Daily Coding Problem Blog

# Daily Coding Problem #102

## Problem

This problem was asked by Lyft.

Given a list of integers and a number K, return which contiguous elements of the list sum to K.

For example, if the list is [1, 2, 3, 4, 5] and K is 9, then it should return [2, 3, 4].

## Solution

The naive approach to solve this question is using brute-force, which has a time complexity of O(N ^ 2). We might look at each possible contiguous sublist and check if its sum matches K. Something like this:

```
def find_continuous_k(list, k):
    for first_idx in xrange(len(list)):
        sum = 0
        for last_idx in xrange(first_idx, len(list)):
            sum += list[last_idx]
            if sum == k:
                return list[first_idx:last_idx + 1]
    return None
```

Note that for this function work as expected with negative numbers we must keep looking even if `sum > k`.

This question doesn't have a complexity requirement, but here's another way to solve it. Take a look at this code:

```
def find_continuous_k(list, k):
    previous = dict()
    sum = 0
    # sublist starting at the zeroth position work.
    previous[0] = -1
    for last_idx, item in enumerate(list):
        sum += item
        previous[sum] = last_idx
        first_idx = previous.get(sum - k)
        if first_idx is not None:
            return list[first_idx + 1:last_idx + 1]
    return None
```

As you can see, we store all previous sums and their positions to look for a specific number required to satisfy K. This way, we have O(n) time complexity and O(n * 2) space.