Regarding question-1 :

<time is captured using python time library, The time.time() function returns the number of seconds >

For Question:-1

Key	Req1 (first time)	Req2 (second time)	Req3 (third time)	Average Time
Key1	0.00838	.00713	.00733	0.00761
Key2	0.00666	0.00764	.00438	0.00623
Key3	0.00327	0.00384	0.00575	0.00429
Key4	0.00640	0.00528	0.00742	0.00637
Key5	0.00602	0.00656	0.00697	0.00652
Key6	0.00810	0.00671	0.00700	0.00727

Observations and justification:
1. Average time required to access key is in milli seconds and (approximately) equal for each key

2. Little difference in access times of keys observed due to python dictionary operations (because in getting keys , we are using has_key(), split() function
   For has_key() - time complexity for worst case is O(n).while time complexity for split() function is O(n))
   Because of this little difference is observed between different keys access time

For question-2 :

Key	Req1 (first time)	Req2 (second time)	Req3 (third time)
Key1	0.03575	0.00774	0.00751
Key2	0.03999	0.00836	0.00580
Key3	0.01126	0.00880	0.00855
Key4	0.03474	0.00433	0.00916
Key5	0.01402	0.00824	0.00677
Key6	0.03821	0.00743	0.00621
Average Time	0.028995	0.00748	0.00733

Observations and justification:
1. Time taken to get keys first time is in 10^-2 while for second and third time it is in milliseconds
   a. This difference is because , when we access first key is not present in cache , so cache has to access from server and then client need can be fulfill
      But when we access second and third time , key is present in cache and can be access directly from cache , that's why it took less time
   <main purpose of cache is to storing data for re-access in a device's memory, so that it can be accessed speedily, but in first time it took more time , as cache accessed from server and then cached it in memory which took time , but further access of same data take very less time as we can see)
2. Time for second and third access of keys is almost same (little difference is due to python dictionary operations, same as above)