

实验五 Python数据结构与数据模型

班级： 21计科03班

学号： 20210302327

姓名： 廖超逸

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

CodeWars地址： <https://www.codewars.com/users/Licifer>

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 game。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 "RRGBRBBB"，你应该返回 "G"。

限制条件： $1 \leq \text{length}(\text{row}) \leq 10^5$

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>

第二部分

使用Mermaid绘制程序流程图

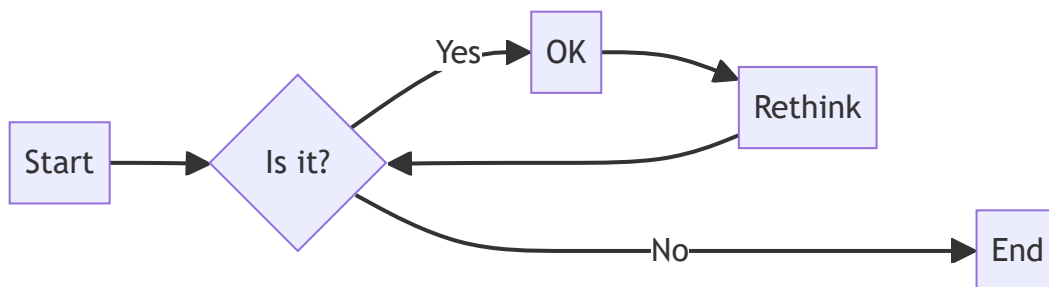
安装VSCode插件：

- 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 Codewars 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"
```

```
def spin_words(sentence):
    sentences = sentence.split(' ')
    new_sentence = []

    for i in sentences:
        if len(i) < 5:
            new_sentence.append(i)
        else:
            new_sentence.append(i[::-1])

    string = ''

    for i in range(len(new_sentence)):
        if i == len(new_sentence) - 1:
            string += new_sentence[i]
        else:
            string += new_sentence[i] + ' '

    return string
```

You have passed all of the tests! 😊

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)
```

```
def find_outlier(integers):
    odd = []
    even = []
    for i in integers:
        if i % 2 == 0:
            even.append(i)
        else:
            odd.append(i)

    if len(odd) == 1:
        return odd[0]
    else:
        return even[0]
```

You have passed all of the tests! 😊

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。

```
def is_pangram(s):
    list_letter = ['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']

    string = ''
    letters = s.split(' ')
    for i in letters:
        string += i.lower()

    for i in string:
        if i in list_letter:
            list_letter.remove(i)
        else:
            pass

    if len(list_letter) > 0:
        return False
    else:
        return True
```

You have passed all of the tests! 😊

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。

```
def validate_sudoku(board):
    for numbers in board:
        for number in numbers:
            if number == 0:
                return False

    numbers = []
    numbers_s = [[], [], [], [], [], [], [], [], []]
    count = 0

    for i in range(1, 9, 3):
        for j in range(0, 9, 3):
            numbers.append(board[i - 1][j:j + 3] + board[i][j:j + 3] + board[i + 1][j:j + 3])

    # 取竖值
    for i in board:
        for j in range(1, 10):
            numbers_s[j - 1].append(i[j - 1])

    for i in numbers:
        set_number = set(i)
        if len(set_number) == 9:
            count += 1
    for i in board:
        set_number = set(i)
        if len(set_number) == 9:
            count += 1
    for i in numbers_s:
        set_number = set(i)
        if len(set_number) == 9:
            count += 1

    if count == 27:
        return True
    else:
        return False
```

You have passed all of the tests! 😊

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 "RRGBRBBB"，你应该返回 "G"。

限制条件： $1 \leq \text{length}(\text{row}) \leq 10 \times 5$

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'
```

```

import timeit

MAPPING = {
    'R': 0, 0: 'R',
    'G': 1, 1: 'G',
    'B': 2, 2: 'B',
}

def signature(n):
    retval = []
    i = 0
    while n > 0:
        n, m = divmod(n, 3)
        if m > 0:
            retval.append((i, m))
        i += 1
    return retval[::-1]

# Lucas's theorem - https://en.wikipedia.org/wiki/Lucas%27s\_theorem
def triangle_rec(row, sig, pos, acc, accm):
    if pos == len(sig):
        # return (MAPPING[row[acc]] * accm) % 3
        return MAPPING[row[acc]] * accm
    retval = 0
    for i in range(sig[pos][1] + 1):
        acc_part = acc + (i * (3 ** sig[pos][0]))
        accm_part = accm
        if sig[pos][1] == 2 and i == 1:
            accm_part *= 2
        retval += triangle_rec(row, sig, pos + 1, acc_part, accm_part)
    return retval % 3

def triangle_rec_g(row, sig, pos, acc, accm):
    if pos == len(sig):
        yield (MAPPING[row[acc]] * accm) % 3
        return
    # return (MAPPING[row[acc]] * accm) % 3
    # retval = 0
    for i in range(sig[pos][1] + 1):
        acc_part = acc + (i * (3 ** sig[pos][0]))
        accm_part = accm
        if sig[pos][1] == 2 and i == 1:
            accm_part *= 2

```

```

    # retval += triangle_rec_g(row, sig, pos + 1, acc_part, accm_part)
    yield from triangle_rec_g(row, sig, pos + 1, acc_part, accm_part)
# return retval % 3

```

```

def triangle(row):
    sig = signature(len(row) - 1)
    color_val = triangle_rec(row, sig, 0, 0, 1)
    if len(row) % 2 == 0:
        color_val = (-1 * color_val) % 3
    return MAPPING[color_val]

def triangle_g(row):
    sig = signature(len(row) - 1)
    color_val = sum(triangle_rec_g(row, sig, 0, 0, 1)) % 3
    if len(row) % 2 == 0:
        color_val = (-1 * color_val) % 3
    return MAPPING[color_val]

```

You have passed all of the tests! 😊

- 第二部分 使用Mermaid绘制程序流程图

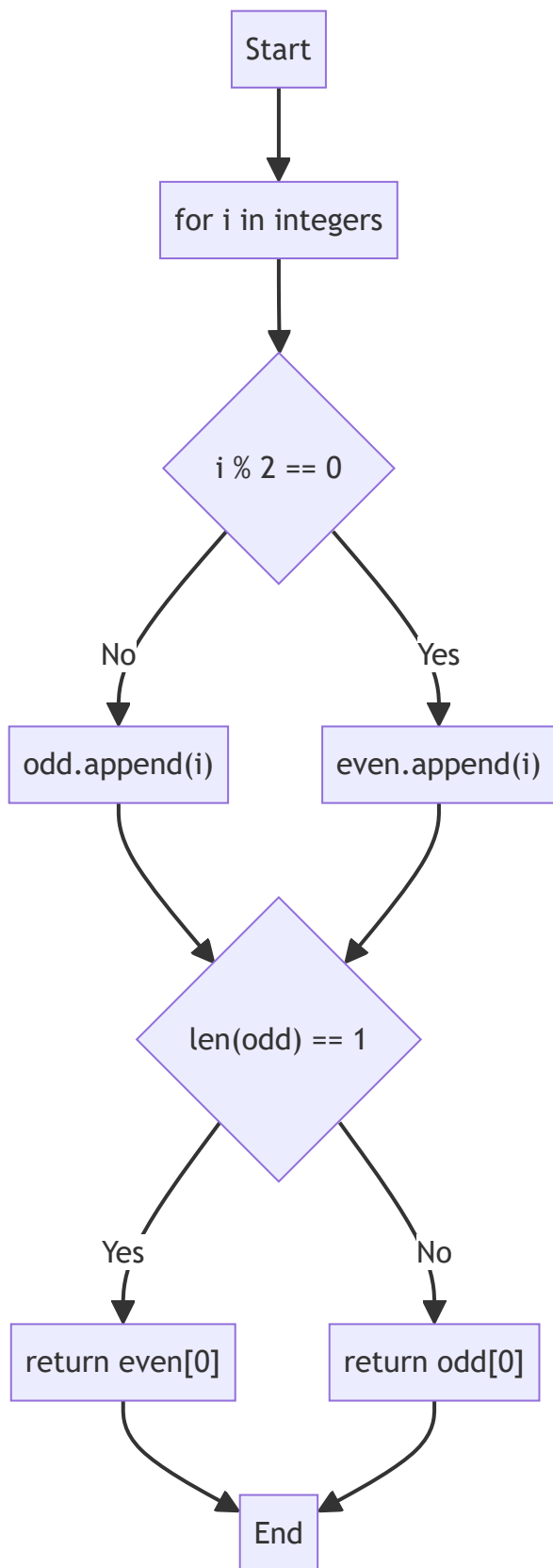
第一部分 第二题

flowchart TD

```

A[Start] --> B[for i in integers]
B --> C{i % 2 == 0}
C --> |No| D["odd.append(i)"]
C --> |Yes| E["even.append(i)"]
D --> F{"len(odd) == 1"}
E --> F
F --> |No| G["return odd[0]"]
F --> |Yes| H["return even[0]"]
H --> I[End]
G --> I

```



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) 类型具有以下特点：

唯一性：集合中的元素是唯一的，不会出现重复的元素。

无序性：集合中的元素没有固定的顺序，每次遍历的结果可能不同。

可变性：集合是可变的，可以添加、删除和修改元素。

与列表 (list) 类型相比，集合类型有以下区别：

唯一性：集合中的元素是唯一的，而列表中的元素可以重复。

无序性：集合中的元素没有固定的顺序，而列表中的元素有顺序。

可变性：集合是可变的，可以添加、删除和修改元素，而列表也是可变的。

2. 集合 (set) 类型主要有那些操作？

集合 (set) 类型主要有以下操作：

添加元素：可以使用 `add()` 方法向集合中添加元素。

删除元素：可以使用 `remove()` 方法从集合中删除指定元素，或使用 `pop()` 方法随机删除一个元素。

清空集合：可以使用 `clear()` 方法清空集合中的所有元素。

判断元素是否在集合中：可以使用 `in` 关键字或者使用 `contains()` 方法判断元素是否在集合中。

集合运算：可以使用交集 (`&`)、并集 (`|`)、差集 (`-`) 等运算操作两个集合。

集合长度：可以使用 `len()` 方法获取集合中元素的个数。

遍历集合：可以使用 for 循环遍历集合中的所有元素。

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。

使用*操作符作用到列表上，会将列表中的元素重复多次，生成一个新的列表。例如，`[1, 2, 3] * 3` 的结果为 `[1, 2, 3, 1, 2, 3, 1, 2, 3]`。

但是，*操作符不能作用到嵌套的列表上，因为这会导致嵌套的列表中的子列表也被重复多次，这可能不是我们想要的结果。例如，`[1, 2], [3, 4] * 2` 的结果为 `[1, 2], [3, 4], [1, 2], [3, 4]`，而不是我们期望的 `[1, 2], [3, 4], [1, 2], [3, 4]`。

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）是一种简洁的方式来创建新的列表，它可以在一行代码中完成对现有列表或其他可迭代对象的元素的筛选和处理。语法为：

```
[expression for item in iterable if condition]
```

其中，expression 是对 item 的处理结果，item 是从 iterable 中遍历的元素，condition 是一个可选的条件表达式。

例如，以下代码使用列表解析创建一个由 1 到 10 的偶数组成的列表：

```
even_numbers = [num for num in range(1, 11) if num % 2 == 0]
```

```
print(even_numbers) # [2, 4, 6, 8, 10]
```

集合解析（set comprehension）类似于列表解析，只是它创建的是集合。语法为：

```
{expression for item in iterable if condition}
```

例如，以下代码使用集合解析创建一个由 1 到 10 的平方组成的集合：

```
squares = {num*num for num in range(1, 11)}
```

```
print(squares) # {1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

字典解析（dictionary comprehension）是一种用于从其他字典或可迭代对象创建新字典的技术。

语法为：

```
{key_expression: value_expression for item in iterable if condition}
```

其中，*key_expression* 和 *value_expression* 分别是对 *item* 的键值对进行处理的表达式。

例如，以下代码使用字典解析创建一个由 1 到 10 的数字和它们的平方组成的字典：

```
square_dict = {num: num*num for num in range(1, 11)}
```

```
print(square_dict) # {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

实验总结

学会了Python的set与list数据类型之间的区别 也了解了set的基本语句与操作 学习Python的*在list上的作用 也熟悉了集合列表字典的使用