In [97]:
```
import random
import math
import pandas as pd
import numpy as np
from collections import deque
```

In [111]:
```

arrival_gap = 6

rejection_rate = 0.1

mean_service_time ={
    'A':4,
    'B':20
}

std_dev_service_time ={
    'A':2,
    'B':10

}

time_ = 0

no_of_accepted_As = 0

next_event_times = {
    'arrival':0,
    'A':0,
    'B':0
}

servers_busy = {
    'A':False,
    'B':False
}

Queues ={
    'A':0,
    'B':0
}

arrival_timestamps ={
    'A':deque(),
    'B':deque()
}

tree = {}

timestamp_arrival = {}
```

In [112]:
```
class Arrivals:
    def __init__(self):
        self.arrival_times = np.random.poisson(6, 100)
        self.counter = 0
    def get(self):
        self.counter += 1
        return self.arrival_times[self.counter % 100]

arrivals = Arrivals()

get_inter_arrival_time = lambda _ : arrivals.get()
```

In [113]:
```
class Services:
    def __init__(self, part_type):
        self.service_times = np.random.normal(mean_service_time[part_type], std_d
        self.counter = 0
    def get(self):
        self.counter +=1
        return self.service_times[self.counter % 100]

service_times = {
    'A' : Services('A'),
    'B' : Services('B')
}

get_service_time = lambda part_type : service_times[part_type].get()
```

In [116]:
```
def start_possible_server():
    """
    Check if any server can be started.
        Check the server state of both the servers.
        If any server is not busy and has at least one customer in queue, start t
            1. set the server busy.
            2. reduce its queue by 1.
            3. calculate the service time at the moment and update the time for n
    """
    global time_
    for server in ['A', 'B']:
        if (Queues[server] > 0) and (not servers_busy[server]):
            servers_busy[server] = True
            Queues[server] -= 1
            time_ = round(time_, 2)

            tree[time_] = {}
            tree[time_]['server'] = server
            start_time = time_
            idle_time = time_ - next_event_times[server]
            next_event_times[server] = time_ + get_service_time(server)
            end_time = next_event_times[server]
            waiting_time = time_ - arrival_timestamps[server].pop()
            tree[time_]['data'] = (round(start_time,2), round(end_time,2), round(
```

In [114]:
```
whose_arrival = lambda _ : 'A' if random.random() < 0.9  else 'B'
accept_part = lambda _ : True if random.random() > rejection_rate else False
```

In [115]:
```
def check_for_events():
    """
    Check for events and act accordingly. Events occur when :
        1. a new arrival occurs. It's known when the next arrival time is less or
            => update the Queue of the respective arrival and update the next arr

        2. service process of any server ends. It's known when the ending time(s)
            =>
                a. set the server's busy variable to False.
                b. check for the acceptance of the produced part.
                c. if the produced part is type A and it's accepted,
                        increase no_of_accepted_As

        3. both 1 and 2 occur.
            => 1 and 2.
    """
    global no_of_accepted_As, time_

    next_event_times_ = np.array(list(next_event_times.values()))
    next_event_times_ = np.where([True, *list(servers_busy.values())], next_event

    time_ = next_event_times_.min()


    next_event_correpondence = np.where(next_event_times_ == time_)[0].tolist()


    for i in next_event_correpondence:

        if i == 0:
            # a new arrival occurs. Find which part has arrived.
            new_arrival_part = whose_arrival(_)
            Queues[new_arrival_part] += 1
            current_time = next_event_times['arrival']

            arrival_timestamps[new_arrival_part].append(current_time)
            next_event_times['arrival'] += get_inter_arrival_time(_)
            timestamp_arrival [round(time_, 2) ] = new_arrival_part
        else:
            # the corresponding has completed its assigned task
            task_completer_server =  ['A', 'B'][i - 1]
            servers_busy[task_completer_server] = False
            # if the server is 'A' or the produced part is type 'A', increase the
            no_of_accepted_As += (task_completer_server == 'A') and accept_part(_
```

In [117]:
```
"""
Run the simulation.
    1. Check for the events (arrival or completion)
    2. Start new events.
"""
while( no_of_accepted_As < 50):
    check_for_events()
    start_possible_server()
```

In [139]:
```
df_ = {}
for time_ in tree:
    server = tree[time_]['server']
    df_[time_] = [np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan]
    df_[time_] =[-1,-1,-1,-1,-1,-1,-1,-1]
    if server == 'A':
        df_[time_][:4] = tree[time_]['data']
    if server == 'B':
        df_[time_][4:] = tree[time_]['data']
```

In [140]:
```
df = pd.DataFrame(df_).T
df.to_csv('simulation.csv', header=False)
```

In [182]:
```
header = pd.MultiIndex.from_product([['Type A','Type B'], ['start-time','end-time
df = pd.read_csv('simulation.csv',names=header)
df_with_Arrivals = df.copy()
df_with_Arrivals ['Arrivals'] = pd.DataFrame(timestamp_arrival, index=['Arrivals'
```

Simulation

In [183]:
```
df
```

Out[183]:

	Type A				Type B			
	start-time	end-time	wating Time	idle Time	start-time	end-time	wating Time	idle Time
0.00	0.00	4.74	0.00	0.00	-1.00	-1.00	-1.00	-1.00
1.00	-1.00	-1.00	-1.00	-1.00	1.00	7.64	0.00	1.00
6.00	6.00	13.86	0.00	1.26	-1.00	-1.00	-1.00	-1.00
13.86	13.86	18.46	0.86	-0.00	-1.00	-1.00	-1.00	-1.00
18.46	18.46	22.37	0.46	-0.00	-1.00	-1.00	-1.00	-1.00
24.00	24.00	28.28	0.00	1.63	-1.00	-1.00	-1.00	-1.00
31.00	-1.00	-1.00	-1.00	-1.00	31.00	56.00	0.00	23.36
42.00	42.00	45.44	0.00	13.72	-1.00	-1.00	-1.00	-1.00
48.00	48.00	49.98	0.00	2.56	-1.00	-1.00	-1.00	-1.00
50.00	50.00	56.15	0.00	0.02	-1.00	-1.00	-1.00	-1.00
56.00	-1.00	-1.00	-1.00	-1.00	56.00	79.60	9.00	-0.00
61.00	61.00	65.00	0.00	4.85	-1.00	-1.00	-1.00	-1.00
65.00	65.00	69.33	2.00	0.00	-1.00	-1.00	-1.00	-1.00
79.60	-1.00	-1.00	-1.00	-1.00	79.60	101.73	3.60	0.00
85.00	85.00	89.38	0.00	15.67	-1.00	-1.00	-1.00	-1.00
93.00	93.00	96.13	0.00	3.62	-1.00	-1.00	-1.00	-1.00
100.00	100.00	105.33	0.00	3.87	-1.00	-1.00	-1.00	-1.00
100.64	-1.00	-1.00	-1.00	-1.00	100.64	126.54	42.64	0.00
101.73	-1.00	-1.00	-1.00	-1.00	101.73	100.64	30.73	0.00
105.33	105.33	110.73	2.33	-0.00	-1.00	-1.00	-1.00	-1.00
111.00	111.00	113.03	0.00	0.27	-1.00	-1.00	-1.00	-1.00
117.00	117.00	122.59	0.00	3.97	-1.00	-1.00	-1.00	-1.00
123.00	123.00	126.90	0.00	0.41	-1.00	-1.00	-1.00	-1.00
131.00	131.00	134.63	0.00	4.10	-1.00	-1.00	-1.00	-1.00
143.00	-1.00	-1.00	-1.00	-1.00	143.00	160.06	0.00	16.46
150.00	150.00	155.24	0.00	15.37	-1.00	-1.00	-1.00	-1.00
155.24	155.24	158.85	2.24	-0.00	-1.00	-1.00	-1.00	-1.00
161.00	161.00	162.47	0.00	2.15	-1.00	-1.00	-1.00	-1.00
168.00	168.00	171.90	0.00	5.53	-1.00	-1.00	-1.00	-1.00
175.00	175.00	177.52	0.00	3.10	-1.00	-1.00	-1.00	-1.00
...	...	...	...	...	...	...	...	...
218.18	218.18	224.08	3.18	0.00	-1.00	-1.00	-1.00	-1.00
224.08	224.08	228.46	4.08	-0.00	-1.00	-1.00	-1.00	-1.00

	Type A				Type B			
	start-time	end-time	wating Time	idle Time	start-time	end-time	wating Time	idle Time
230.00	230.00	230.54	0.00	1.54	-1.00	-1.00	-1.00	-1.00
237.00	237.00	242.00	0.00	6.46	-1.00	-1.00	-1.00	-1.00
242.00	242.00	245.62	0.00	-0.00	-1.00	-1.00	-1.00	-1.00
249.00	249.00	252.94	0.00	3.38	-1.00	-1.00	-1.00	-1.00
255.00	255.00	258.85	0.00	2.06	-1.00	-1.00	-1.00	-1.00
263.00	263.00	265.72	0.00	4.15	-1.00	-1.00	-1.00	-1.00
272.00	272.00	280.29	0.00	6.28	-1.00	-1.00	-1.00	-1.00
275.00	-1.00	-1.00	-1.00	-1.00	275.00	312.87	0.00	44.64
280.29	280.29	283.34	7.29	0.00	-1.00	-1.00	-1.00	-1.00
283.34	283.34	288.09	0.34	0.00	-1.00	-1.00	-1.00	-1.00
288.09	288.09	291.31	1.09	-0.00	-1.00	-1.00	-1.00	-1.00
297.00	297.00	300.04	0.00	5.69	-1.00	-1.00	-1.00	-1.00
301.00	301.00	306.19	0.00	0.96	-1.00	-1.00	-1.00	-1.00
306.19	306.19	311.39	1.19	0.00	-1.00	-1.00	-1.00	-1.00
312.87	-1.00	-1.00	-1.00	-1.00	312.87	336.33	2.87	-0.00
314.00	314.00	320.43	0.00	2.61	-1.00	-1.00	-1.00	-1.00
320.43	320.43	324.77	3.43	-0.00	-1.00	-1.00	-1.00	-1.00
324.77	324.77	330.54	3.77	0.00	-1.00	-1.00	-1.00	-1.00
330.54	330.54	333.23	0.54	0.00	-1.00	-1.00	-1.00	-1.00
336.33	-1.00	-1.00	-1.00	-1.00	336.33	355.29	43.33	-0.00
340.00	340.00	342.28	0.00	6.77	-1.00	-1.00	-1.00	-1.00
353.00	353.00	357.92	0.00	10.72	-1.00	-1.00	-1.00	-1.00
355.29	-1.00	-1.00	-1.00	-1.00	355.29	357.78	10.29	-0.00
357.92	357.92	361.35	1.92	0.00	-1.00	-1.00	-1.00	-1.00
367.00	367.00	371.26	0.00	5.65	-1.00	-1.00	-1.00	-1.00
374.00	374.00	376.23	0.00	2.74	-1.00	-1.00	-1.00	-1.00
378.00	378.00	383.39	0.00	1.77	-1.00	-1.00	-1.00	-1.00
383.39	383.39	385.32	0.39	-0.00	-1.00	-1.00	-1.00	-1.00

65 rows × 8 columns

Simulationwith Arrivals

In [185]:
```
df_with_Arrivals
```

Out[185]:

	Type A				Type B				Arrivals
	start-time	end-time	wating Time	idle Time	start-time	end-time	wating Time	idle Time	
0.00	0.00	4.74	0.00	0.00	-1.00	-1.00	-1.00	-1.00	A
1.00	-1.00	-1.00	-1.00	-1.00	1.00	7.64	0.00	1.00	B
6.00	6.00	13.86	0.00	1.26	-1.00	-1.00	-1.00	-1.00	A
13.86	13.86	18.46	0.86	-0.00	-1.00	-1.00	-1.00	-1.00	NaN
18.46	18.46	22.37	0.46	-0.00	-1.00	-1.00	-1.00	-1.00	NaN
24.00	24.00	28.28	0.00	1.63	-1.00	-1.00	-1.00	-1.00	A
31.00	-1.00	-1.00	-1.00	-1.00	31.00	56.00	0.00	23.36	B
42.00	42.00	45.44	0.00	13.72	-1.00	-1.00	-1.00	-1.00	A
48.00	48.00	49.98	0.00	2.56	-1.00	-1.00	-1.00	-1.00	A
50.00	50.00	56.15	0.00	0.02	-1.00	-1.00	-1.00	-1.00	A
56.00	-1.00	-1.00	-1.00	-1.00	56.00	79.60	9.00	-0.00	NaN
61.00	61.00	65.00	0.00	4.85	-1.00	-1.00	-1.00	-1.00	A
65.00	65.00	69.33	2.00	0.00	-1.00	-1.00	-1.00	-1.00	NaN
79.60	-1.00	-1.00	-1.00	-1.00	79.60	101.73	3.60	0.00	NaN
85.00	85.00	89.38	0.00	15.67	-1.00	-1.00	-1.00	-1.00	A
93.00	93.00	96.13	0.00	3.62	-1.00	-1.00	-1.00	-1.00	A
100.00	100.00	105.33	0.00	3.87	-1.00	-1.00	-1.00	-1.00	A
100.64	-1.00	-1.00	-1.00	-1.00	100.64	126.54	42.64	0.00	NaN
101.73	-1.00	-1.00	-1.00	-1.00	101.73	100.64	30.73	0.00	NaN
105.33	105.33	110.73	2.33	-0.00	-1.00	-1.00	-1.00	-1.00	NaN
111.00	111.00	113.03	0.00	0.27	-1.00	-1.00	-1.00	-1.00	A
117.00	117.00	122.59	0.00	3.97	-1.00	-1.00	-1.00	-1.00	A
123.00	123.00	126.90	0.00	0.41	-1.00	-1.00	-1.00	-1.00	A
131.00	131.00	134.63	0.00	4.10	-1.00	-1.00	-1.00	-1.00	A
143.00	-1.00	-1.00	-1.00	-1.00	143.00	160.06	0.00	16.46	B
150.00	150.00	155.24	0.00	15.37	-1.00	-1.00	-1.00	-1.00	A
155.24	155.24	158.85	2.24	-0.00	-1.00	-1.00	-1.00	-1.00	NaN
161.00	161.00	162.47	0.00	2.15	-1.00	-1.00	-1.00	-1.00	A
168.00	168.00	171.90	0.00	5.53	-1.00	-1.00	-1.00	-1.00	A
175.00	175.00	177.52	0.00	3.10	-1.00	-1.00	-1.00	-1.00	A
...	...	...	...	...	...	...	...	...	...
218.18	218.18	224.08	3.18	0.00	-1.00	-1.00	-1.00	-1.00	NaN

	Type A				Type B				Arrivals
	start-time	end-time	wating Time	idle Time	start-time	end-time	wating Time	idle Time	
224.08	224.08	228.46	4.08	-0.00	-1.00	-1.00	-1.00	-1.00	NaN
230.00	230.00	230.54	0.00	1.54	-1.00	-1.00	-1.00	-1.00	A
237.00	237.00	242.00	0.00	6.46	-1.00	-1.00	-1.00	-1.00	A
242.00	242.00	245.62	0.00	-0.00	-1.00	-1.00	-1.00	-1.00	A
249.00	249.00	252.94	0.00	3.38	-1.00	-1.00	-1.00	-1.00	A
255.00	255.00	258.85	0.00	2.06	-1.00	-1.00	-1.00	-1.00	A
263.00	263.00	265.72	0.00	4.15	-1.00	-1.00	-1.00	-1.00	A
272.00	272.00	280.29	0.00	6.28	-1.00	-1.00	-1.00	-1.00	A
275.00	-1.00	-1.00	-1.00	-1.00	275.00	312.87	0.00	44.64	B
280.29	280.29	283.34	7.29	0.00	-1.00	-1.00	-1.00	-1.00	NaN
283.34	283.34	288.09	0.34	0.00	-1.00	-1.00	-1.00	-1.00	NaN
288.09	288.09	291.31	1.09	-0.00	-1.00	-1.00	-1.00	-1.00	NaN
297.00	297.00	300.04	0.00	5.69	-1.00	-1.00	-1.00	-1.00	A
301.00	301.00	306.19	0.00	0.96	-1.00	-1.00	-1.00	-1.00	A
306.19	306.19	311.39	1.19	0.00	-1.00	-1.00	-1.00	-1.00	NaN
312.87	-1.00	-1.00	-1.00	-1.00	312.87	336.33	2.87	-0.00	NaN
314.00	314.00	320.43	0.00	2.61	-1.00	-1.00	-1.00	-1.00	A
320.43	320.43	324.77	3.43	-0.00	-1.00	-1.00	-1.00	-1.00	NaN
324.77	324.77	330.54	3.77	0.00	-1.00	-1.00	-1.00	-1.00	NaN
330.54	330.54	333.23	0.54	0.00	-1.00	-1.00	-1.00	-1.00	NaN
336.33	-1.00	-1.00	-1.00	-1.00	336.33	355.29	43.33	-0.00	NaN
340.00	340.00	342.28	0.00	6.77	-1.00	-1.00	-1.00	-1.00	A
353.00	353.00	357.92	0.00	10.72	-1.00	-1.00	-1.00	-1.00	A
355.29	-1.00	-1.00	-1.00	-1.00	355.29	357.78	10.29	-0.00	NaN
357.92	357.92	361.35	1.92	0.00	-1.00	-1.00	-1.00	-1.00	NaN
367.00	367.00	371.26	0.00	5.65	-1.00	-1.00	-1.00	-1.00	A
374.00	374.00	376.23	0.00	2.74	-1.00	-1.00	-1.00	-1.00	A
378.00	378.00	383.39	0.00	1.77	-1.00	-1.00	-1.00	-1.00	A
383.39	383.39	385.32	0.39	-0.00	-1.00	-1.00	-1.00	-1.00	NaN

65 rows × 9 columns

Calculation For Perfomance

In [172]:
```
waiting = df['Type A']['wating Time'].values
A_avg_waiting_time = np.mean(waiting[waiting>0])
```

In [173]:
```
waiting = df['Type B']['wating Time'].values
B_avg_waiting_time = np.mean(waiting[waiting>0])
```

In [174]:
```
idle = df['Type A']['idle Time'].values
A_avg_idle_time = np.mean(waiting[waiting>0])
```

In [175]:
```
waiting = df['Type B']['idle Time'].values
B_avg_idle_time = np.mean(waiting[waiting>0])
```

Performance

In [176]:
```
A_avg_waiting_time
```

Out[176]: 2.194375

In [177]:
```
B_avg_waiting_time
```

Out[177]: 20.351428571428574

In [178]:
```
A_avg_idle_time
```

Out[178]: 20.351428571428574

In [179]:
```
B_avg_idle_time
```

Out[179]: 25.879999999999995