Daily Coding Problem #74

Problem

This problem was asked by Apple.

Suppose you have a multiplication table that is N by N. That is, a 2D array where the value at the `i`-th row and `j`-th column is `(i + 1) * (j + 1)` (if 0-indexed) or `i * j` (if 1-indexed).

Given integers N and X, write a function that returns the number of times X appears as a value in an N by N multiplication table.

For example, given N = 6 and X = 12, you should return 4, since the multiplication table looks like this:

1	2	3	4	5	6
2	4	6	8	10	12
3	6	9	12	15	18
4	8	12	16	20	24
5	10	15	20	25	30
6	12	18	24	30	36

And there are 4 12's in the table.

Solution

We can do this naively in O(N^2) time by actually trying out all the possible combinations, and incrementing a counter each time we see one:

```
def multi_tables(n, x):
```

```python
    count = 0
    for i in range(1, n + 1):
        for j in range(1, n + 1):
            if i * j == x:
                count += 1
    return count
```

We can do this faster, though. Notice, in our example, that if we look at the rows or columns of the cells that matched with 12, that they are all factors of 12. There can also only be one matching cell per row. So, we can determine whether a particular row will match X if:

1. It is a factor of X
2. Its corresponding factor is less than N (so it's still in the matrix).

```python
def multi_tables(n, x):
    count = 0
    for i in range(1, n + 1):
        if x % i == 0 and x / i <= n:
            count += 1
    return count
```

This only takes O(N) time.