实验五 Python数据结构与数据模型

班级： 21计科1

学号： B20210302127

姓名： 刘嘉伟

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

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

实验目的

1. 学习Python数据结构的高级用法
2. 学习Python的数据模型

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

在Codewars网站注册账号，完成下列Kata挑战：

第一题：停止逆转我的单词

难度： 6kyu

编写一个函数，接收一个或多个单词的字符串，并返回相同的字符串，但所有5个或更多的字母单词都是相反的（就像这个Kata的名字一样）。传入的字符串将只由字母和空格组成。只有当出现一个以上的单词时，才会包括空格。 例如：

```
spinWords( "Hey fellow warriors" ) => returns "Hey wollef sroirraw"
spinWords( "This is a test") => returns "This is a test"
spinWords( "This is another test" )=> returns "This is rehtona test"
```

代码提交地址： https://www.codewars.com/kata/5264d2b162488dc400000001

提示：

- 利用str的split方法可以将字符串分为单词列表 例如：

```
words = "hey fellow warrior".split()
# words should be ['hey', 'fellow', 'warrior']
```

- 利用列表推导将长度大于等于5的单词反转(利用切片word[::-1])
- 最后使用str的join方法连结列表中的单词。

第二题： 发现离群的数(Find The Parity Outlier)

难度：6kyu

给你一个包含整数的数组（其长度至少为3，但可能非常大）。该数组要么完全由奇数组成，要么完全由偶数组成，除了一个整数N。请写一个方法，以该数组为参数，返回这个 "离群 "的N。

例如：

```
[2, 4, 0, 100, 4, 11, 2602, 36]
# Should return: 11 (the only odd number)

[160, 3, 1719, 19, 11, 13, -21]
# Should return: 160 (the only even number)
```

代码提交地址： https://www.codewars.com/kata/5526fc09a1bbd946250002dc

---

第三题： 检测Pangram

难度：6kyu

pangram是一个至少包含每个字母一次的句子。例如，"The quick brown fox jumps over the lazy dog "这个句子就是一个pangram，因为它至少使用了一次字母A-Z（大小写不相关）。

给定一个字符串，检测它是否是一个pangram。如果是则返回`True`，如果不是则返回`False`。忽略数字和标点符号。 代码提交地址： https://www.codewars.com/kata/545cedaa9943f7fe7b000048

第四题： 数独解决方案验证

难度：6kyu

数独背景

数独是一种在 9x9 网格上进行的游戏。游戏的目标是用 1 到 9 的数字填充网格的所有单元格，以便每一列、每一行和九个 3x3 子网格（也称为块）中的都包含数字 1 到 9。更多信息请访问：
http://en.wikipedia.org/wiki/Sudoku

编写一个函数接受一个代表数独板的二维数组，如果它是一个有效的解决方案则返回 true，否则返回 false。数独板的单元格也可能包含 0，这将代表空单元格。包含一个或多个零的棋盘被认为是无效的解决方案。棋盘总是 9 x 9 格，每个格只包含 0 到 9 之间的整数。

代码提交地址： https://www.codewars.com/kata/63d1bac72de941033dbf87ae

---

第五题： 疯狂的彩色三角形

难度： 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：

```
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'。

例如：

```
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

---

第一题：停止逆转我的单词

```
def reverse_words(word):
    string = ''.join(reversed(word))
    return string

def spin_words(sentence):
    str = []
    str = sentence.split(' ')
    print(str)
    for i in range(0,len(str)):
        if len(str[i]) >= 5:
            str[i] = reverse_words(str[i])
    string = ' '.join(str)
    return string

#print(spin_words("Hey fellow warriors"))
```

算法思路

> 利用split() 将函数传入的sentence分成一个一个单词保存进单词数组，然后进行遍历。 如果遍历单词数组，单词长度大于等于5则反转单词，最后以字符串的形式返回单词数组中的内容。

第二题： 发现离群的数(Find The Parity Outlier)

```
def find_outlier(integers):
    cnt = 0
    N =0
    print(integers)
    for integer in integers:
        if integer % 2  ==  0:
            cnt+=1
        else :
            cnt-=1
    print(cnt)
    if cnt < 0:
        for integer in integers:
            if integer % 2 == 0 :
                N = integer
    if cnt >0:
        for integer in integers:
            if integer % 2 != 0 or integer == 1:
                N = integer
```

```
    print(N)
    return N
```

算法思路

> 首先计算数组intergers里的奇数和偶数的个数，如果奇数的数量多，则找出数组中不是奇数的数字 然后返回该数字N (离群数)。如果为偶数，则反之。

第三题： 检测Pangram

```
ZIMU_DIC=[

'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u
','v','w','x','y','z'
]
def remove_duplicates_using_set(string): #去除重复的元素
    unique_chars = set(string)
    result = ''.join(unique_chars)
    return result

def is_pangram(s):
    s = remove_duplicates_using_set(s).lower()
    print(s)
    string = ''
    for str in s:
        if  str in ZIMU_DIC:
            string += str

    string  = remove_duplicates_using_set(string)
    print(string)
    if len(string) >= 26 :
        print(len(string))
        return True
    else:
        return False


#print(is_pangram('sklteziuqprcvgbjwhyod1fxnm'))

#print(remove_duplicates_using_set('The quick brown fox jumps over the lazy dog'))
```

算法思路

> 利用set集合去重函数传入的字符串，然后计算该字符串的长度如果大于等于26则返回True,反之则返回False

第四题： 数独解决方案验证

```
def validate_sudoku(board):
    #1~9 的集合
    num = set(range(1,10))
    for i in board:
        if set(i) != num:
            return False
    for i in zip(*board):
        if set(i) != num:
            return False
    # 储存每个3X3的矩阵 验证 9个数字集合是否跟num集合相等
    for i in range(3,10,3):
        for j in range(3,10,3):
            if num != { (board[q][w])  for w in range(j-3, j) for q in range(i-3,
i)}:
                return False
    return True
```

算法思路

> 生成一个1~9的集合num，用于判断所给的数独矩阵行列元素是否与之相等,以及判断数独举证每9宫格里的数字是否与之相等，若满足上述所有条件，则返回True，反之则返回False

第五题： 疯狂的彩色三角形

```
# length of row not in the reduce:
# closest reduce-1 is 9:
# for i in range(12 - 10 + 1 =1)
# for i in range(1)
# e.g.    RGGBBRRGGGRG
# B B G
#  B R
#   G
def Get_nextLineColor(row):
    DIC_ROW ={
    'GG': 'G',  'BB': 'B','RR': 'R',
    'GB': 'R',  'BG': 'R', 'GR': 'B',
    'RG': 'B',  'BR': 'G','RB': 'G'
    }
    reduce=[3**i+1 for i in range(11) if 3**i<=100000][::-1]
    new_row=''
    str=[]

    #如果为3的幂次+1     (3**i+1)
    if len(row) in reduce:
        #print(True)
        str.append(row[0])    #每两个数据为一组：首末相加
        str.append(row[-1])
        str_temp = ''.join(str)
        new_row = DIC_ROW[str_temp]
        return new_row
```

```
    #如果不为3的幂次+1 间隔3幂次相加，来减少计算
    for duce in reduce:
        if len(row) >= duce:
            str=[]
            for i in range(0,len(row)):
                str = []
                str.append(row[i])
                if i+duce <= len(row):
                    str.append(row[i+duce-1])
                    str_temp = ''.join(str)
                    new_row = new_row + DIC_ROW[str_temp]
                else:
                    break
            row = new_row
            break
    return new_row

def triangle(row):
    while len(row) > 1:
        row = Get_nextLineColor(row)
    return row

# 测试数据:
# print(triangle('RGGBBRRGGGRG'),'结果')
# print(triangle('RGGBBRRGGGRGBG'),'结果')
# print(triangle('RGGBBRRGGGRGBRRG'),'结果')
```

算法思路:

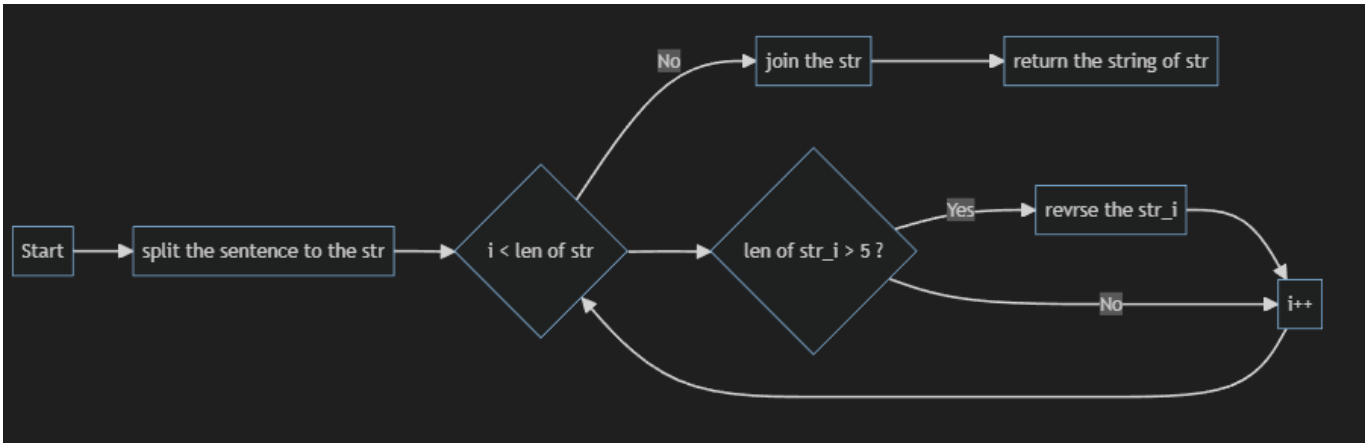
> 如果字符串的长度为3的幂次方加一，则可以直接通过首末的两个字符来得出最终的字符。

> 如果字符串的长度不为3的幂次方加一，可以缩短要计算的颜色代码字符，即可以相隔3的幂次进行计算，然后得到最终的字符。

Mermaid 流程图

第一题：停止逆转我的单词 流程图

```
flowchart LR
    A[Start] --> F[split the sentence to the str]
    F --> E{ i < len of str}
    E -->|No| H[ join the str]
    H --> J[return the string of str]
    E --> B{len of str_i > 5 ?}
    C --> G[i++]
    B -->|Yes| C[revrse the str_i]
    G --> E
    B -->|No| G
```



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. 集合（set）类型有什么特点？它和列表（list）类型有什么区别？ 集合（Set）类型和列表（List）类型是常见的数据结构，它们在以下几个方面有不同的特点和区别：

   集合是无序的，其中的元素没有特定的顺序。列表是有序的，其中的元素按照插入的顺序排列。 集合中的元素是唯一的，不允许重复。列表中的元素可以重复。

   总结来说，集合适用于需要快速查找和判断元素是否存在，并且不关心元素的顺序的场景。列表适用于有序的元素集合，并且需要按照索引访问和操作元素的场景。选择集合还是列表取决于具体的需求和使用场景。

2. 集合（set）类型主要有那些操作?

   集合（Set）类型主要具有以下常见操作： 添加元素：向集合中添加一个元素，确保元素的唯一性，不会添加重复的元素。 删除元素：从集合中移除指定的元素。
   成员检查：检查集合中是否包含指定的元素。
   集合大小：获取集合中元素的数量。
   清空集合：移除集合中的所有元素，使集合变为空集。
   集合运算：进行集合之间的交集、并集、差集等运算。

3. 使用*操作符作用到列表上会产生什么效果？为什么不能使用*操作符作用到嵌套的列表上？使用简单的代码示例说明。

   在Python中，*操作符用于复制列表。当将*操作符应用于列表时，会生成一个新的列表，其中包含原始列表中的元素的多个副本。 下面是一个示例，展示了如何使用*操作符复制列表：

   ```
   original_list = [1, 2, 3]
   copied_list = original_list * 3
   print(copied_list)
   ```

   输出结果为：

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

4. 总结列表,集合，字典的解析（comprehension）的使用方法。使用简单的代码示例说明。 列表解析（List Comprehension）、集合解析（Set Comprehension）和字典解析（Dictionary Comprehension）是在Python中用于简洁地创建和转换这些数据结构的强大工具。它们提供了一种简洁、可读性高的语法来生成列表、集合和字典，避免了繁琐的循环和条件语句。

   以下是它们的使用方法以及简单的代码示例

   列表解析（List Comprehension）：

```python
# 生成一个包含1到5的平方的列表
squares = [x**2 for x in range(1, 6)]
print(squares)  # 输出: [1, 4, 9, 16, 25]

# 使用条件过滤元素
evens = [x for x in range(1, 11) if x % 2 == 0]
print(evens)  # 输出: [2, 4, 6, 8, 10]
```

   集合解析（Set Comprehension）：

```python
# 生成一个包含1到5的平方的集合
squares = {x**2 for x in range(1, 6)}
print(squares)  # 输出: {1, 4, 9, 16, 25}

# 使用条件过滤元素
evens = {x for x in range(1, 11) if x % 2 == 0}
print(evens)  # 输出: {2, 4, 6, 8, 10}
```

   字典解析（Dictionary Comprehension）：

```python
# 生成一个包含1到5的平方的字典，键为数字，值为平方值
squares = {x: x**2 for x in range(1, 6)}
print(squares)  # 输出: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

# 使用条件过滤元素
even_squares = {x: x**2 for x in range(1, 11) if x % 2 == 0}
print(even_squares)  # 输出: {2: 4, 4: 16, 6: 36, 8: 64, 10: 100}
```

   通过列表解析、集合解析和字典解析，可以以一行简洁的代码创建和转换这些数据结构，提高代码的可读性和编写效率。

实验总结

通过实验课程，深入学习了Python数据结构高级用法和数据模型，掌握了列表推导式、生成器、迭代器等高级概念，以及相关数据模型的应用，提升了我对Python编程的理解和技能，收获颇多。