Daily Coding Problem #175

Problem

This problem was asked by Google.

You are given a starting state `start`, a list of transition probabilities for a Markov chain, and a number of steps `num_steps`. Run the Markov chain starting from `start` for `num_steps` and compute the number of times we visited each state.

For example, given the starting state a, number of steps 5000, and the following transition probabilities:

```
[
  ('a', 'a', 0.9),
  ('a', 'b', 0.075),
  ('a', 'c', 0.025),
  ('b', 'a', 0.15),
  ('b', 'b', 0.8),
  ('b', 'c', 0.05),
  ('c', 'a', 0.25),
  ('c', 'b', 0.25),
  ('c', 'c', 0.5)
```

```
]
```

One instance of running this Markov chain might produce `{ 'a': 3012, 'b': 1656, 'c': 332 }`.

Solution

We need to run the Markov chain and keep counts of each state we visit.

It might be useful to define a `next_state` function that takes in the current state and perhaps the possible transitions and their probabilities. Then we can run our Markov chain, starting with `start`, by running `next_state` the desired number of times while keeping track of a current state. All we have to do then it to keep track of each state's counts.

Finally, even though we get the probabilities as a list of tuples, it would be best if we transformed the list into a dict so that we can lookup the possible transitions and their probabilities faster:

```
from collections import defaultdict
from random import random

def histogram_counts(start, trans_probs, num_steps):
    probs_dict = transform_probs(trans_probs)
    count_histogram = defaultdict(int)
    current_state = start

    for i in range(num_steps):
        count_histogram[current_state] += 1
        next_state_val = next_state(current_state, probs_dict)
        current_state = next_state_val

    return count_histogram


def next_state(current_state, probs_dict):
    r = random()
```

```
    for possible_state, probability in probs_dict[current_state].items():
        r -= probability
        if r <= 0:
            return possible_state


def transform_probs(trans_probs):
    d = defaultdict(dict)
    for start, end, prob in trans_probs:
        d[start][end] = prob
    return d
```