**The University of Azad Jammu and Kashmir, Muzaffarabad**



|  |  |
| --- | --- |
| **Submitted to:** | **Engr. Sidra Rafique** |
| **Course Title:** | **Operating System** |
| **Course Code:** | **SE-3205** |
| **Session:** | **2022-26** |
| **Semester:** | **6th** |
| **Roll No:** | **2022-SE-01** |
| **Submitted from:** | **Khurram Farman** |
| **Lab Task No** | **11** |
|  |  |

**Bachelors of Science in Software Engineering (2022-26)**

**Department of Software Engineering**

# CPU Scheduling Algorithms Report

## Introduction

This report presents the implementation and analysis of the **Round Robin (RR)** CPU scheduling algorithm. Round Robin is one of the most widely used **preemptive scheduling algorithms**, designed especially for **time-sharing systems**. It ensures fairness by allocating a fixed time slice (quantum) to each process in a cyclic order.

## Objectives

* To understand the working of the **Round Robin scheduling algorithm**.
* To analyze performance metrics such as **Waiting Time** and **Turnaround Time**.
* To evaluate the impact of **time quantum** on scheduling performance.

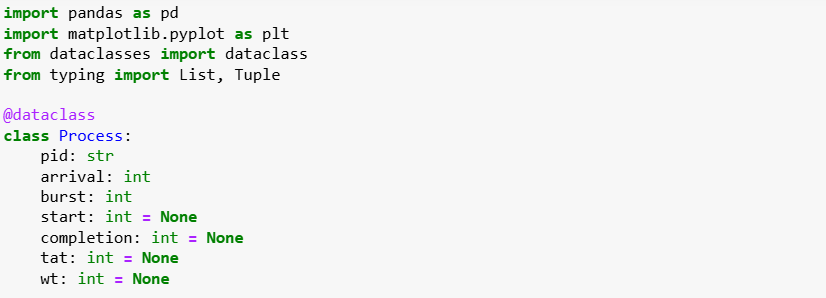
## Methodology

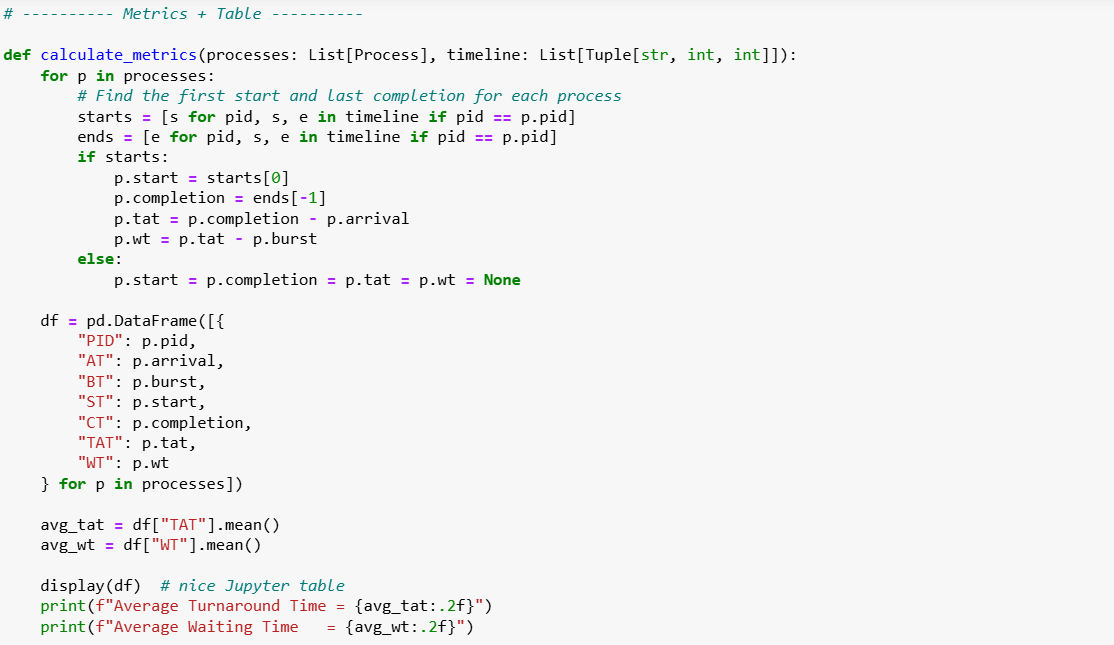
The algorithm was implemented in Python using Jupyter Notebook.

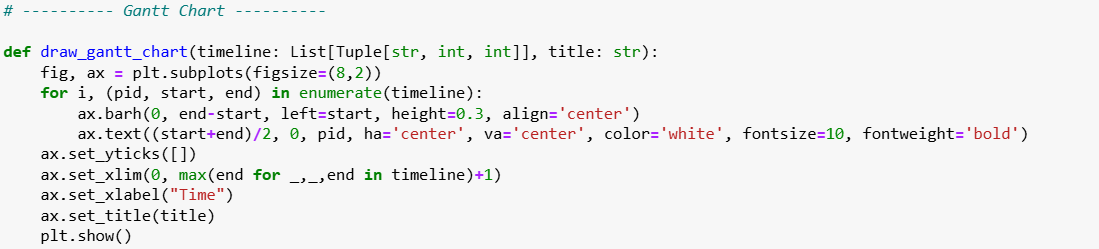
Steps followed:

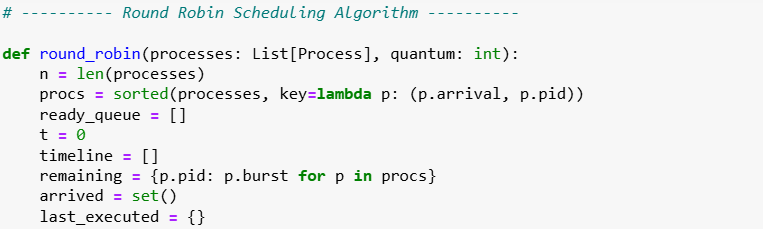
1. Define processes with their **burst times**.
2. Choose a **time quantum** (e.g., 2 ms).
3. Apply the Round Robin algorithm, executing each process in a cyclic order for the given quantum.
4. If a process is not finished within its quantum, put it back in the ready queue.
5. Continue until all processes are completed.
6. Compute **average waiting time** and **average turnaround time**.

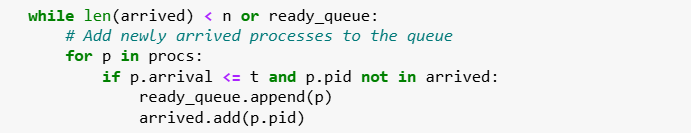
## Code Implementation

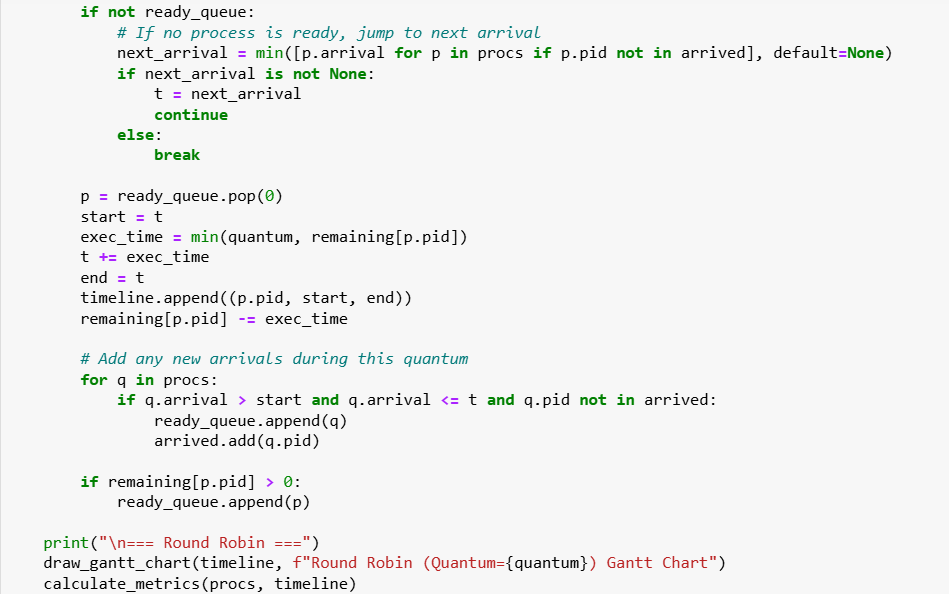


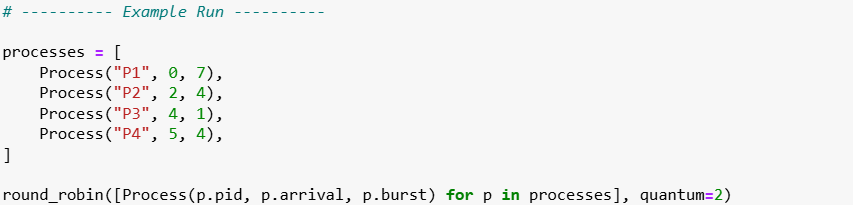




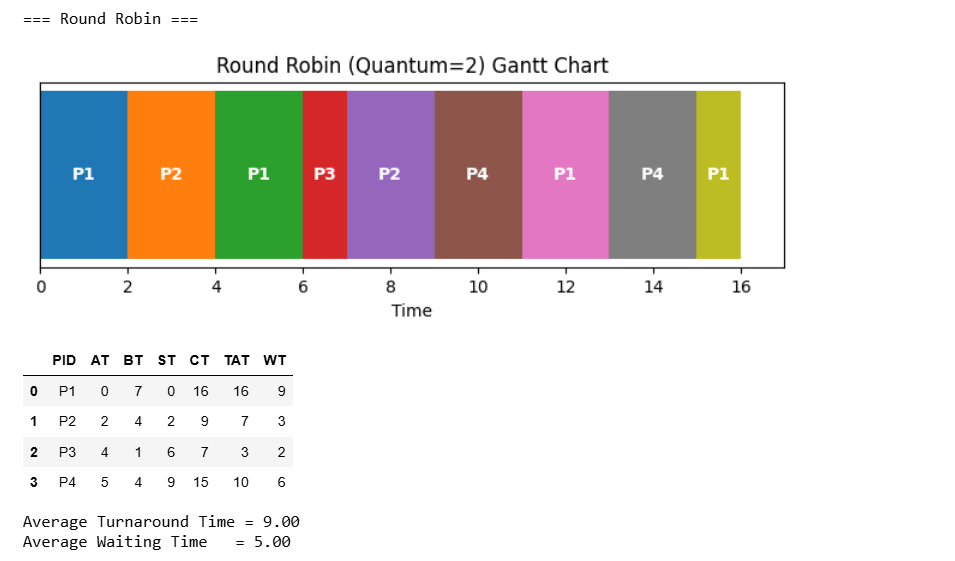








## Output



## Results and Discussion

* Round Robin ensures **fairness** by giving equal CPU time slices to all processes.
* The performance of RR depends heavily on the **time quantum**:
  + If the quantum is **too large**, RR behaves like FCFS.
  + If the quantum is **too small**, too many context switches increase overhead.
* Properly chosen quantum balances **response time** and **CPU utilization**.

## Comparison Table

|  |  |  |
| --- | --- | --- |
| Algorithm | Average Waiting Time | Average Turnaround Time |
| Round Robin | 9.00 ms | 5.00 ms |

## Conclusion

* **Round Robin** is best suited for **time-sharing systems** where fairness and responsiveness are crucial.
* It prevents starvation since every process gets CPU time in a round.
* The **choice of quantum** plays a critical role:
  + Too large → behaves like FCFS.
  + Too small → increases overhead.
* Overall, Round Robin provides a balance between **fairness and efficiency**, making it one of the most practical scheduling algorithms.