实验五 Python数据结构与数据模型

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 (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。

In [1]:
```
# 将字符串分解为单词列表
words = "hey fellow warrior".split()
# words should be ['hey', 'fellow', 'warrior']

# 将长度大于5的单词反转
spinning_words = [word[::-1] if len(word) >= 5 else word for word in words]

# 使用空格连接单词列表
result = " ".join(spinning_words)
print(result)
```

```
hey wollef roirraw
```

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 (https://www.codewars.com/kata/5526fc09a1bbd946250002dc)

In [ ]:
```python
def find_outlier(int):
    # 保存所有奇数
    odds = [x for x in int if x%2!=0]

    # 保存所有偶数
    evens= [x for x in int if x%2==0]

    # 比较偶数列表和奇数列表的长度，返回长度较小的列表的第一个元素
    return odds[0] if len(odds)<len(evens) else evens[0]
```

In [ ]:
```python
integers = [2, 4, 0, 100, 4, 11, 2602, 36]

# 设置偶数和奇数计数器，初始值都为0
even_count = 0
odd_count = 0

# 遍历整数列表
for value in integers:
    # 如果是偶数，偶数计数器加1
    if value % 2 == 0:
        # 如果当前奇数计数器的值大于1，说明离群的数是该偶数
        if odd_count > 1:
            print(value)  #
            break
        even_count += 1

    # 如果是奇数，奇数计数器加1
    else:
        # 如果当前偶数计数器的值大于1，说明离群的数是该奇数
        if even_count > 1:
            print(value)
            break
        odd_count += 1
```

In [ ]:
```python
def find_outlier(integers):
    # 设置偶数和奇数计数器，初始值都为0
    even_count = 0
    odd_count = 0

    # 设置找到的第一个奇数和偶数，初始值都为None
    first_odd = None
    first_even = None

    # 遍历整数列表
    for value in integers:
        # 如果是偶数
        if value % 2 == 0:
            # 如果这是找到的第一个奇数，记录下来
            if first_even is None:
                first_even = value
            # 如果当前奇数计数器的值大于1，说明离群的数是该偶数
            if odd_count > 1:
                return value

            # 偶数计数器加1
            even_count += 1

            # 如果偶数计数器的值大于1，说明离群的数是该奇数
            # 如果第一个奇数不是None，返回第一个奇数
            if even_count > 1 and first_odd is not None:
                return first_odd

        # 如果是奇数，奇数计数器加1
        else:
            # 如果这是找到的第一个偶数，记录下来
            if first_odd is None:
                first_odd = value
            # 如果当前偶数计数器的值大于1，说明离群的数是该奇数
            if even_count > 1:
                return value
            # 奇数计数器加1
            odd_count += 1

            # 如果奇数计数器的值大于1，说明离群的数是该偶数
            # 如果第一个偶数不是None，返回第一个偶数
            if odd_count > 1 and first_even is not None:
                return first_even
    return None
```

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 (https://www.codewars.com/kata/545cedaa9943f7fe7b000048)

In [ ]:
```python
def is_pangram(s):
    # 将字符串转换为小写
    s = s.lower()

    # 遍历所有小写字母，如果有字母不在字符串中，返回False
    for char in 'abcdefghijklmnopqrstuvwxyz':
        if char not in s:
            return False

    # 遍历结束，说明所有字母都在字符串中，返回True
    return True
```

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 (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
(https://www.codewars.com/kata/63d1bac72de941033dbf87ae)

In [ ]:
```python
def validate_sudoku(board):

    # 利用集合进行比较 {1,2,3,4,5,6,7,8,9}
    elements = set(range(1, 10))

    # row
    for b in board:
        if set(b) != elements:
            return False

    # column
    for b in zip(*board):   # zip(*board) 可以将矩阵转置
        if set(b) != elements:
            return False

    # magic squares
    for i in range(3, 10, 3):
        for j in range(3, 10, 3):
            if elements != {(board[q][w]) for w in range(j-3, j) for q in range(i-3, i)}:
                return False

    return True
```

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'
```

提示：

```
Let R,G,B=0,1,2
for x y
     z

notice that z==(2*(x+y)) (Mod 3)

a         b         c
  2(a+b)    2(b+c)
     4(a+2b+c)

a       b          c          d
 2(a+b)    2(b+c)    2(c+d)
   4(a+2b+c)  4(b+2c+d)
      8(a+3b+3c+d) = a + d
```

根据三进制运算的特点，当字符长度恰好等于3的幂加1时，可以直接运算第一个颜色和最后一个颜色得到结果，例如：计算'RGGB'时可以直接计算'RB'得到'G',计算'RGGBBRRGGG'时可以直接计算'RG'得到'B'。

如果字符长度不是3的幂加1，仍然可以利用上面的方法加快运算，每次用间隔3的幂长度的字符运算，例如：计算'RG{7个任意字符}GB'时可以先计算'R{8个字符}G'得到'B'，再计算'G{8个字符}B'得到'R'，最后计算'BG'得到'R'。

In [ ]:
```
def triangle3(row):
    # 最长的测试用例长度不会超过100000
    # 找到小于100000的所有的3的幂加1，从大到小排序
    # reduce 应该等于[3**9+1, 3**8+1, ... , 3**1+1,  3**0+1]
    reduce=[3**i+1 for i in range(10) if 3**i<=100000][::-1]

    COLOR = {'GG':'G', 'BB':'B', 'RR':'R', 'BR':'G',
            'BG':'R', 'GB':'R', 'GR':'B', 'RG':'B', 'RB':'G'}

    # 从reduce里面最长的长度间隔，取出row里面的元素相加
    for length in reduce:
        while len(row)>=length:
            # row=[row[i] if row[i]==row[i+length-1] else ({"R","G","B"}-{row[i],row[i+length-1]})
            row=[ COLOR[row[i] + row[i+length-1]] for i in range(len(row)-length+1)]
    return row[0]
```

In [1]:
```
print('R' + 'B')
```

```
RB
```