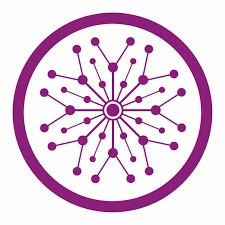
**Operating Systems Lab – Project**



**The Superior University**

Task Schedular Management System

#### 📌 **Project Title**

**Simulation of Round Robin Scheduling in Python**

#### 👥 **Group Members**

* Muzammal Toor (Roll No. 108)
* Hafiz Muhammad (Roll No.140)
* Hafiz Abubakar Ijaz(Roll No.063)
* Hammad Ahmad(Roll No 065)

#### 📂 **GitHub Repository**

**Link:** <https://github.com/Muzamal5858/Task-Schedular-Managment-Systems>

**Repository Includes:**

* ✔ Python .py file (Round Robin code)
* ✔ README.md (add if not already created)
* ✔ This documentation file
* ✔ Output screenshots (add 2–3 screenshots from your terminal)

#### 🔧 **Scheduling Algorithm Implemented**

**✅** Round Robin

#### **📄 Project Description**

This project simulates the **Round Robin CPU scheduling algorithm** using Python.  
It helps visualize how the operating system schedules multiple tasks fairly by assigning them equal time slices.

**Inputs:**

* Task name
* Arrival time
* Burst time
* Time quantum

**Outputs:**

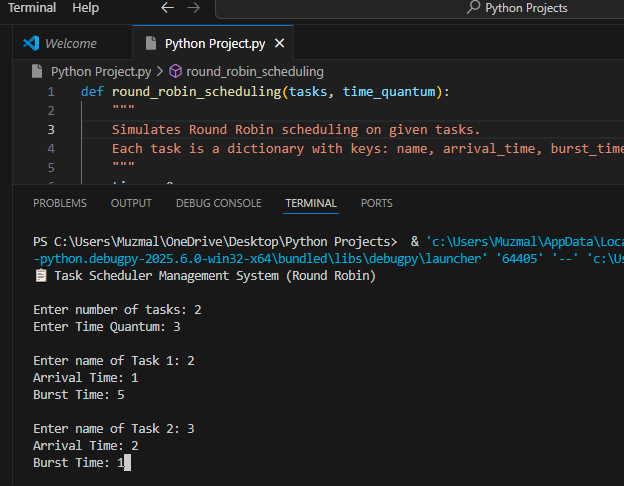
* Task execution order with timestamps
* Waiting time for each task
* Turnaround time for each task
* Average waiting and turnaround time

**Implementation Summary:**

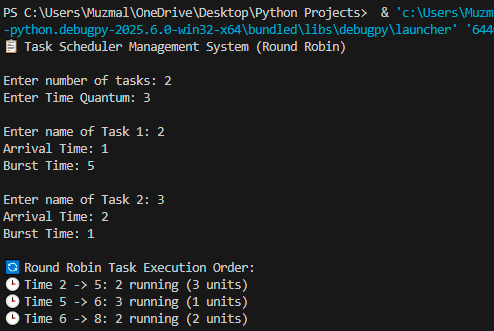
* The program uses dictionaries to manage task attributes.
* Tasks are added to a queue and executed in a round-robin cycle.
* It checks arrival time before execution and rotates unfinished tasks back to the queue.
* Waiting time and turnaround time are calculated and printed.

#### 📸 **Output Screenshots**

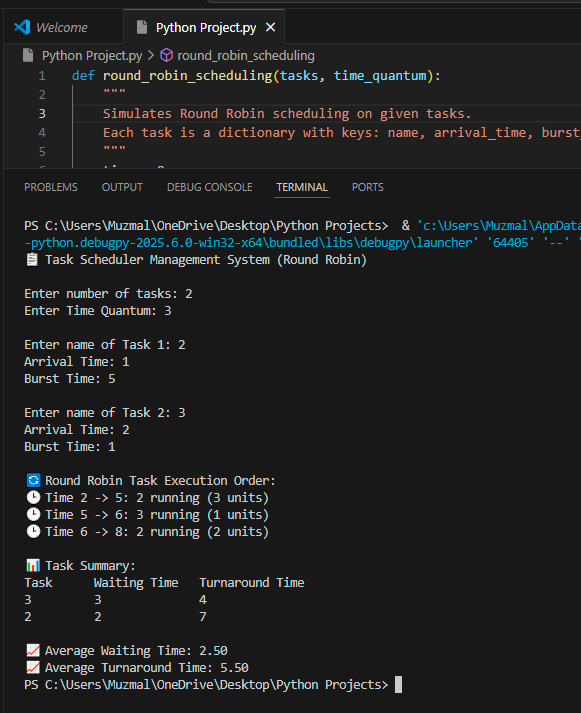
* Input tasks entered



* Task execution order printout



* Final table of waiting and turnaround times



#### **Code Structure & Explanation**

**Functions Used:**

* round\_robin\_scheduling() – Contains the main scheduling logic.
* main() – Collects user input, calls the scheduler, and prints results.

**Core Logic:**

* Uses a queue to simulate task rotation.
* Tracks remaining burst times using a dictionary.
* Calculates turnaround time = finish time - arrival time.
* Calculates waiting time = turnaround time - burst time.

**No external libraries** are used – it's simple and clean Python code with dictionaries and lists.

#### 📊 **Performance Metrics**

| **Metric** | **Value** |
| --- | --- |
| Average Waiting Time | Auto-calculated (shown in output) |
| Average Turnaround Time | Auto-calculated (shown in output) |
| Time Quantum | User-defined input |

#### 🚧 **Challenges Faced**

* 🐛 Adjusting for tasks arriving at different times – resolved by checking arrival\_time <= current\_time.
* 🔁 Correctly managing queue rotation and remaining burst time – solved by reappending tasks until complete.
* 📈 Making output clean and readable – used formatted print statements and spacing.