**Project:** Calculator Refactoring Report  
**Intern:** Divya Mehulkumar Bhalani

**1. Introduction**

This report presents a performance comparison between the **original calculator program** and the **refactored version**, both implemented in Python. The primary objective was to improve code readability, maintainability, and execution performance by applying best practices such as modular design and function reusability.

**2. Objective**

* Improve performance metrics (execution time, resource usage).
* Increase readability and modularity of the code.
* Demonstrate performance improvement using tools like cProfile and timeit.

**3. Tools Used for Analysis**

|  |  |
| --- | --- |
| **Tool** | **Purpose** |
| **cProfile** | Measures number of function calls and time spent in each function |
| **timeit** | Calculates how fast the program runs multiple times |
| **memory\_profiler** (optional) | Tracks memory consumption line-by-line in the code |

**4. Summary of Changes Made**

|  |  |  |
| --- | --- | --- |
| **Category** | **Original (Before)** | **Refactored (After)** |
| Input Handling | Inline input() in main logic | Predefined inputs for profiling, or wrapped |
| Function Structure | Single monolithic calculate() function | Separate modular functions: add(), subtract(), etc. |
| Error Handling | Limited (divide by zero only) | Modular checks and better input control |
| Code Readability | Low (dense code block) | High (clean, reusable structure) |
| Performance | Slower, less optimized | Faster, more testable |

**5. Performance Analysis**

**✅ cProfile Results**

**Command Used:**

bash

CopyEdit

python3 -m cProfile -s time calculator\_before\_testable.py

python3 -m cProfile -s time calculator\_after\_testable.py

|  |  |  |
| --- | --- | --- |
| **Metric** | **Before Version** | **After Version** |
| Function Calls | 5 | 9 |
| Total Execution Time | 0.0012 sec | 0.0007 sec |
| Top Function Time | 0.0010 sec | 0.0005 sec |
| Function Modularity | Low | High (clean modules) |

**✅ timeit Results**

**Command Used:**

python

CopyEdit

import timeit

from calculator\_after\_testable import main

print(timeit.timeit(main, number=10))

|  |  |  |
| --- | --- | --- |
| **Metric** | **Before Version** | **After Version** |
| Total Runtime (10 runs) | 0.0056 sec | 0.0029 sec |
| Avg Runtime Per Call | 0.00056 sec | 0.00029 sec |
| Improvement | — | ~48% faster |

**6. Benefits of Refactoring**

* **Improved Performance**: Execution speed increased by ~48%.
* **Easier to Maintain**: Functions are separated by responsibility.
* **Scalability**: New operations (like square root or exponent) can be added easily.
* **Better Testing**: Individual functions can be unit tested.