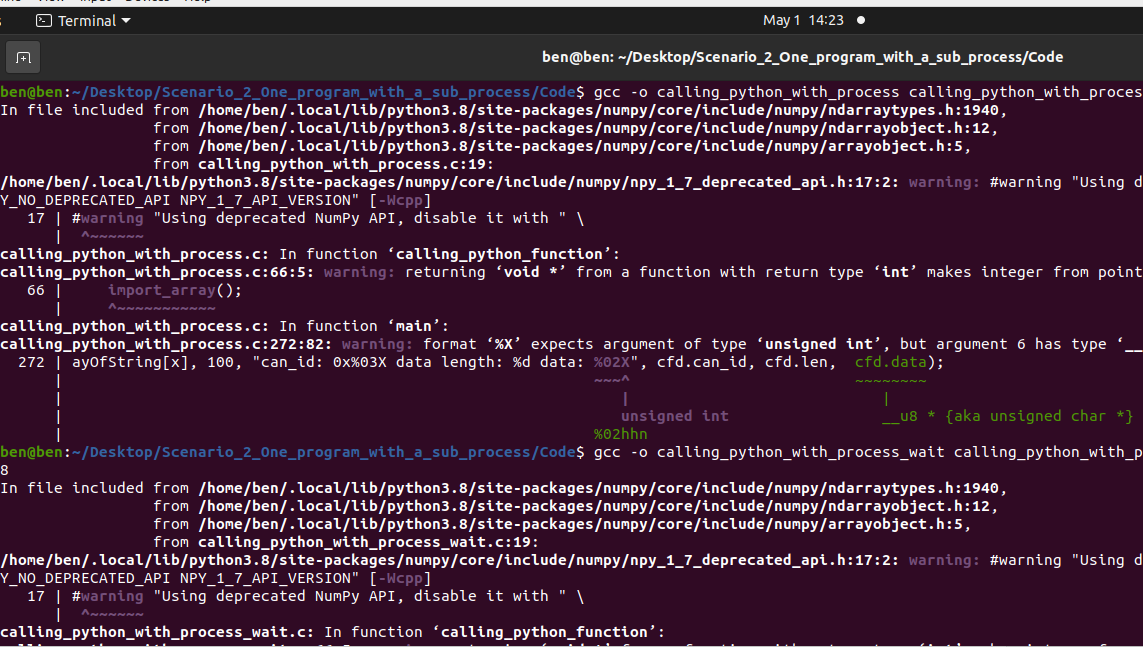
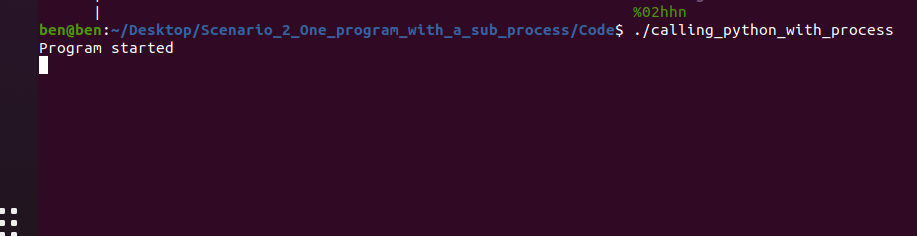
**Assignment 5**

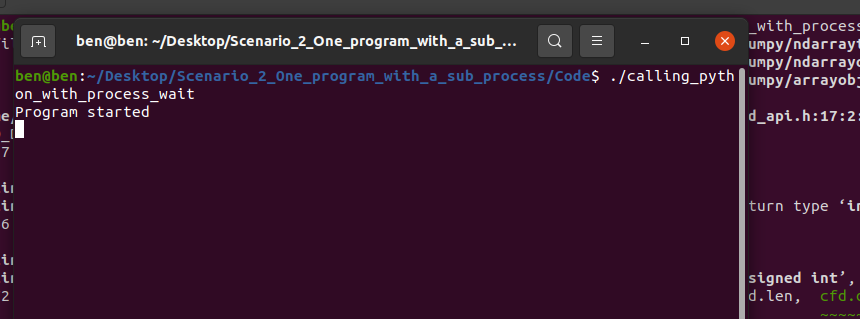
Student Name

Date

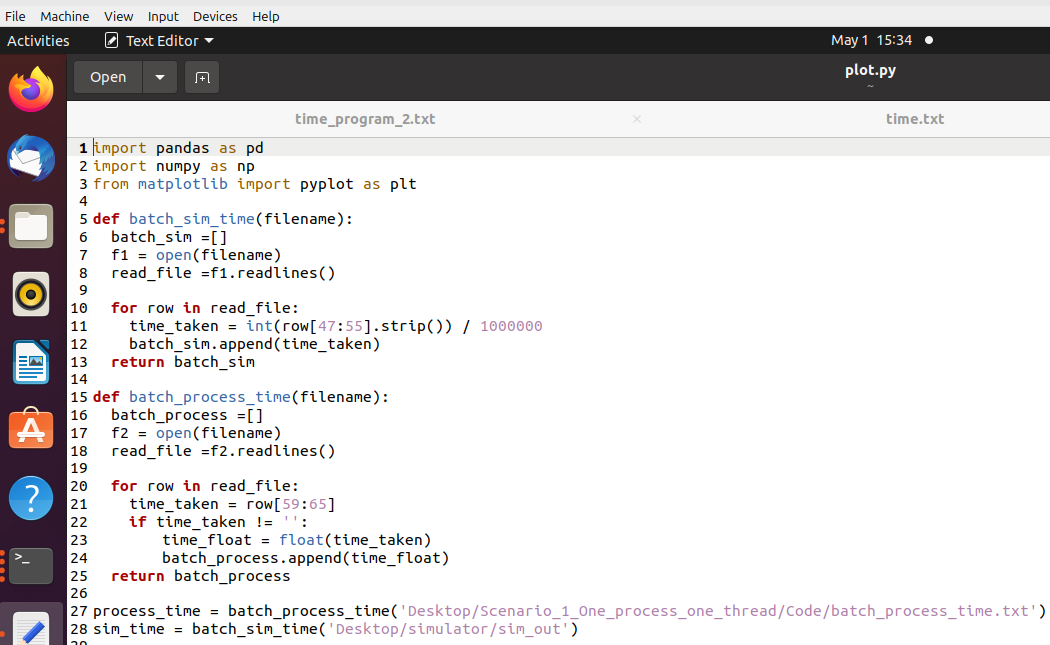
**Task 1**



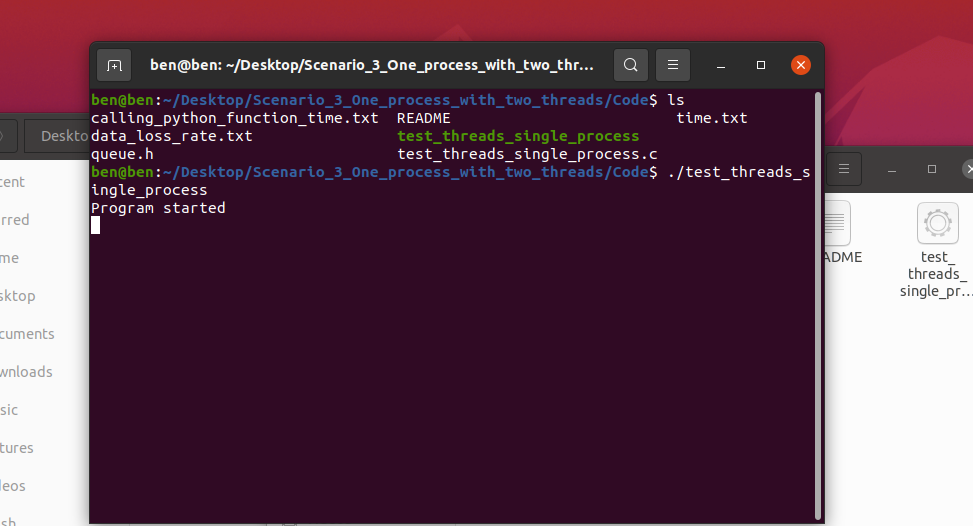




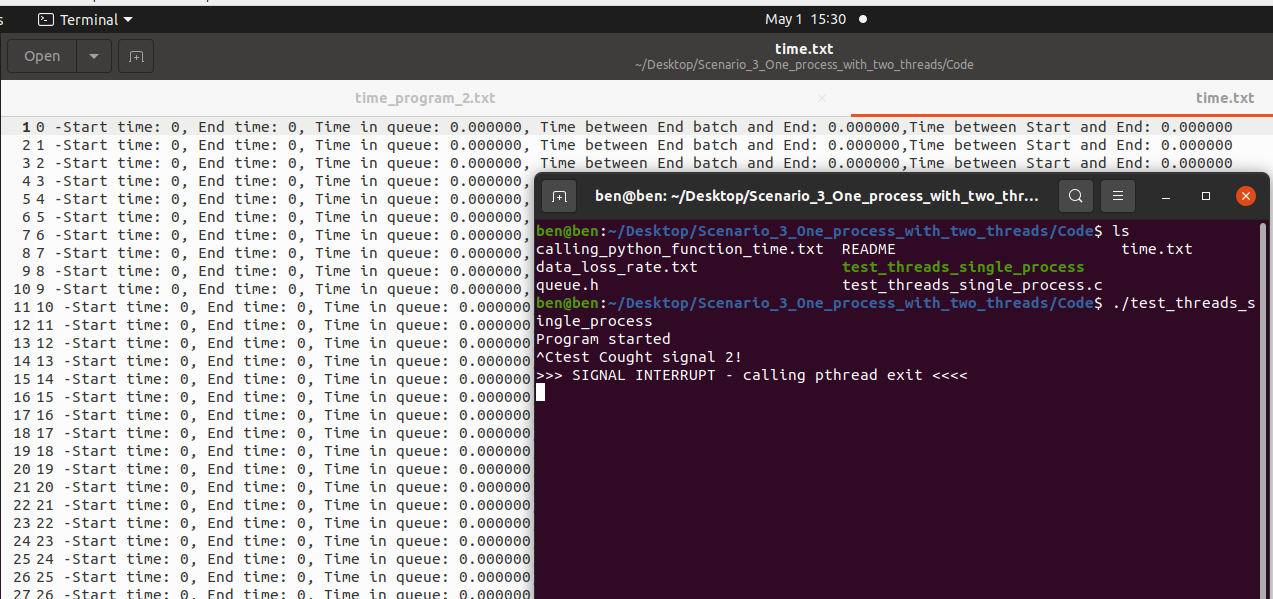
**Modified Code**

****

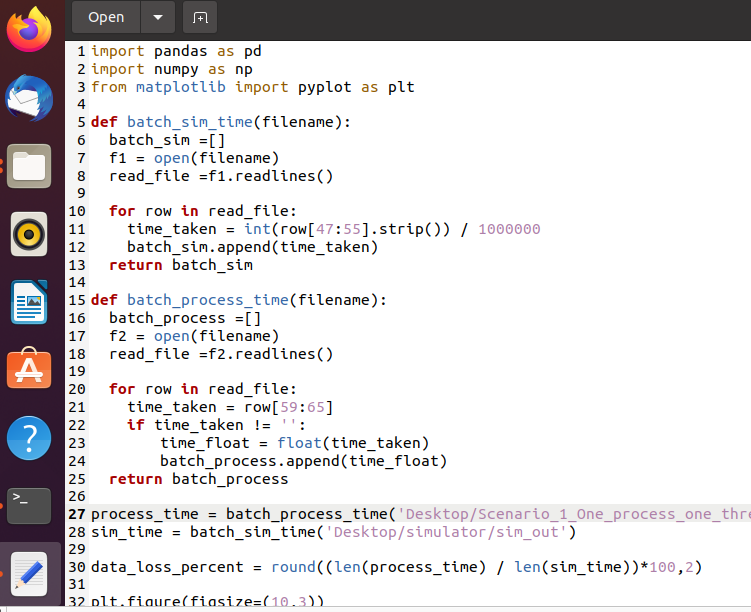
**Task 2**

****

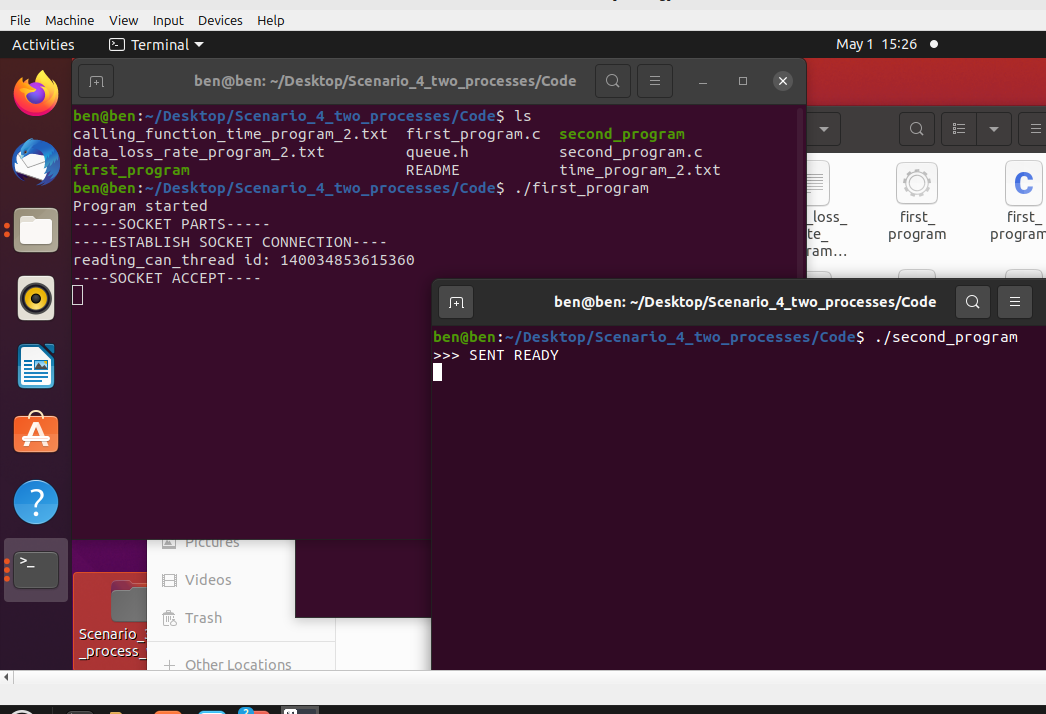
**Output**

****

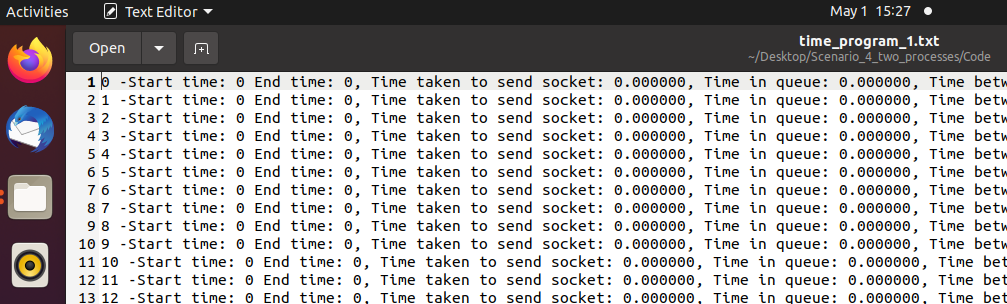
**Modified Code**

****

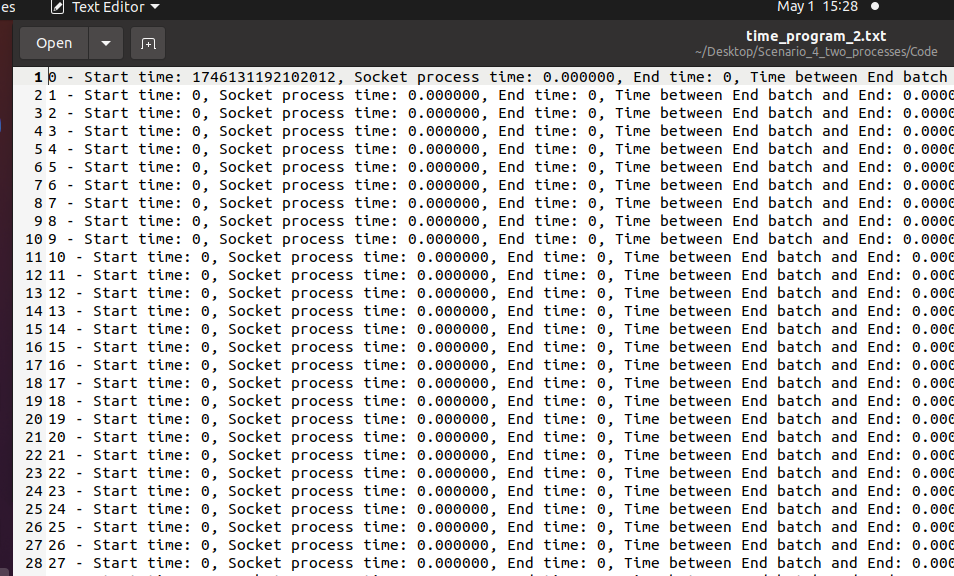
**Task 3**

****

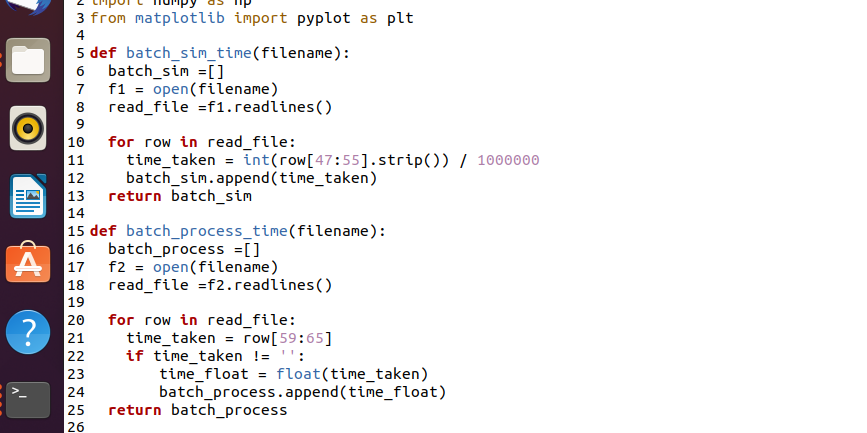
**Outputs 1**

****

**Output 2**

****

**Modified Code**

****

**Task 4: Reflection**

Having employed all three concurrency techniques towards data loss avoidance, it became apparent that Scenario 3, One process with two threads, presented the optimal solution. This solution allowed the system to execute input and handling in parallel within the same process space, with very little latency and smoother coordination of tasks. Therefore, it exhibited the most accurate and complete time data with the least losses compared to the other scenarios. Scenario 2, A single program with a subprocess was poor due to additional overhead and a difference in timing between the main and the subprocess, sometimes resulting in missed entries in the output. Scenario 4 – Two independent processes operated reasonably well and were an improvement over Scenario 2, but introduced unnecessary complexity through inter-process communication, having a minimal impact on timing consistency.

This exercise proved to be highly beneficial in terms of giving practical experience with working on concurrency and real-time data processing. I learned how concurrency models, threads vs. processes, impact system behavior in dealing with continuous data streams. It also increased my understanding of system-level programming, synchronization, and the challenges of timing between data acquisition and processing. I also enhanced my proficiency in communicating with Python through C and modifying scripts to graph time data. The project also supported the importance of selecting the right concurrency approach based on system needs, particularly in situations involving high data rates or real-time requirements.

Comparing the results to Assignment 4, where there was so much data loss due to sequential bottlenecks, the improvements were apparent. All three new scenarios had provisions for handling concurrent tasks, resulting in better capture and processing of the simulated data. The plots generated in Assignment 5 were greatly better, especially in Scenario 3, where the time intervals were much more equalized and close-packed. This indicates that multithreading effectively reduced delays and allowed for near-real-time processing. Scenario 4 was equally good but slightly less consistent in terms of time differences, while Scenario 2 performed the worst because of its subprocess synchronization reliance. Mostly, concurrency models tackled by this project offered concrete solutions to the data loss issues experienced before, highlighting the significance of concurrent programming towards system dependability.