```
"""Tools for implementing Calvo-Armengo/Jackson, 2003."""

from random import random, shuffle
import math

def give_to_neighbor(graph, neighbors):

  shuffle(neighbors)

  for nei_id in neighbors:
      for nei in graph:
          if nei["id"] == nei_id and not nei["hist"][-1]:
              nei["empl"] = True
              return


def iterate(graph, a = 0.100, b = 0.015, N = 100000):

    for n in graph:
        n["hist"] = []
        n["empl"] = True

    for i in range(N):

        for n in graph:
            n["hist"].append(n["empl"])

        for n1 in graph:
            if random() < a:
                if not n1["hist"][-1]:
                    n1["empl"] = True

                else: give_to_neighbor(graph, n1["nei"])

        for n in graph:
            if random() < b:
                n["empl"] = False



def unemployment(l): return 1 - sum(l)/len(l)


def correlation(l1, l2):

    # Here, var is E[X^2] - E[X]^2.
    # Since 1^2 = 1, E[X^2] = E[X].

    avg1 = sum(l1)/len(l1)
    var1 = avg1 - avg1**2

    avg2 = sum(l2)/len(l2)
    var2 = avg2 - avg2**2

    exp = sum([(x1 - avg1)*(x2 - avg2) for x1, x2 in zip(l1, l2)]) / len(l1)

    return exp / math.sqrt(var1 * var2)
```

```
def avg_path_length_single(node, graph):

    max_d = 0
    dist  = {node : 0}

    while max_d in dist.values():
        for n in graph:
            if n["id"] in dist and dist[n["id"]] == max_d:
                for nei in n["nei"]:
                    if nei not in dist:
                        dist[nei] = max_d+1
        max_d += 1

    if len(graph) > len(dist): return float('inf')
    else: return sum(dist.values())/(len(dist)-1)


def avg_path_length(graph):

    return sum([avg_path_length_single(n["id"], graph) for n in graph])/len(graph)
```