# Part 3 Report

**Repo URL** : https://github.com/nikhil-01a/567FinalProject

**Related Source File:**

- perfTest.py: code for executing performance tests and generating csv results
- MRTD.py: the tested code we have to do perf testing on two given files records_encoded.json and records_decoded.json.

  We are using same functionality from part 2, but we need to rearrange some functions so that it'll easy to handle strings. we have combined few functions and add other functionality for encode and decode.

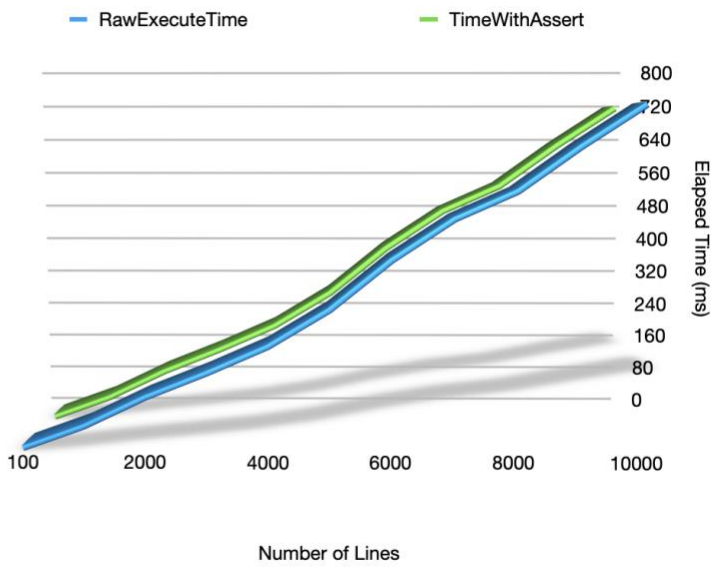
**Function Implementation:**

- **class HardwareScanner():** An empty class function for hardware scanner that will be mocked to return actual values during testing.
- **decode():** A function that takes 2 MRZ Strings as its inputs from the mocked HardwareScanner() and returns a string of '4 check digits concatenated together' and a dictionary of all the fields.
- **class dummydatabase():** An empty class function for database scanner that will be mocked to return actual values during testing.
- **encode():** A function that takes 2 dictionary lines as input from the mocked dummydatabase() and returns a string of ' 4 check digits concatenated together' and a string of MRZ lines joined by a semi-colon ';'.
- **getCheckCode(inputString: str) -> int**: return the correct check code for the given string.
- **compare_EndDec():** A function that compares the check digit output of both encode() and decode() and then it reports if any mismatch is found in the corresponding fields of those check digits.

## Excel File:

output

| NumOfLines | RawExecuteTime | TimeWithAssert |
|---|---|---|
| 100 | 5.483875051140790 | 5.509332986548540 |
| 1000 | 51.18941701948640 | 51.21212499216200 |
| 2000 | 111.83858301956200 | 111.86083301436200 |
| 3000 | 162.34487504698300 | 162.3672500718390 |
| 4000 | 217.78520802035900 | 217.80770807527000 |
| 5000 | 294.0003330586480 | 294.02245802339200 |
| 6000 | 396.25129208434400 | 396.27408306114400 |
| 7000 | 479.708541999571 | 479.7310420544820 |
| 8000 | 535.844707977958 | 535.8672080328690 |
| 9000 | 626.4325000811370 | 626.4559170231220 |
| 10000 | 708.911000052467 | 708.933167043142 |




## Explanation and Observation:

The above chart shows the trends of timestamps vs the number of executed codes. Based on the generated chart we can see that for both raw execute time and execute time with assertions, the trend is similar and grow linearly, while the timestamp of execute time with assertions are generally greater than the timestamp of raw executing time, the difference is minimal.