实验三 Python列表

班级： 21计科1

学号： 202302200000

姓名： 张三

Github地址：https://github.com/yourusername/python_course

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

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：

```python
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

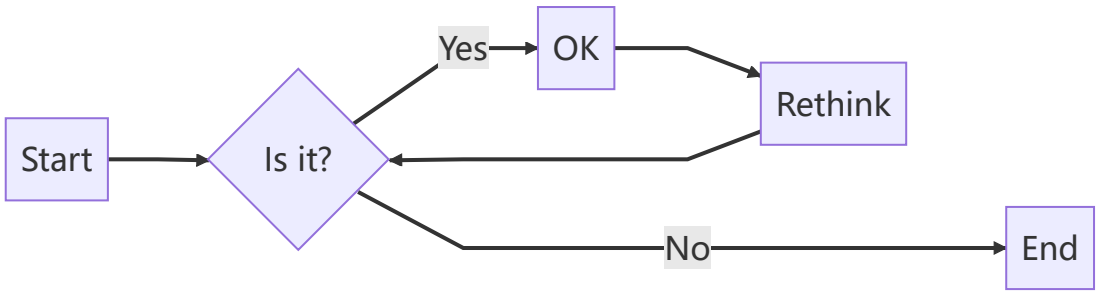
使用Mermaid绘制程序流程图

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：

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

第一部分

Python列表操作

- 3.1 姓名

```python
names = ['Liuxiunneg','JayR','DzK','Gkh','tWk','HYX']
for name in names:
    print(name)
```

- 3.2 问候语

```python
names = ['Liuxiunneg','JayR','DzK','Gkh','tWk','HYX']
for name in names:
    print(f"Hello,{name}")
```

- 3.3 自己的列表

```
Vehicle = ["Yadi", "Honda", "Toyota", "Xds", "Bus", "Metro"]

for vehicle in Vehicle:
    print(f"I would like to own a {vehicle}")
```

- 3.4&3.5_嘉宾名单

```
Ren = ["Liuxiunneg", "JayR", "gag"]

print(f"{Ren[0]} is absent")
Ren[0] = "Lxiunneg"
for men in Ren:
    print(f"{men},Can you join my party?")

```

- 3.6&3.7_添加嘉宾

```
Ren = ["Liuxiunneg", "JayR", "gag"]

Ren.insert(0, "FirstMen")
Ren.insert(2, "TheMiddleMan")
Ren.append("LastMen")

for men in Ren:
    print(f"{men},Can you join my party?")

print('\n')

while len(Ren) != 2:
    Ren.pop()


for men in Ren:
    print(f"{men},Can you join my party?")

```

- 3.8_放眼世界

```
places = ["Hengyang", "Hangzhou", "Beijing", "Shanghai",
          "Guangzhou", "Shenzhen", "Tokyo", "NewYork"]

for place in places:
    print(place)

print()

for place in sorted(places):
    print(place)

print()

for place in reversed(sorted(places)):
    print(place)

```

第二部分：

- 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

```python
def Sum(Base:int,M:int) -> int:
    return (Base + Base * M) * M / 2

def func(number:int,n) -> int:
    temp = int(number/n)
    if number % n == 0:
        temp -= 1
    return temp

def solution(number) -> int:
    pass
    if number <= 3:
        return 0
    elif number <= 5:
        return 3
    return Sum(3, func(number,3)) + Sum(5, func(number,5)) - Sum(15, func(number,15))

print(solution(10))
```

- 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

```python
def duplicate_encode(word):
    word = word.lower()
    dict = {}
    for c in word:
        if c in dict:
            dict[c] += 1
        else:
            dict[c] = 1

    message = ""

    for c in word:
        if dict[c] > 1:
            message = message + ')'
        else:
            message = message + '('

    return message
```

- 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

```python
def valid_braces(string) -> bool:
    stack = []
    if not string:
        return False
    for c in string:
        if c == '(' or c == '{' or c == '[':
            stack.append(c)
        else:
            if not stack:
                return False
            elif c == ')':
                if stack[-1] != '(':
                    return False
                else:
                    stack.pop()
```

```
            elif c == '}':
                if stack[-1] != '{':
                    return False
                else:
                    stack.pop()
            elif c == ']':
                if stack[-1] != '[':
                    return False
                else:
                    stack.pop()
    if not stack:
        return True
    else:
        return False
```

- 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

```
secret = "whatisup"
triplets = [
    ['t', 'u', 'p'],
    ['w', 'h', 'i'],
    ['t', 's', 'u'],
    ['a', 't', 's'],
    ['h', 'a', 'p'],
    ['t', 'i', 's'],
    ['w', 'h', 's']
]
```

```python


def recoverSecret(triplets):
    letters = set()

    before = {}
    after = {}

    start_letters = []

    for triplet in triplets:
        for i in range(0, len(triplet)):
            if triplet[i] not in before:
                before[triplet[i]] = set()
            if triplet[i] not in after:
                after[triplet[i]] = set()

            cur_before = set(triplet[:i])
            cur_after = set(triplet[i+1:])

            before[triplet[i]].update(cur_before)
            after[triplet[i]].update(cur_after)

            letters.add(triplet[i])

    for letter in letters:
        if len(before[letter]) == 0:
            start_letters.append(letter)

    # print(start_letters)

    ans = ""
    visited = set()

    def dfs(letter):
        nonlocal ans

        if letter in visited:
            return

        visited.add(letter)

        for next_after in after[letter]:
            dfs(next_after)

        ans = letter + ans

    for start_letter in start_letters:
        dfs(start_letter)

    return ans


print(recoverSecret(triplets))
```

- 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

```
def disemvowel(string_):
    vowels = ['a','A','o','O','e','E','i','I','u','U']

    for c in vowels:
        string_ = string_.replace(c,'')

    return string_
```

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?

```
# 创建列表
list = []
# 增加元素
list.append(1)
# 删除元素
list.remove()
#修改元素
list[index] = ''
#元素是否在列表
value in list
# 切片
list_new = list_old[:]
```

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?

```
#不修改原序列的排序
sorted(list)
#修改原序列
sort(list)
```

3. 如何将Python列表逆序打印?

```
for element in reversed(list):
    print(element)
```

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?

   高效操作：

   1. 获取单个元素：通过索引直接访问列表中的元素的效率很高，因为它是基于常数时间复杂度 O(1) 完成的。
   2. 添加元素到列表末尾：使用 append() 方法将元素添加到列表的末尾是高效的，因为它的平均时间复杂度也是常数时间 O(1)。

   相对较低效的操作：

   1. 插入或删除元素：在列表的中间位置插入或删除元素会涉及到元素的移动操作，导致时间复杂度为 O(n)，其中 n 是列表的长度。这是因为列表是基于数组实现的，插入或删除元素涉及到元素的移动。
   2. 切片操作：切片操作会创建一个新的列表，并且复制原始列表中的元素，因此时间和空间复杂度都是 O(k)，其中 k 是切片的长度。

   元组和队列考研代替列表

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。

实验总结

学会了列表的操作和使用其来解决算法问题。