

Circular Counter

```
a = ['1', '2', '3', '4', '5', '6', '7', '8', '9']
def circ_counter(in_t_list, skip):
    skip = skip - 1 # list starts with 0 index
    idx = 0
    while len(in_t_list) > 0:
        # hashing to keep changing the
        index to every 3rd
        idx = (skip + idx) % len(in_t_list)
        print(int_list.pop(idx))
    circ_counter(a, 3)
```

Flatten Array

```
from functools import reduce
def list_flatten(l):
    def flatten(pv, cv):
        if not isinstance(cv, list):
            return pv + [cv]
        else:
            return pv + list_flatten(cv)
    return reduce(flatten, l, [])
print(list_flatten([2, 1, [3, [4, 5], 6], 7, [8]]))
print('output should be [2, 1, 3, 4, 5, 6, 7, 8]')
```

Three Sum Array

```
# TODO: FFT implementation for MlogM algo
def three_sum(nums):
    res = []
    nums.sort()
    for i in range(len(nums) - 2):
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        l, r = i + 1, len(nums) - 1
        while l < r:
            s = nums[i] + nums[l] +
nums[r]
            if s > 0:
                r -= 1
            elif s < 0:
```

Three Sum Array (cont)

```
> l += 1
else:
    # found three sum
    res.append([nums[i], nums[l], nums[r]])
    # remove duplicates
    while l < r and nums[l] == nums[l + 1]:
        l += 1
    while l < r and nums[r] == nums[r - 1]:
        r -= 1
    l += 1
    r -= 1
return res
if __name__ == "__main__":
    x = [-1, 0, 1, 2, -1, -4]
    print("input: ", x)
    print("output should be: ", [[-1, -1, 2], [-1, 0, 1]])
    print("output: ", three_sum(x))
```

Summary Ranges

```
def summary_ranges(nums):
    res = []
    l = len(nums)
    if l == 1:
        return [str(nums[0])]
    i = 0
    while i < l:
        start = nums[i]
        while i + 1 < l and nums[i + 1] -
nums[i] == 1:
            i += 1
        if nums[i] != start:
            res.append(str(start) +
" ->" + str(nums[i]))
        else:
            res.append(str(start))
        i += 1
    return res
a = [0, 1, 2, 4, 5, 7]
```



Summary Ranges (cont)

```
> print("input: ")
print(a)
print("output should be")
print(["0->2", "4->5", "7"])
print("output: ")
print(summary_ranges(a))
```

Rotate Arrays

```
def rotate(arr, k):
    n = len(arr)
    k = k % n
    return arr[n - k:] + arr[:n - k]
a = [1, 2, 3, 4, 5, 6, 7]
print("in: ", a)
print(" expected: ", [5, 6, 7, 1, 2, 3, 4])
print("out: ", rotate(a, 3))
```

Longest Non Repeat

```
from typing import Dict
def longest_n_on_repeat(s: str) -> int:
    start, maxlen = 0, 0
    used_char = {} # type: Dict[str, int]
    for i, char in enumerate(s):
        if char in used_char and start <=
used_c har [char]:
            start = used_c har [char] +
1
        else:
            maxlen = max(maxlen, i -
start + 1)
            used_c har [char] = i
    return maxlen
a = " abc abc def bb"
print(" input: ", a)
print(" result: ", longest_n_on_repeat(a))
print(" output should be 6")
```

Missing Ranges

```
def missing_ranges(arr, low, high):
    hashed = set(arr)
    for n in range(low, high):
        if n not in hashed:
            print(n)
inpt = [10, 12, 11, 15]
low, hi = 10, 15
print(" input: ", inpt)
print(" result: ")
missing_rang es( inpt, low, hi)
```



By **Terracotta45**

Not published yet.

Last updated 30th November, 2024.

Page 2 of 2.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>