1、对以下每个应用程序，你认为最多一次语义和最少一次语义哪个更好？

(a)从文件服务器上读写文件；

最少一次

(b)编译一个程序；

最多一次

(c)远程银行；

最少一次

2、简述Flooding、PAXOS、RAFT、PBFT、PoW的使用场景？

- Flooding：
  洪泛法被使用在桥接器上，Usenet以及点对点档案分享等，通常被应用在网络互相交叉数量较少的场景，比如上级部门单向地向各个下级部门发送任务文件。
- PAXOS：
  Paxos最大的用途就是保持多个节点数据的一致性，通常要在分布式系统的各个进程就某个值达成一致，如应用在分布式文件管理系统。
- RAFT：
  Raft 在实际应用场景中的一致性更多的是体现在不同节点之间的数据一致性，和PAXOS算法应用环境类似，但需要保证能有让日志记录的文件。
- PBFT：
  PBFT在很多场景都有应用，在区块链场景中，一般适合于对强一致性有要求的私有链和联盟链场景。或者说应用在容易受到恶意攻击以及软件比较容易发生错误的分布式系统中。
- PoW：
  POW算法一般应用在拥有能量化工作量的环境，如比特币竞争的系统。

3、任意选择一种编程语言实现的RAFT程序，运行该程序并测试记录结果。

- 我选择python实现的RAFT程序，即第二个链接对应的程序

- 测试Memory_Board

```
PS E:\code\py\simpleRaft-master\simpleRaft> python .\test_MemoryBoard.py
test_memoryboard_post_message (__main__.TestMemoryBoard) ... ok
test_memoryboard_post_message_make_sure_they_are_ordered (__main__.TestMemoryBoard) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.005s

OK
```

- 测试LeadServer

```
PS E:\code\py\simpleRaft-master\simpleRaft> python .\test_LeaderServer.py
test_leader_server_sends_appendentries_to_all_neighbors_and_is_appended_to_their_logs (__main__.TestLeaderServer) ... ok
test_leader_server_sends_appendentries_to_all_neighbors_but_some_have_dirtied_logs (__main__.TestLeaderServer) ... ok
test_leader_server_sends_heartbeat_to_all_neighbors (__main__.TestLeaderServer) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.011s

OK
```

- 测试FollowerServer

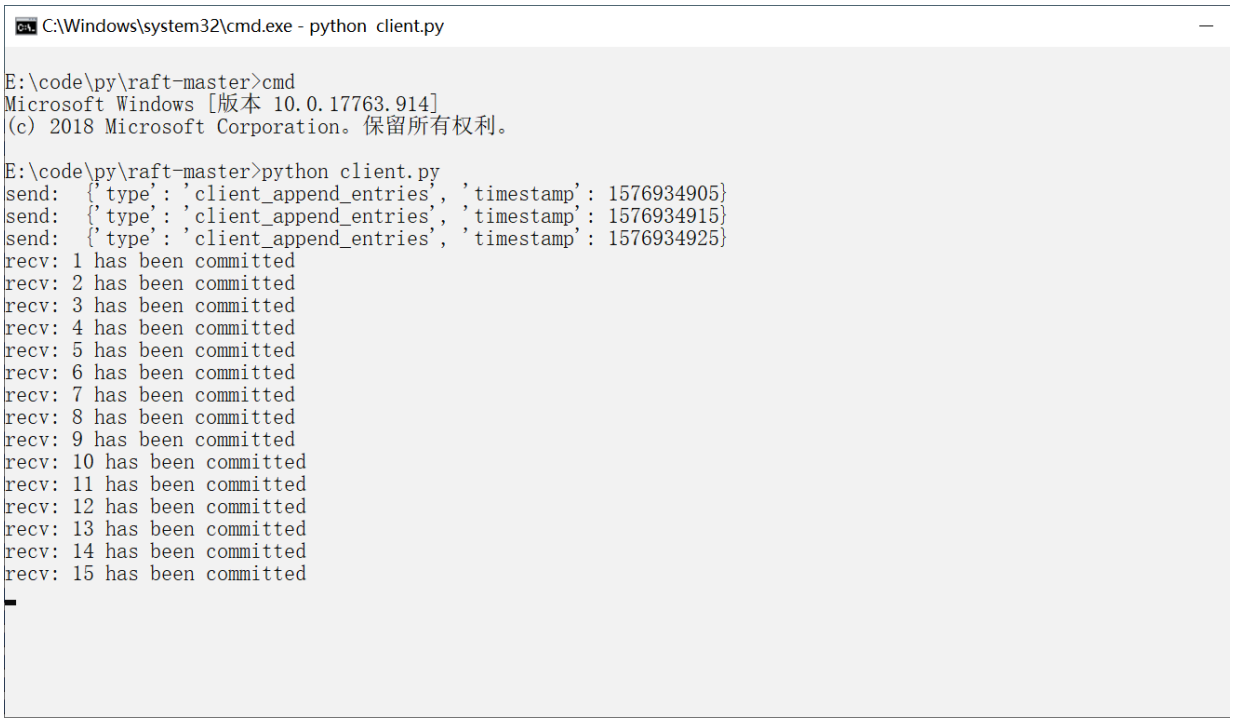
```
PS E:\code\py\simpleRaft-master\simpleRaft> python .\test_FollowerServer.py
test_follower_server_on_message (__main__.TestFollowerServer) ... ok
test_follower_server_on_receive_message_where_log_contains_conflicting_entry_at_new_index (__main__.TestFollowerServer) ...
ok
test_follower_server_on_receive_message_where_log_does_not_have_prevLogTerm (__main__.TestFollowerServer) ... ok
test_follower_server_on_receive_message_where_log_is_empty_and_receives_its_first_value (__main__.TestFollowerServer) ... ok

test_follower_server_on_receive_message_with_greater_term (__main__.TestFollowerServer) ... ok
test_follower_server_on_receive_message_with_lesser_term (__main__.TestFollowerServer) ... ok
test_follower_server_on_receive_vote_request_after_sending_a_vote (__main__.TestFollowerServer) ... ok
test_follower_server_on_receive_vote_request_message (__main__.TestFollowerServer) ... ok

----------------------------------------------------------------------
Ran 8 tests in 0.010s

OK
```

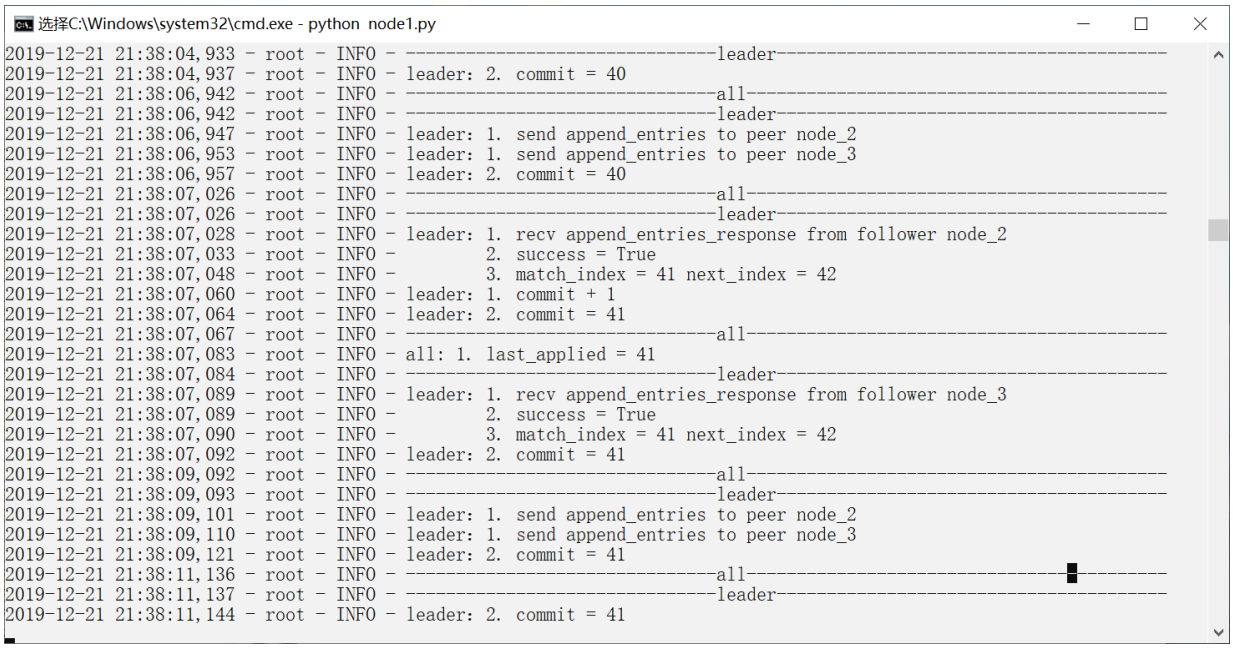
- 测试CandidateServer

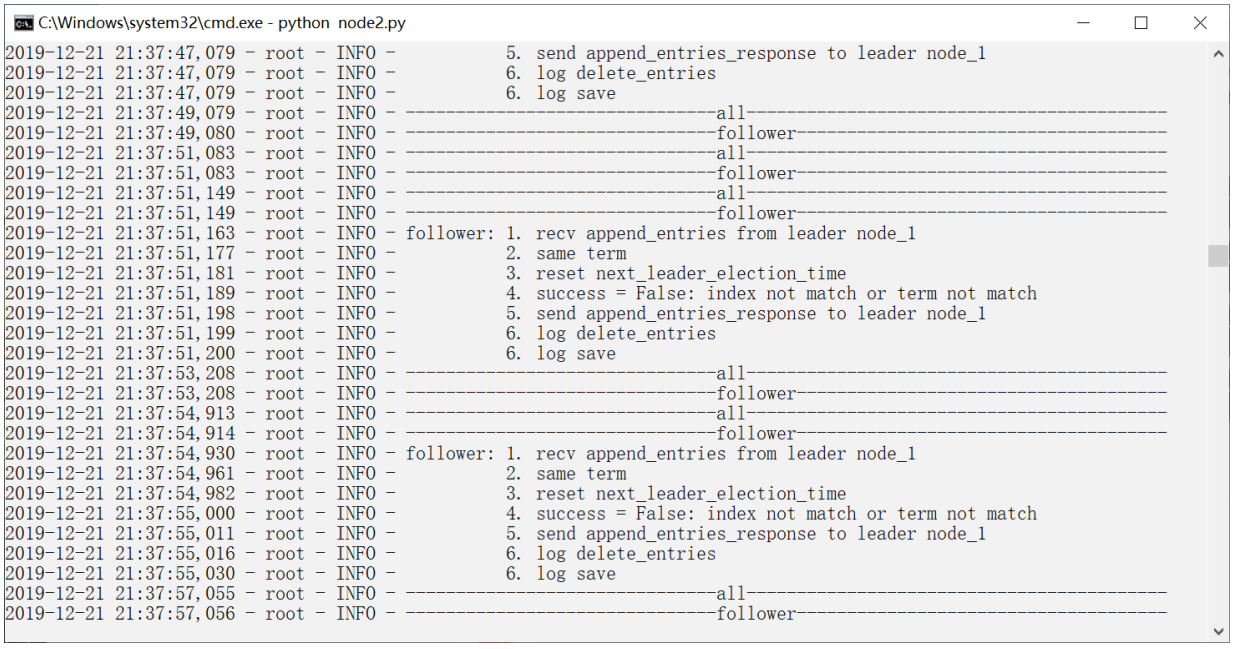
```
PS E:\code\py\simpleRaft-master\simpleRaft> python .\test_CandidateServer.py
test_candidate_fails_to_win_election_so_resend_request (__main__.TestCandidateServer) ... ok
test_candidate_server_had_gotten_the_vote (__main__.TestCandidateServer) ... ok
test_candidate_server_had_intiated_the_election (__main__.TestCandidateServer) ... ok
test_candidate_server_wins_election (__main__.TestCandidateServer) ... ok
test_multiple_candidates_fail_to_win_so_resend_requests (__main__.TestCandidateServer) ... ok
test_two_candidates_one_wins (__main__.TestCandidateServer) ... ok
test_two_candidates_tie (__main__.TestCandidateServer) ... ok

----------------------------------------------------------------------
Ran 7 tests in 0.012s

OK
```

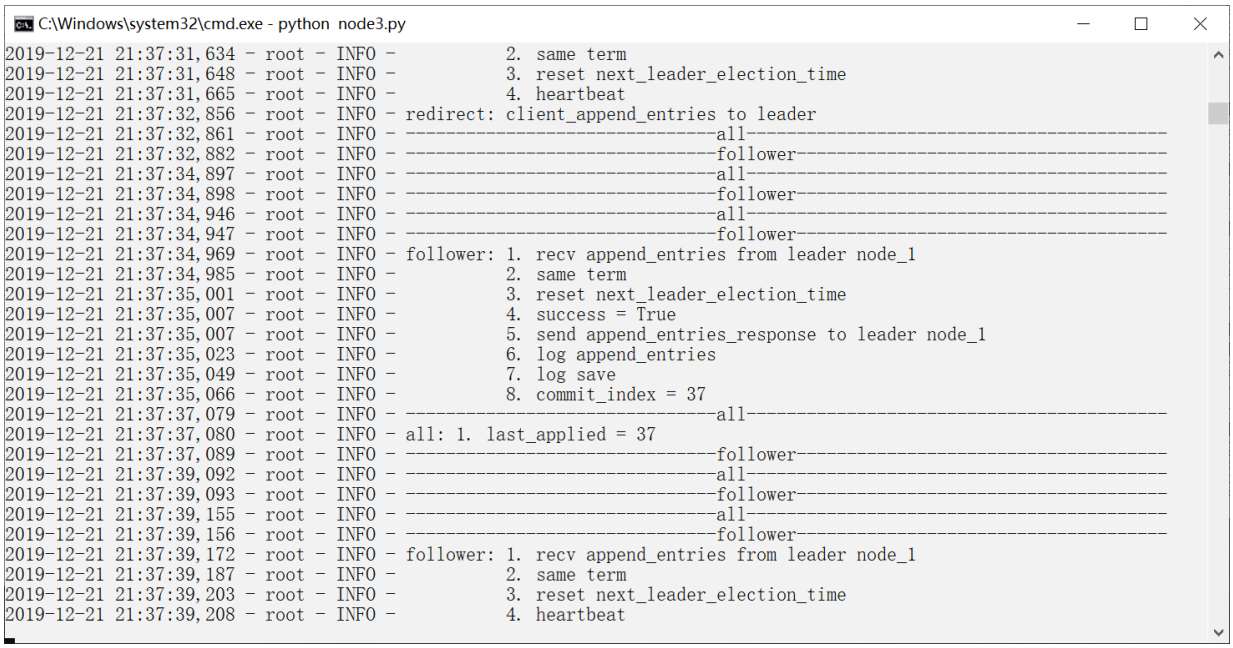
- 同样用python实现的RAFT程序还有这个。

  - 依次运行 client.py，node1.py，node2.py，node3.py，截图如下：
    Leader：负责更新log数据，会先更新自己的本地的数据，然后同步到每个副本。
    Follower：负责接收Leader的更新请求，更新自己本地的log数据。


C:\Windows\system32\cmd.exe - python client.py
E:\code\py\raft-master>cmd
Microsoft Windows [版本 10.0.17763.914]
(c) 2018 Microsoft Corporation。保留所有权利。
E:\code\py\raft-master>python client.py
send: {'type': 'client_append_entries', 'timestamp': 1576934905}
send: {'type': 'client_append_entries', 'timestamp': 1576934915}
send: {'type': 'client_append_entries', 'timestamp': 1576934925}
recv: 1 has been committed
recv: 2 has been committed
recv: 3 has been committed
recv: 4 has been committed
recv: 5 has been committed
recv: 6 has been committed
recv: 7 has been committed
recv: 8 has been committed
recv: 9 has been committed
recv: 10 has been committed
recv: 11 has been committed
recv: 12 has been committed
recv: 13 has been committed
recv: 14 has been committed
recv: 15 has been committed

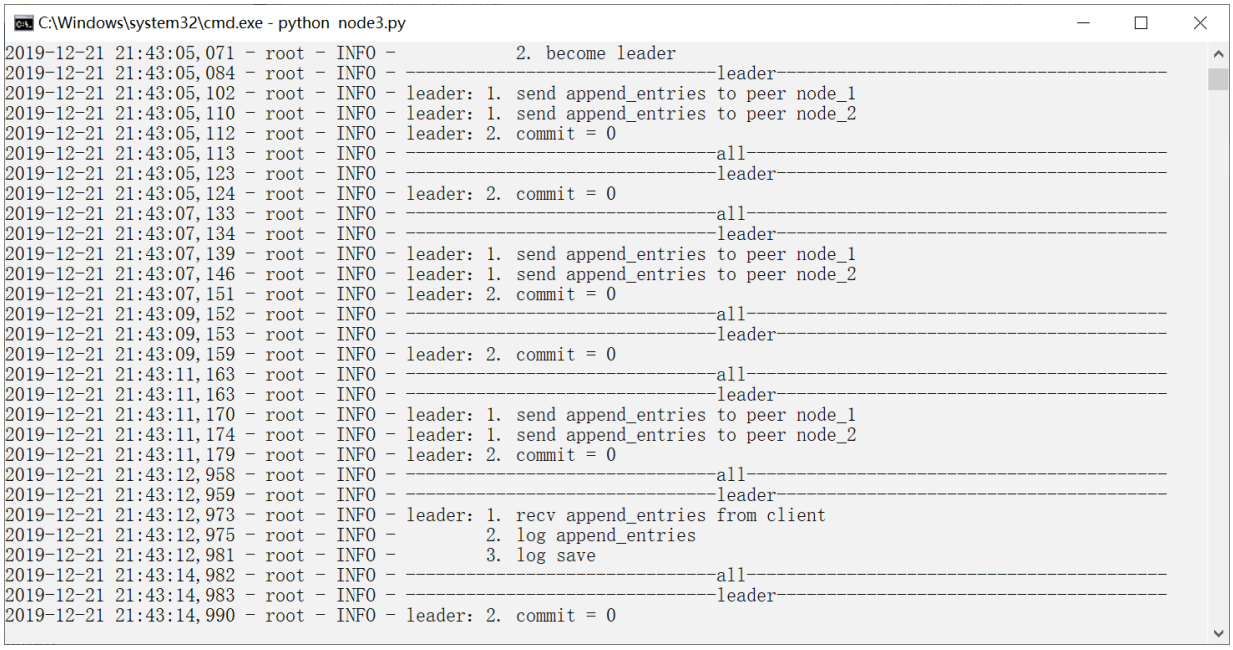

选择C:\Windows\system32\cmd.exe - python node1.py
2019-12-21 21:38:04,933 - root - INFO - ----------leader----------
2019-12-21 21:38:04,937 - root - INFO - leader: 2. commit = 40
2019-12-21 21:38:06,942 - root - INFO - ----------all----------
2019-12-21 21:38:06,942 - root - INFO - ----------leader----------
2019-12-21 21:38:06,947 - root - INFO - leader: 1. send append_entries to peer node_2
2019-12-21 21:38:06,953 - root - INFO - leader: 1. send append_entries to peer node_3
2019-12-21 21:38:06,957 - root - INFO - leader: 2. commit = 40
2019-12-21 21:38:07,026 - root - INFO - ----------all----------
2019-12-21 21:38:07,026 - root - INFO - ----------leader----------
2019-12-21 21:38:07,028 - root - INFO - leader: 1. recv append_entries_response from follower node_2
2019-12-21 21:38:07,033 - root - INFO - 2. success = True
2019-12-21 21:38:07,048 - root - INFO - 3. match_index = 41 next_index = 42
2019-12-21 21:38:07,060 - root - INFO - leader: 1. commit + 1
2019-12-21 21:38:07,064 - root - INFO - leader: 2. commit = 41
2019-12-21 21:38:07,067 - root - INFO - ----------all----------
2019-12-21 21:38:07,083 - root - INFO - all: 1. last_applied = 41
2019-12-21 21:38:07,084 - root - INFO - ----------leader----------
2019-12-21 21:38:07,089 - root - INFO - leader: 1. recv append_entries_response from follower node_3
2019-12-21 21:38:07,089 - root - INFO - 2. success = True
2019-12-21 21:38:07,090 - root - INFO - 3. match_index = 41 next_index = 42
2019-12-21 21:38:07,092 - root - INFO - leader: 2. commit = 41
2019-12-21 21:38:09,092 - root - INFO - ----------all----------
2019-12-21 21:38:09,093 - root - INFO - ----------leader----------
2019-12-21 21:38:09,101 - root - INFO - leader: 1. send append_entries to peer node_2
2019-12-21 21:38:09,110 - root - INFO - leader: 1. send append_entries to peer node_3
2019-12-21 21:38:09,121 - root - INFO - leader: 2. commit = 41
2019-12-21 21:38:11,136 - root - INFO - ----------all----------
2019-12-21 21:38:11,137 - root - INFO - ----------leader----------
2019-12-21 21:38:11,144 - root - INFO - leader: 2. commit = 41

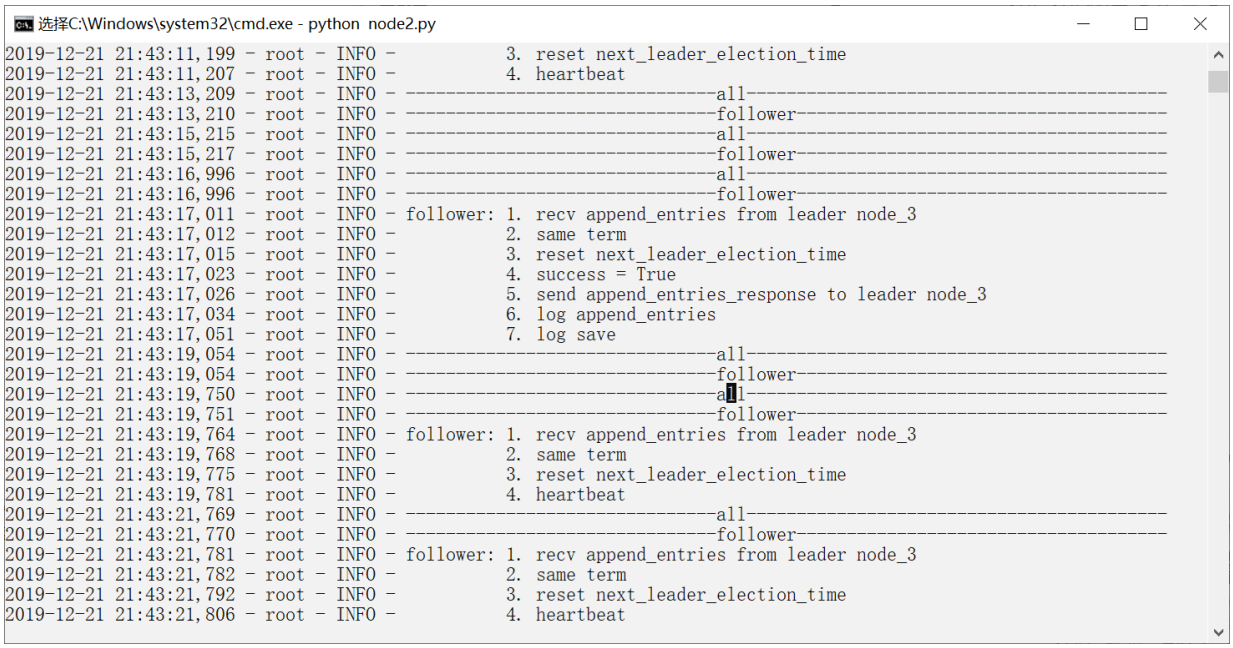

C:\Windows\system32\cmd.exe - python node2.py
2019-12-21 21:37:47,079 - root - INFO - 5. send append_entries_response to leader node_1
2019-12-21 21:37:47,079 - root - INFO - 6. log delete_entries
2019-12-21 21:37:47,079 - root - INFO - 6. log save
2019-12-21 21:37:49,079 - root - INFO - ----------all----------
2019-12-21 21:37:49,080 - root - INFO - ----------follower----------
2019-12-21 21:37:51,083 - root - INFO - ----------all----------
2019-12-21 21:37:51,083 - root - INFO - ----------follower----------
2019-12-21 21:37:51,149 - root - INFO - ----------all----------
2019-12-21 21:37:51,149 - root - INFO - ----------follower----------
2019-12-21 21:37:51,163 - root - INFO - follower: 1. recv append_entries from leader node_1
2019-12-21 21:37:51,177 - root - INFO - 2. same term
2019-12-21 21:37:51,181 - root - INFO - 3. reset next_leader_election_time
2019-12-21 21:37:51,189 - root - INFO - 4. success = False: index not match or term not match
2019-12-21 21:37:51,198 - root - INFO - 5. send append_entries_response to leader node_1
2019-12-21 21:37:51,199 - root - INFO - 6. log delete_entries
2019-12-21 21:37:51,200 - root - INFO - 6. log save
2019-12-21 21:37:53,208 - root - INFO - ----------all----------
2019-12-21 21:37:53,208 - root - INFO - ----------follower----------
2019-12-21 21:37:54,913 - root - INFO - ----------all----------
2019-12-21 21:37:54,914 - root - INFO - ----------follower----------
2019-12-21 21:37:54,930 - root - INFO - follower: 1. recv append_entries from leader node_1
2019-12-21 21:37:54,961 - root - INFO - 2. same term
2019-12-21 21:37:54,982 - root - INFO - 3. reset next_leader_election_time
2019-12-21 21:37:55,000 - root - INFO - 4. success = False: index not match or term not match
2019-12-21 21:37:55,011 - root - INFO - 5. send append_entries_response to leader node_1
2019-12-21 21:37:55,016 - root - INFO - 6. log delete_entries
2019-12-21 21:37:55,030 - root - INFO - 6. log save
2019-12-21 21:37:57,055 - root - INFO - ----------all----------
2019-12-21 21:37:57,056 - root - INFO - ----------follower----------



可见，node1当选为leader，node2，node3为follower。

- 关闭node1
  - Raft采用心跳机制来触发Leader的选举过程。Leader通过周期性地向其他服务器发送心跳来保持其Leader地位。
  - 如果Follower过了一定时间段没有接收到任何心跳信息，则会认为Leader已经不存在（Leader可能是宕机，也可能是网络中断），然后就会启动新的Leader的选举过程。
  - Candidate：如果Follower一段时间没有收到Leader的心跳，那么就会认为系统中没有Leader，于是它就会把自己的状态变成Candidate，开始Leader的选举过程，直到选举结束，会一直处于Candidate这个状态。
  - 由于实在太快处于candidate的截图，并未截到。





node2，node3重新竞选，node3当选leader。