '}') and (braces[-1] == '{'):
                braces.pop()
            elif braces[-1] == '[':
                braces.pop()
    if len(braces) == 0:
        return True
    else:
        return False
```

```
Time: 501ms Passed: 13Failed: 0
Test Results:
Valid Braces
You have passed all of the tests! :)
```

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。

```
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)
```

```
def check_first_letter(triplets, first_letter):
    count = 0
    for triplet in triplets:
        for letter in triplet:
            if first_letter == letter:
                if triplet.index(letter) != 0:
                    count = 1
    if count == 0:
        return True
    else:
        return False


def remove_first_letter(triplets, first_letter):
    for triplet in triplets:
        for i in triplet:
            if i == first_letter:
                triplet.remove(i)
                triplet.insert(2, '0')

    return triplets


def recoverSecret(triplets):
    letters = [letter for triplet in triplets for letter in triplet]
    string = ''
    new_letters = []
    new_letter = []

    for i in letters:
        if i not in new_letters:
            new_letters.append(i)

    while len(string) < len(new_letters):
        for i in range(len(new_letters)):
            if new_letters[i] not in new_letter:
                result = check_first_letter(triplets, new_letters[i])
                if result:
                    new_letter.append(new_letters[i])
                    triplets = remove_first_letter(triplets, new_letters[i])
                    string += new_letters[i]
```

```
    return string


Time: 644ms Passed: 1Failed: 0
Test Results:
Test Passed
You have passed all of the tests! :)
```

第五题： 去掉喷子的元音（Disemvowel Trolls）

难度： 7kyu

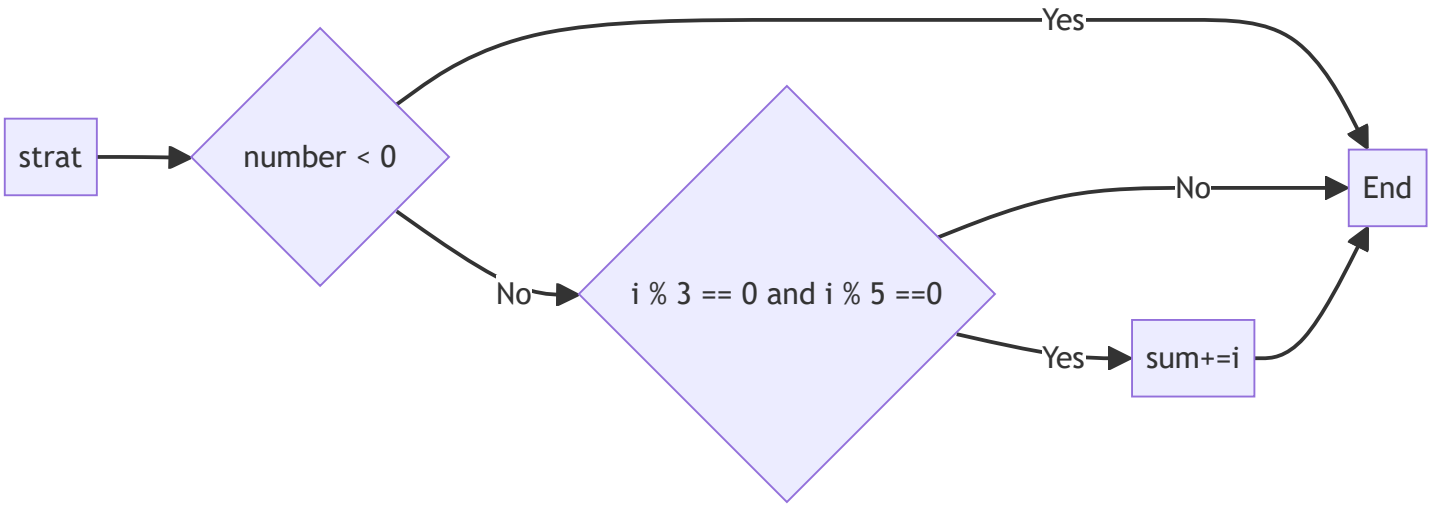
喷子正在攻击你的评论区!
处理这种情况的一个常见方法是删除喷子评论中的所有元音(字母：a,e,i,o,u)，以消除威胁。
你的任务是写一个函数，接收一个字符串并返回一个去除所有元音的新字符串。
例如，字符串 "This website is for losers LOL!" 将变成 "Ths wbst s fr lsrs LL!".

```
def disemvowel(string_):
    list_vowel_lower = ['a', 'e', 'i', 'o', "u"]
    list_vowel_upper = ['A', 'E', 'I', 'O', 'U']
    list_string = list(i for i in string_)
    list_string1 = list_string[:]
    for i in list_string1:
        if i in list_vowel_lower:
            list_string.remove(i)
        if i in list_vowel_upper:
            list_string.remove(i)

    return ''.join(list_string)


Time: 466ms Passed: 3Failed: 0
Test Results:
Fixed Tests
You have passed all of the tests! :)
```

- 第三部分 使用Mermaid绘制程序流程图
-

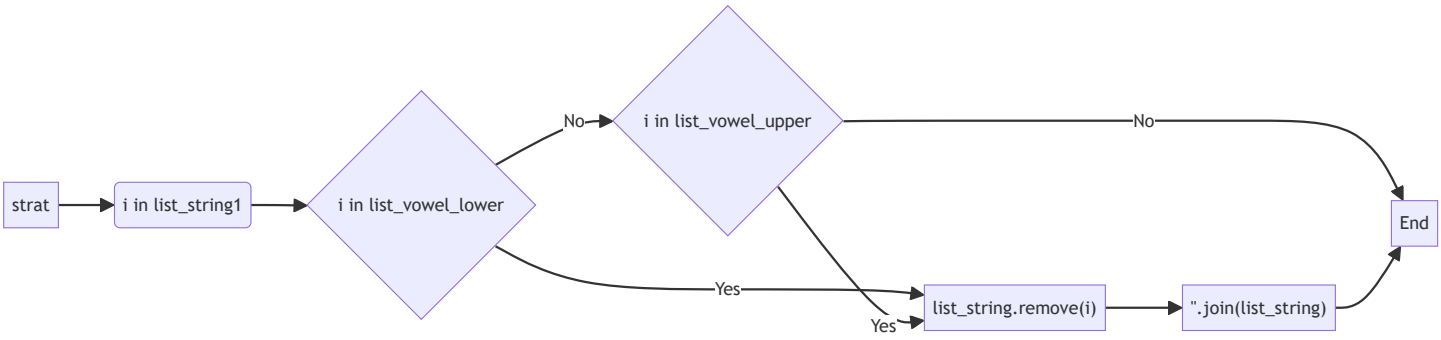
第二部分 第一题

```
flowchart LR
    A[strat] --> B{number < 0}
    B --> |Yes| C[End]
    B --> |No| D{i % 3 == 0 and i % 5 ==0}
    D --> |Yes| E[sum+=i]
    D --> |No| C
    E --> C
```



第二部分 第五题

```
flowchart LR
    A[strat] --> B(i in list_string1)
    B --> C{i in list_vowel_lower}
    C --> |No| D{i in list_vowel_upper}
    C --> |Yes| E["list_string.remove(i)"]
    D --> |Yes| E["list_string.remove(i)"]
    D --> |No| F[End]
    E --> G["''.join(list_string)"]
    G--> F
```



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?
   创建、遍历、切片、增删改查、排序、反转等
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?
   sort() 与 sorted()；
   sorted是临时的
3. 如何将Python列表逆序打印?
   使用reverse()
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?
   访问元素 添加和删除末尾元素效率比较高；
   插入、删除、查找元素效率较差；
   元组可代替列表
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。
   元组是不可变的序列：元组是一种不可变的序列，一旦创建就不能修改。元组可以包含任意类型的对象，并且可以嵌套其他元组；
   元组的创建和访问：可以使用逗号分隔的值来创建元组，也可以使用tuple()函数将其他可迭代对象转换为元组。可以通过索引访问元组中的元素，也可以使用切片来获取子元组；
   元组拆包：可以将元组的元素拆包到多个变量中，这样可以方便地同时获取多个值；
   元组作为记录：元组可以用来表示记录，其中每个元素都有特定的含义。可以通过索引或拆包来访问记录中的字段；
   元组作为不可变列表的替代：元组可以用来表示不可变的列表，因为元组的不可变性可以提供更好的安全性和性能

元组的不可变性：元组的不可变性是指元组的元素不能修改，但是如果元组的元素是可变对象，那么可变对象的状态是可以修改的

实验总结

通过此次试验，学会了Python简单的列表使用，也了解了列表的基本操作，如增删改查、切片等，也学会了Python中的各类if语句搭配使用方法