华东师范大学计算机科学与技术学院上机实践报告

课程名称：数据结构	**年级**：2022 级	**上机实践成绩**：
指导教师：金健	**姓名**：岳锦鹏	**上机实践时间**：2 学时
上机实践名称：第四章作业	**学号**：10213903403	**上机实践日期**：2023/10/27

一、 实验目的

1. 使用第四章知识解决数组和字符串的问题。

二、 实验内容

1. 给你一个 m 行 n 列的矩阵 matrix，请按照顺时针螺旋顺序，返回矩阵中的所有元素。
2. 给定一个字符串，输出所有长度至少为 2 的回文子串。回文子串即从左往右输出和从右往左输出结果是一样的字符串，比如：abba，cccdeedccc 都是回文字符串。

三、 实验原理

1. 程序设计原理。

四、 实验步骤

1. 问题抽象
2. 编写程序
3. 调试程序
4. 完善总结

五、 调试过程、结果和分析

1. Python 的切片取反时要注意 stop 的参数可能会小于 0，就变成负向索引了。例如正向切片时使用 a[i:j + 1] 可以取出 [i,j + 1) 区间，即 [i,j] 闭区间内的元素，但要是反向切片的话，因为切片的区间包含 start 但不包含 stop，所以应该为 (i - 1, j]，也就是 start 为 j，stop 为 i - 1，即 a[j:i - 1:-1]，但是如果 i = 0 的时候，正向切片是取出 [0, j + 1) 区间内的元素，但反向取就变成了 (-1, j]，而-1 在切片中是负向索引，而每个元素的位置肯定小于等于最后一个元素所在的位置，因此不管 j 怎么取，最终的结果都是空的序列。因此最终的代码采用了先赋值再切片防止出现这样的情况。

六、 总结

1. 在不考虑复杂度的情况下，Python 实现功能真的特别方便；

2. Python 的 and 和 or 有短路规则，即：and 连接的前后两个操作数，如果前一个操作数的逻辑取值为假，则会直接返回前一个操作数（注意是返回原操作数而不是 False 也不是 0，这和 C/C++ 中不一样）；如果前一个操作数的逻辑取值为真，则会返回后一个操作数（注意也是返回原操作数）。同理，or 连接的前后两个操作，如果前一个操作数的逻辑取值为真，则会返回前一个操作数；如果前一个操作数的逻辑取值为假，则会返回后一个操作数；

3. 空列表、空字典、空元组、空集合、空字符串、0 的逻辑取值为假；

4. 列表可以直接使用加号拼接；

5. 单个星号“*”表示按位置参数解包操作符，即把星号后面的操作数按顺序依次取出作为位置参数；

6. zip 函数的功能是依次把传递给它的每个位置参数同时取出一个元素并进行组合，（不好描述，建议看官方文档）可以理解为矩阵转置，以下是官方文档中的解释：
`https://docs.python.org/zh-cn/3/library/functions.html#zip`

7. 如果将二维列表看成是矩阵的话，zip 函数和 * 操作符一起使用是真正的矩阵转置，之后 [::-1] 这样的切片操作表示将外层列表反向，一个二维矩阵转置再反向后就相当于这个二维矩阵逆时针旋转了 90 度；

8. matrix[0] 表示这个矩阵的第一行，matrix[1:] 表示这个矩阵去掉第一行后的部分，所以将矩阵去掉第一行的部分逆时针旋转 90 度，再去掉第一行，再逆时针旋转 90 度，不断重复这样的操作就可以顺时针遍历矩阵中的元素；

9. ___import___ 可以导入模块并返回模块，sys.stdin 是标准输入，sys.stdin.readlines() 是读取标准输入并按行存储，之后使用列表解析式，对每一行 line.split() 表示按空白字符分割，之后返回的就是一个二维列表；

10. “:=”是赋值表达式，会将表达式右端赋值给左端并返回表达式右端的值，相当于 C/C++ 中的“=”；

11. f"..." 是格式化字符串字面值，参考官方文档：
`https://docs.python.org/zh-cn/3/reference/lexical_analysis.html#formatted-string-literals`

12. “_”存储最后一次求值的结果，也可以用来命名无需使用的变量，参考官方文档：
`https://docs.python.org/zh-cn/3/reference/lexical_analysis.html#reserved-classes-of-identifiers`
这里使用是为了防止在 Jupyter 等交互式解释器中运行此段代码时 Jupyter 自动返回最后一次求值的结果。

七、 附件

1. 给你一个 m 行 n 列的矩阵 matrix，请按照顺时针螺旋顺序，返回矩阵中的所有元素。

```python
def reverse(matrix):
    return matrix and list(matrix[0]) + reverse(list(zip(*matrix[1:]))[::-1])
print(reverse([line.split() for line in __import__("sys").stdin.readlines()]))

# 输入:
# 1 2 3
# 4 5 6
# 7 8 9
# * 0 #
# ^Z
# 输出:
# ['1', '2', '3', '6', '9', '#', '0', '*', '7', '4', '5', '8']

# 输入:
# 西北    北      东北
# 西      中      东
# 西南    南      东南
# ^Z
# 输出:
# ['西北', '北', '东北', '东', '东南', '南', '西南', '西', '中']
```

2. 给定一个字符串，输出所有长度至少为 2 的回文子串。回文子串即从左往右输出和从右往左输出结果是一样的字符串，比如：abba，cccdeedccc 都是回文字符串。

```python
# 13:23
# 13:47

# pprint 是第三方库，这里暂时不使用

raw = input() # 下标从 0 开始
_ = [print(part, f"\t位置为 [{i}, {j + 1}]") for j in range(1, len(raw)) for i in range(0, j) if (part := raw[i:j + 1]) == part[::-1]]

# 输入:
# cccdeedccc
# 输出:
# cc      位置为 [0, 2]
```

```
# ccc     位置为 [0, 3]
# cc      位置为 [1, 3]
# ee      位置为 [4, 6]
# deed    位置为 [3, 7]
# cdeedc  位置为 [2, 8]
# ccdeedcc        位置为 [1, 9]
# cc      位置为 [7, 9]
# cccdeedccc      位置为 [0, 10]
# ccc     位置为 [7, 10]
# cc      位置为 [8, 10]

# 输入:
# 上海自来水来自海上
# 输出:
# 来水来  位置为 [3, 6]
# 自来水来自      位置为 [2, 7]
# 海自来水来自海  位置为 [1, 8]
# 上海自来水来自海上      位置为 [0, 9]
```