1.
a)
(SLEEP mode)
speeds collected from the ten tests:
[1005.398834, 1002.863188, 1001.940431, 1000.284042, 1000.307409, 1000.389646, 1000.215066, 1000.304209, 1000.457028, 1000.191596]

Max: 1005.398834
Min: 1000.191596
Average: 1001.2351449
Standard deviation: 1.7180764950341254

b)
(BUSYWAIT mode)
speeds collected from the ten tests:
[999.940917, 999.97125, 999.980528, 999.983229, 999.984925, 999.988355, 999.992459, 999.99262, 999.993274, 999.993321]

Max: 999.993321
Min: 999.940917
Average: 999.9820878
Standard deviation: 0.01610162301550046

c)
BUSYWAIT method yields more stable (but not precise) results as the test results have a smaller standard deviation compared to the SLEEP mode.

The CPU clock speed for M1 Pro is 2060 - 3220 MHZ, which is different from the estimate. After research, I believe this may be due to the program is running under a container. According to web, TSC is a register present on x86 processors to count the number of CPU cycles. M1 pro is not a x86 processor, so it may not have this exact component. Rather, maybe the container simulates the x86 environment for it. So the results may not be precise. Another reason could be the mechanism from M1pro to go into low-power mode.

2.
a)
receipt timestamp of the first request: 123.377618
completion timestamp of the last request: 174.690871
window size = 51.313253
number of requests completed = 500

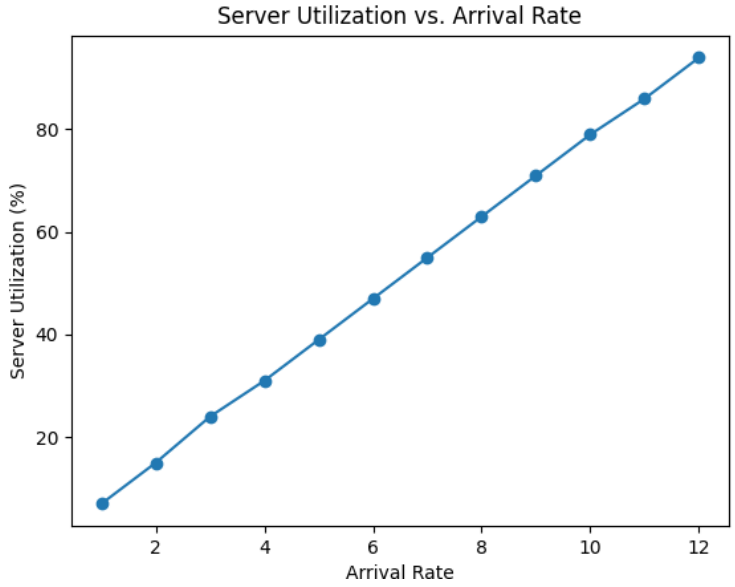
throughput = 500/51.313253 = 9.7440713805

b)
I used the Python to read and sparse the text value to float to add up all the request length (server is busy) and get 40.664097.
40.664097/51.313253 = 79.25%

c)
Yes, they match.

```
System time (seconds): 0.02
Percent of CPU this job got: 79%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:51.34
Average shared text size (kbytes): 0
```

d)



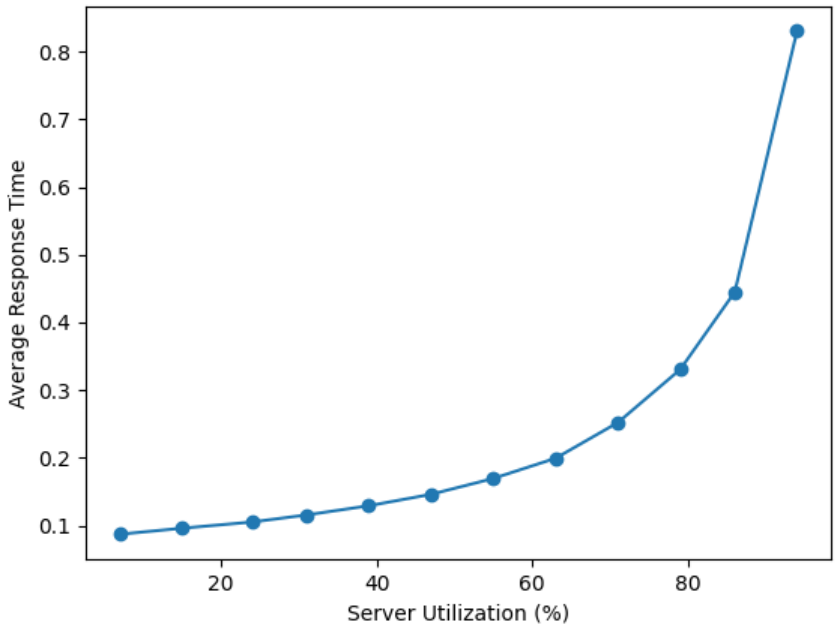
Before reaching 100% utilization, the higher arrival rate of requests yields higher server utilization. And it's a linear relationship.

e)
Average response time (response time for each request calculated by completion time - sent time): 0.33057177499392854

max = 1.4475961680000182

min = 0.0003985420007666107
standard deviation = 0.31427077373961215

f)



As the Server utilization goes higher, Average response time also increases. Besides, when the utilization is higher, the average response time increases in a faster speed than when utilization is lower.