Daily Coding Problem #90

Problem

This question was asked by Google.

Given an integer `n` and a list of integers `l`, write a function that randomly generates a number from `0` to `n-1` that isn't in `l` (uniform).

Solution

One way we can approach this problem is by using rejection sampling. First, we generate a (uniformly) random integer between `0` and `n-1` (inclusive). Then, we check whether the random integer is found within the list `l`. If it is found, then generate repeat the process. If it is not found, then return that number. To make sure we can check for the presence of a number in `O(1)` time, we can put all integers in `l` in a set.

Since this solution involves repeatedly generating a new random number, it could have up to infinite worst-case runtime. The initial call also incurs `O(N)` to convert list into a set. The probability of selecting a random number depends on the the ratio of numbers in `l` that are within the bounds `0` to `n-1`, versus the number `n`.

Another way we can approach this problem is by generating a random number strictly from the numbers available. We can construct the list of numbers of available by subtracting the set of integers in `l` from the set of integers in the range `0` to `n-1`. Then, we can simply generate a random number within `0` and the length of this new list, and return the integer at that index.

This solution takes `O(N)` time to pre-process the list, and `O(1)` time to generate a random integer.

```
from random import randrange
```

```
def process_list(n, l):
    all_nums_set = set()
    for i in range(n):
        all_nums_set.add(i)

    l_set = set(l)
    nums_set = all_nums_set - l_set
    return list(nums_set)

def random_number_excluing_list(n, l):
    nums_list = process_list(n, l)
    idx = randrange(0, len(nums_list))
    return nums_list[idx]

print(random_number_excluing_list(4, [1, 2, 5]))
```