學號：N26111871 姓名：廖威任

```
5   lam = int(input("請輸入lambda:")) #Arrival Rate
6   mu = int(input("請輸入mu:")) #Service Time
7   t = int(input("請輸入t:")) #營業時間的長度
```

首先輸入 lambda、mu 及 t，分別代表 Arrival Rate、Service Time 及營業時間的長度。

```
 9   arrival = [0] * lam #每位顧客的抵達時間
10   service = [0] * lam #每位顧客的服務時間
11   start_service = [0] * lam #每位顧客的開始服務時間
12   end_service = [0] * lam #每位顧客的結束服務時間
```

利用 list 儲存每位顧客的各項活動時間。

```
14  for i in range(lam): #利用lambda和mu算出每位顧客的抵達時間和服務時間
15      if i == 0:
16          arrival[i] = (-1 / lam) * math.log(random.random())
17      else:
18          arrival[i] = arrival[i - 1] + (-1 / lam) * math.log(random.random())
19
20      service[i] = (-1 / mu) * math.log(random.random())
```

利用 $X = -\frac{1}{\lambda}\log U$ 公式產生出 Exponential random variable，帶入 lambda 和 mu 後計算出每位顧客的抵達時間和服務時間。

```python
#第一位顧客的開始服務時間和結束服務時間
start_service[0] = arrival[0]
end_service[0] = arrival[0] + service[0]
total_service = 1 #總共服務多少位顧客

for i in range(1, lam):
    #若前一位顧客已經離開且商店還沒打烊
    if end_service[i - 1] <= arrival[i] and arrival[i] <= 1:
        start_service[i] = arrival[i] #則顧客抵達後可直接被服務
        total_service += 1
    #若前一位顧客還沒離開且商店還沒打烊
    elif end_service[i - 1] > arrival[i] and arrival[i] <= 1:
        start_service[i] = end_service[i - 1] #則顧客排隊等待前一位顧客離開
        total_service += 1
    #商店打烊
    else:
        total_service = i
        break

    end_service[i] = start_service[i] + service[i]
```

接著計算每位顧客開始服務的時間和結束服務時間，第一位顧客不需要排隊因此另外計算，其他顧客則視前一位顧客的情況決定是否需要排隊，若商店打烊則無法接受服務。

```python
for i in range(lam): #將時間單位調整程適當大小
    arrival[i] *= t
    service[i] *= t
    start_service[i] *= t
    end_service[i] *= t
```

將時間單位調整成適當大小以方便理解。

```python
#利用數學公式計算出utilization
math_utilization = lam / mu
print("Server utilization(math):", math_utilization)

#利用模擬的數據計算出utilization
busy = 0
for i in range(total_service):
    busy += end_service[i] - start_service[i]
simulation_utilization = busy / t
print("Server utilization(simulation):", simulation_utilization)
```

利用兩種方法計算 utilization。

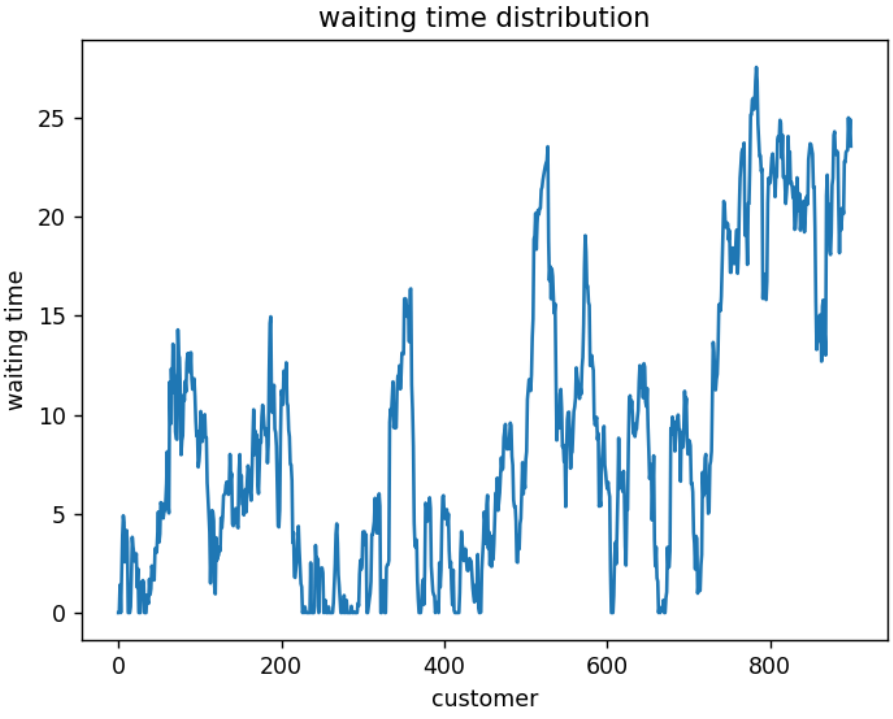
利用公式：utilization = lambda/mu 計算出 utilization。

另外再用前面所模擬出的數據計算出店員的 busy time，將 busy time 除以 t 即為 utilization。

```
61  waiting = [0] * total_service
62  for i in range(total_service): #計算出每位顧客的等待時間
63      waiting[i] = start_service[i] - arrival[i]
64  avg_waiting = sum(waiting) / total_service #計算出平均的等待時間
65  print("Average waiting time:",avg_waiting)
```

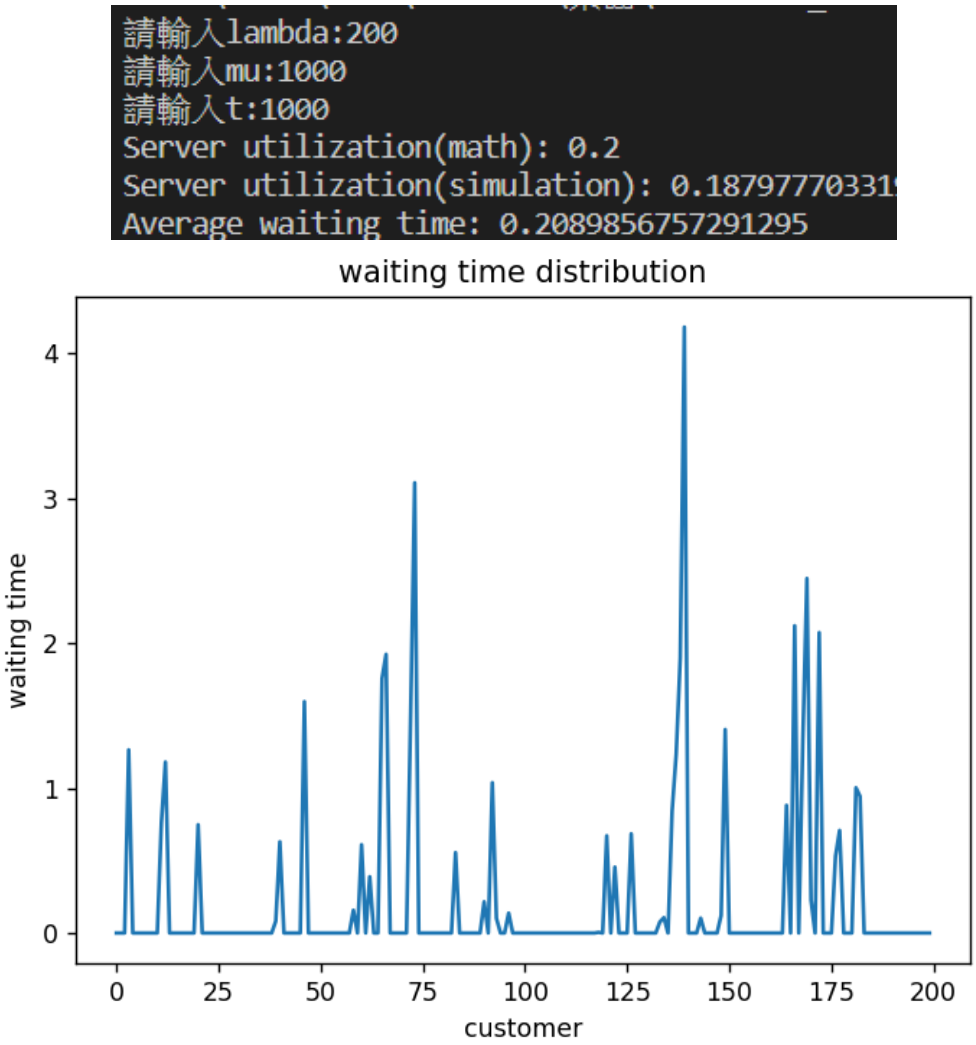
統計出每位顧客的等待時間後取平均計算出平均等待時間。

```
請輸入lambda:900
請輸入mu:1000
請輸入t:1000
Server utilization(math): 0.9
Server utilization(simulation): 0.9081765
Average waiting time: 8.9220467680585
```



首先討論當 lambda=900、mu=1000、t=1000 時的情況，代表顧客較

多，此時利用 math 和 simulation 兩種方法計算出的 utilization 皆約為 0.9，圖表顯示每位顧客的 waiting time 較久，需要排隊的顧客也較多。



接著討論當 lambda=200、mu=1000、t=1000 時的情況，代表顧客較少，此時利用 math 和 simulation 兩種方法計算出的 utilization 皆約為 0.2，圖表顯示每位顧客的 waiting time 較短，需要排隊的顧客也較少。