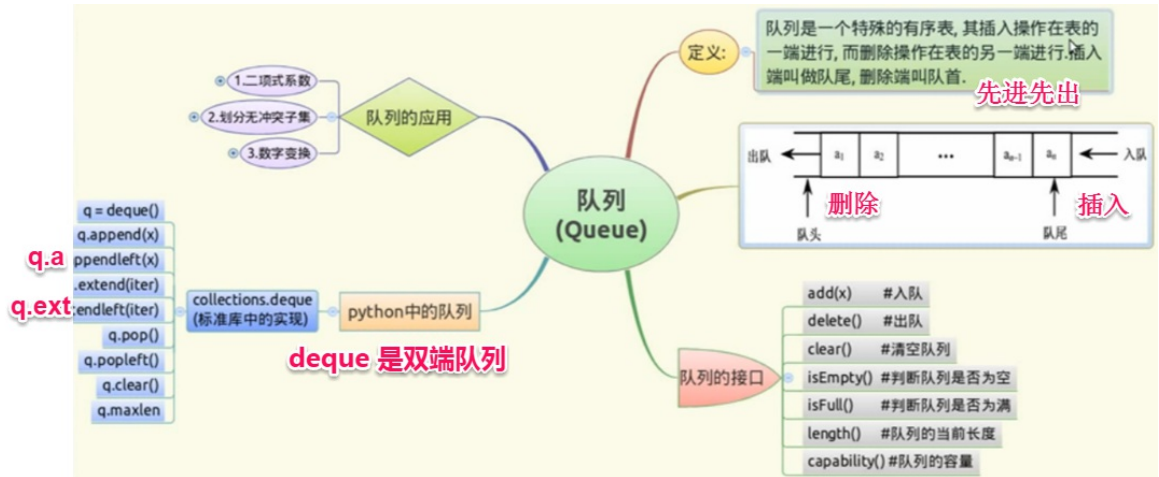


队列-1

? PascalTriangle.py

• 概述



• 实现

- **双端队列?** 队列的每一端都可以插入数据项和移除数据项。这些方法可以叫作 **insertLeft()** 和 **insertRight()**, 以及 **removeLeft()** 和 **removeRight()**。如果严格禁止调用 **insertLeft()** 和 **removeLeft()** 方法 (或禁用右端的操作), 双端队列功能就和栈一样。禁止调用 **insertLeft()** 和 **removeRight()** (或相反的另一对方法), 它的功能就和队列一样了。双端队列与栈或队列相比, 是一种多用途的数据结构, 在容器类库中有时会用双端队列来提供栈和队列两种功能。

◦ 初始化队列

- **q.deque()**: 初始化一个空队列。
- **q.deque([1, 2, 3])**: 初始化一个包含 [1, 2, 3] 三个元素的队列, **必须是通过可迭代对象初始化**。
- **q.deque([1, 2, 3], 5)**: 初始化一个包含 [1, 2, 3] 三个元素的队列, 且限定队列容量是 5。

```
In [18]: q = deque([1, 2, 3], 3)
In [19]: q
Out[19]: deque([1, 2, 3])
In [20]: q.append(4)
In [21]: q
Out[21]: deque([2, 3, 4])
```

↓ "1" 出队列

◦ 入队列

- **q.append(4)**: 右端入队列一个元素
- **q.appendleft(-1)**: 左端入队列一个元素

```
In [27]: q = deque()
In [28]: q = deque([1, 2, 3])
In [29]: q.append(4)
In [30]: q.appendleft(-1)
In [31]: q
Out[31]: deque([-1, 1, 2, 3, 4])
```

- **q.extend([5, 6])**: 右端入队列多个元素
- **q.extendleft([-3, -2])**: 左端入队列多个元素

```

In [31]: q
Out[31]: deque([-1, 1, 2, 3, 4])

In [32]: q.extend([5, 6])

In [33]: q.extendleft([-3, -2])

In [34]: q
Out[34]: deque([-2, -3, -1, 1, 2, 3, 4, 5, 6])

```

注意顺序

- 出队列
 - `q.pop()` : 右端出队列一个元素
 - `q.popleft()` : 左端出队列一个元素
- 清空队列 : `q.clear()`
- 队列容量 : `q.maxlen` (属性)
- 队列为空 : `not q` 为 `True` 队列为空
- 队列满了 : `q.maxlen == len(q)` 为真, 队列满了 (前提是队列有容量限制)
- 队列当前长度 : `len(q)`
- 队列其他函数
 - `q[0]` : 得到队首元素
 - `q[len(q)-1]` : 得到队尾元素
 - `q.count(1)` : 统计队列中某元素个数, "1" 在队列中出现次数
 - `q.remove(1)` : 删除队列中一个元素, 删除队列中元素 "1"
 - `q.reverse()` : 队列中元素逆序
 - `q.rotate(x)` : `x` 个元素从右到左, `-x` 个元素从左到右

```

In [59]: q
Out[59]: deque([1, 2, 2, 3])

In [60]: q.rotate()

In [61]: q
Out[61]: deque([3, 1, 2, 2])

```

• 案例

◦ 案例-1: 杨辉三角

▪ 概述

编写程序, 求第二项式系数表中(杨辉三角)第k层系数.

```

0层  1
1层  1 1
    1 2 1
    1 3 3 1
    1 4 6 4 1
    .....

```

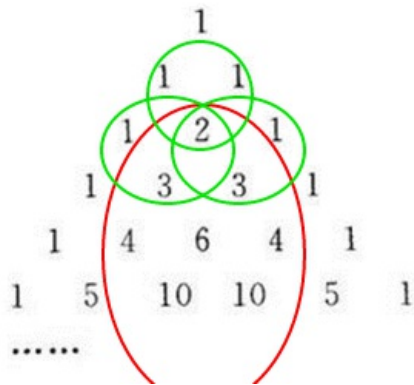
1. 把第k行的系数存储在队列中

2. 依次出队k层的系数(每行最后一个1不出队), 并推算k+1层系数, 添加到队尾. 最后在队尾添加一个1, 便变成了k+1行.

yanghui.py

▪ 思路

- 杨辉三角与二项系数? 杨辉三角 n 层的一行数字, 代表 $(a + b)^n$ 展开式的系数. EG $(a + b)^2 = a^2 + 2ab + b^2$, 与杨辉三角第二行数字一一对应.
- 杨辉三角怎么算出来的? 外层都是 1, 内层 (从第三层 2 开始) 等于上层两个元素之和 (绿色圈)。



- 杨辉三角怎么通过队列实现？① 第 0 行通过 k 次循环得到 k 行元素 ② 第 i 行通过出队 i 个元素，形成 i + 1 行。

第零行	1						
第一行	1	1					
第二行	1	1	2	1			
第三行	1	2	1	3			
	1	2	1	3	3	1	

队列首元素出队列，与它出队列后的队列首元素相加，结果入队（队列最后一个元素保留不出队列，前面元素都如此操作）

队列尾追加+1

- 代码：参见 [PascalTriangle.py](#) 文件，实现了输出杨辉三角第 k 层数字。如下程序实现了输出 0~k 层杨辉三角数字。

```
# coding:utf-8

from collections import deque

def PascalTriangle(k):

    q = deque([1])
    pt = []
    pt.append(list(q)) # 通过定义一个 pt 列表来存储每一层杨辉三角数字，必须通过 list(q) 强制转换队列为列表，否则实现不了

    for i in range(k):
        for _ in range(i):
            q.append(q.popleft() + q[0])
            q.append(1)
        pt.append(list(q)) # 外层循环结束一次，生成一层杨辉三角数字，加入到 pt 中

    return pt

if __name__ == '__main__':

    print PascalTriangle(3)
```

- 案例-2：划分无冲突子集
- And So On