In [1]:
```
import random
import numpy as np
import simpy
import pandas as pd




class Checker(object):
    """

    """
    def __init__(self, env, num_checkers, checktime):
        self.env = env
        self.checkers = simpy.Resource(env, num_checkers)



    def check(self, person):
        """"""
        yield self.env.timeout(CHECKTIME.pop(0))


class Scanner(object):
    """

    """
    def __init__(self, env, num_scanners, scantime):
        self.env = env
        self.machine = simpy.Resource(env, num_scanners)

    def scan(self, person):

        yield self.env.timeout(SCANTIME.pop(0))



def person(env, name, checker, scanner):
    """

    """
    arrival_name = name
    arrival_time = env.now

    results["arrival"]['arrival_time'].append(arrival_time)
    results["arrival"]['index'].append(arrival_name)

    with checker.checkers.request() as request:
        yield request

        check_name = name
        yield env.process(checker.check(check_name))
        check_time = env.now

        results["check"]['check_time'].append(check_time)
        results["check"]['index'].append(check_name)
```

```
        scan_name = name
        yield env.process(scanner.scan(scan_name))
        scan_time = env.now

        results["scan"]['scan_time'].append(scan_time)
        results["scan"]['index'].append(scan_name)


def setup(env, num_checkers, checktime, num_scanners, scantime):
    """"""
    # Create queue
    checker = Checker(env, num_checkers, checktime)
    scanner = Scanner(env, num_scanners, scantime)
    # Create 4 initial persons
    for person_id in range(4):
        env.process(person(env, person_id, checker, scanner))

    # Create more persons while the simulation is running
    person_id += 1
    arrivals = np.random.poisson(lam=ARRIVALRATE, size=10000)
    while True:

        for i in arrivals:
            yield env.timeout(1)
            for c in range(i+1):
                env.process(person(env, person_id, checker, scanner))
                person_id += 1
```

In [5]:
```
combos = []
for c in range(30,150):
    for s in range(1,11):
        combos.append((c,s))
```

In [6]:
```
#Super ugly for loop, sorry, just wanted to finish after I figured it ou
t...

for combo in combos:

    results = {
        'arrival':
            {'arrival_time':[], 'index':[]},

        'check':
            {'check_time':[], 'index':[]},
        'scan':
            {'scan_time':[], 'index':[]},
        }


    RANDOM_SEED = 42
    RANDOM_SEED = 42
    NUM_CHECKERS,NUM_SCANNERS = combo
    ARRIVALRATE = 50
    CHECKERRATE = .75
    MINIMUMSCAN = 0.5
    MAXIMUMSCAN = 1



    CHECKTIME = list(np.random.exponential(scale=CHECKERRATE, size=10000
))
    SCANTIME = list(np.random.uniform(low=MINIMUMSCAN, high=MAXIMUMSCAN,
 size=10000))
    SIM_TIME = 100     #
    random.seed(RANDOM_SEED)  # This helps reproducing the results

    # Create an environment and start the setup process
    env = simpy.Environment()
    s = setup(env, NUM_CHECKERS, CHECKTIME.pop(0), NUM_SCANNERS, SCANTIM
E.pop(0))
    env.process(s)

    # Execute!
    env.run(until=SIM_TIME)


    df_list = []
    for k,v in results.items():
        val = [x for x in v.keys() if x != 'index'][0]
        data = v[val]
        index = v['index']
        d= pd.DataFrame(data = data, index = index)
        d.columns = [val]
        df_list.append(d)
    df = pd.concat(df_list, axis = 1)
    df['wait_time'] = df['scan_time']-df['arrival_time']
    wt = df['wait_time'].mean()
    if wt<15:
        break
```

In [7]:
```
#the number of checkers, number of scanners and average wait time
print('Checkers:{} Scanners:{} Wait time:{}'.format(NUM_CHECKERS,NUM_SCA
NNERS, wt))
```

```
Checkers:54 Scanners:2 Wait time:14.696062857365384
```