Operating Systems Lab
Fall 2024-25(L59+60)
Student Name:- Parth Suri
Registration No:- 22BDS0116
Class No. :- VL2024250102445
Faculty Name:- Rahul Srivastava

ROUND ROBIN ALGORITHM

Step 1: Define the Process Class

First, define a Process class to store the details of each process, including the arrival time.

Step 2: Create the Round Robin Scheduler Function

Update the Round Robin scheduling function to handle arrival times.

Step 3: Calculate Average Waiting and Turnaround Times

Step 4: Main Function to Execute the Scheduler

Update the main function to take user inputs for arrival time and burst time.

```
print(f"{process.pid}\t\t{process.arrival_time}\t\t{process.burst_time}\t\
t{process.waiting_time}\t\t{
process.turnaround_time}")
```

PREEMPTIVE PRIORITY CPU SCHEDULING ALGORITHM

Step 1: Define the Process Class

First, define a Process class to store the details of each process, including priority.

Step 2: Create the Preemptive Priority Scheduling Function

Now, create a function to perform the Preemptive Priority scheduling.

Step 3: Calculate Average Waiting and Turnaround Times

Create a function to calculate the average waiting time and turnaround time.

Step 4: Main Function to Execute the Scheduler

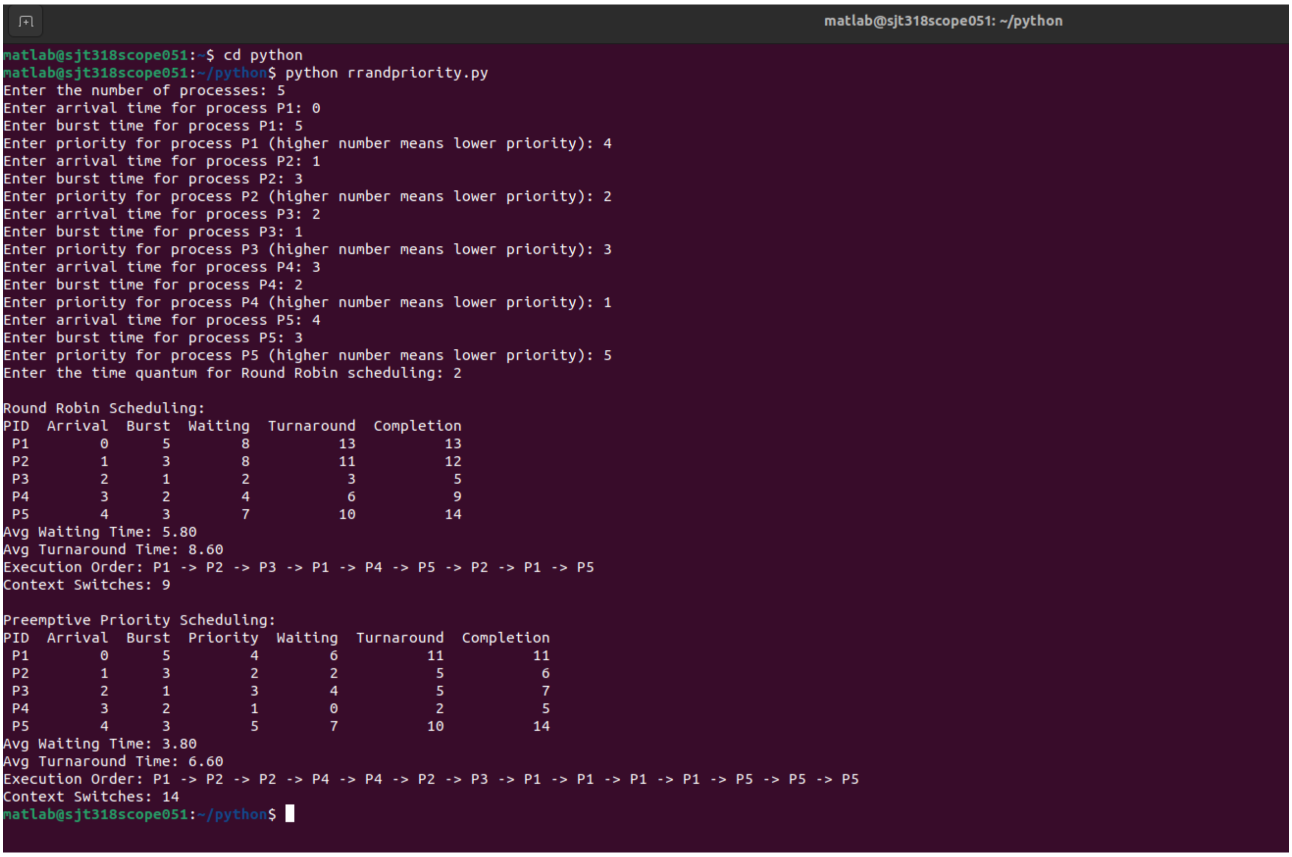
Finally, create a main function to execute the scheduling algorithm.

Code:-

Description:- It gives a combined output of both Round Robin and Priority Scheduling Algorithm

```
class Process:
    def __init__(self, pid, arrival_time, burst_time, priority=None):
        self.pid = pid
        self.arrival_time = arrival_time
        self.burst_time = burst_time
        self.remaining_time = burst_time
        self.priority = priority
        self.waiting_time = 0
        self.turnaround_time = 0
        self.completion_time = 0
def round_robin(processes, time_quantum):
    time = 0
    context_switches = 0
    queue = []
    execution_order = []
    processes = processes[:] # Make a copy of the list
    while processes or queue:
        # Add newly arrived processes to the queue
        while processes and processes[0].arrival_time <= time:
            queue.append(processes.pop(0))
        if queue:
            current_process = queue.pop(0)
            execution_order.append(current_process.pid)
            context_switches += 1
            if current_process.remaining_time > time_quantum:
                time += time_quantum
                current_process.remaining_time -= time_quantum
                while processes and processes[0].arrival_time <= time:
                    queue.append(processes.pop(0))
                queue.append(current_process)
            else:
                time += current_process.remaining_time
                current_process.waiting_time = time - current_process.burst_time -
current_process.arrival_time
                current_process.turnaround_time = time - current_process.arrival_time
                current_process.completion_time = time
                current_process.remaining_time = 0
    return execution_order, context_switches
def preemptive_priority(processes):
    time = 0
    context_switches = 0
    execution_order = []
    queue = []
    processes = processes[:] # Make a copy of the list
    while processes or queue:
        while processes and processes[0].arrival_time <= time:
            queue.append(processes.pop(0))
        queue.sort(key=lambda x: (x.priority, x.arrival_time))
        if queue:
            current_process = queue.pop(0)
            execution_order.append(current_process.pid)
            context_switches += 1
            if current_process.remaining_time > 1:
                time += 1
                current_process.remaining_time -= 1
                while processes and processes[0].arrival_time <= time:
                    queue.append(processes.pop(0))
                queue.append(current_process)
            else:
                time += current_process.remaining_time
                current_process.waiting_time = time - current_process.burst_time - 
current_process.arrival_time
                current_process.turnaround_time = time - current_process.arrival_time
                current_process.completion_time = time
                current_process.remaining_time = 0
    return execution_order, context_switches
def calculate_average_times(processes):
    total_waiting_time = sum(p.waiting_time for p in processes)
    total_turnaround_time = sum(p.turnaround_time for p in processes)
    n = len(processes)
    return total_waiting_time / n, total_turnaround_time / n
def main():
    import pandas as pd
    # User input
    num_processes = int(input("Enter the number of processes: "))
    arrival_times = []
    burst_times = []
    priorities = []
    for i in range(num_processes):
        arrival_time = int(input(f"Enter arrival time for process P{i+1}: "))
        burst_time = int(input(f"Enter burst time for process P{i+1}: "))
        priority = int(input(f"Enter priority for process P{i+1} (higher number means lower priority): 
"))
        arrival_times.append(arrival_time)
        burst_times.append(burst_time)
        priorities.append(priority)
    time_quantum = int(input("Enter the time quantum for Round Robin scheduling: "))
    # Create processes for Round Robin
    processes_rr = [Process(f'P{i+1}', arrival_times[i], burst_times[i]) for i in 
range(len(arrival_times))]
    processes_rr.sort(key=lambda x: x.arrival_time)
    # Create processes for Preemptive Priority
    processes_pp = [Process(f'P{i+1}', arrival_times[i], burst_times[i], priorities[i]) for i in 
range(len(arrival_times))]
    processes_pp.sort(key=lambda x: x.arrival_time)
    # Round Robin Scheduling
    rr_execution_order, rr_context_switches = round_robin(processes_rr[:], time_quantum) # Pass a 
copy
    rr_avg_waiting_time, rr_avg_turnaround_time = calculate_average_times(processes_rr)
    # Preemptive Priority Scheduling
    pp_execution_order, pp_context_switches = preemptive_priority(processes_pp[:]) # Pass a copy
    pp_avg_waiting_time, pp_avg_turnaround_time = calculate_average_times(processes_pp)
    # Display Results
    print("\nRound Robin Scheduling:")
    rr_data = {
        "PID": [p.pid for p in processes_rr],
        "Arrival": [p.arrival_time for p in processes_rr],
        "Burst": [p.burst_time for p in processes_rr],
        "Waiting": [p.waiting_time for p in processes_rr],
        "Turnaround": [p.turnaround_time for p in processes_rr],
        "Completion": [p.completion_time for p in processes_rr]
    }
    df_rr = pd.DataFrame(rr_data)
    print(df_rr.to_string(index=False))
    print(f"Avg Waiting Time: {rr_avg_waiting_time:.2f}")
    print(f"Avg Turnaround Time: {rr_avg_turnaround_time:.2f}")
    print(f"Execution Order: {' -> '.join(rr_execution_order)}")
    print(f"Context Switches: {rr_context_switches}")
    print("\nPreemptive Priority Scheduling:")
    pp_data = {
        "PID": [p.pid for p in processes_pp],
        "Arrival": [p.arrival_time for p in processes_pp],
        "Burst": [p.burst_time for p in processes_pp],
        "Priority": [p.priority for p in processes_pp],
        "Waiting": [p.waiting_time for p in processes_pp],
        "Turnaround": [p.turnaround_time for p in processes_pp],
        "Completion": [p.completion_time for p in processes_pp]
    }
    df_pp = pd.DataFrame(pp_data)
    print(df_pp.to_string(index=False))
    print(f"Avg Waiting Time: {pp_avg_waiting_time:.2f}")
    print(f"Avg Turnaround Time: {pp_avg_turnaround_time:.2f}")
    print(f"Execution Order: {' -> '.join(pp_execution_order)}")
    print(f"Context Switches: {pp_context_switches}")
if __name__ == "__main__":
    main()
```

Test Case 1:-

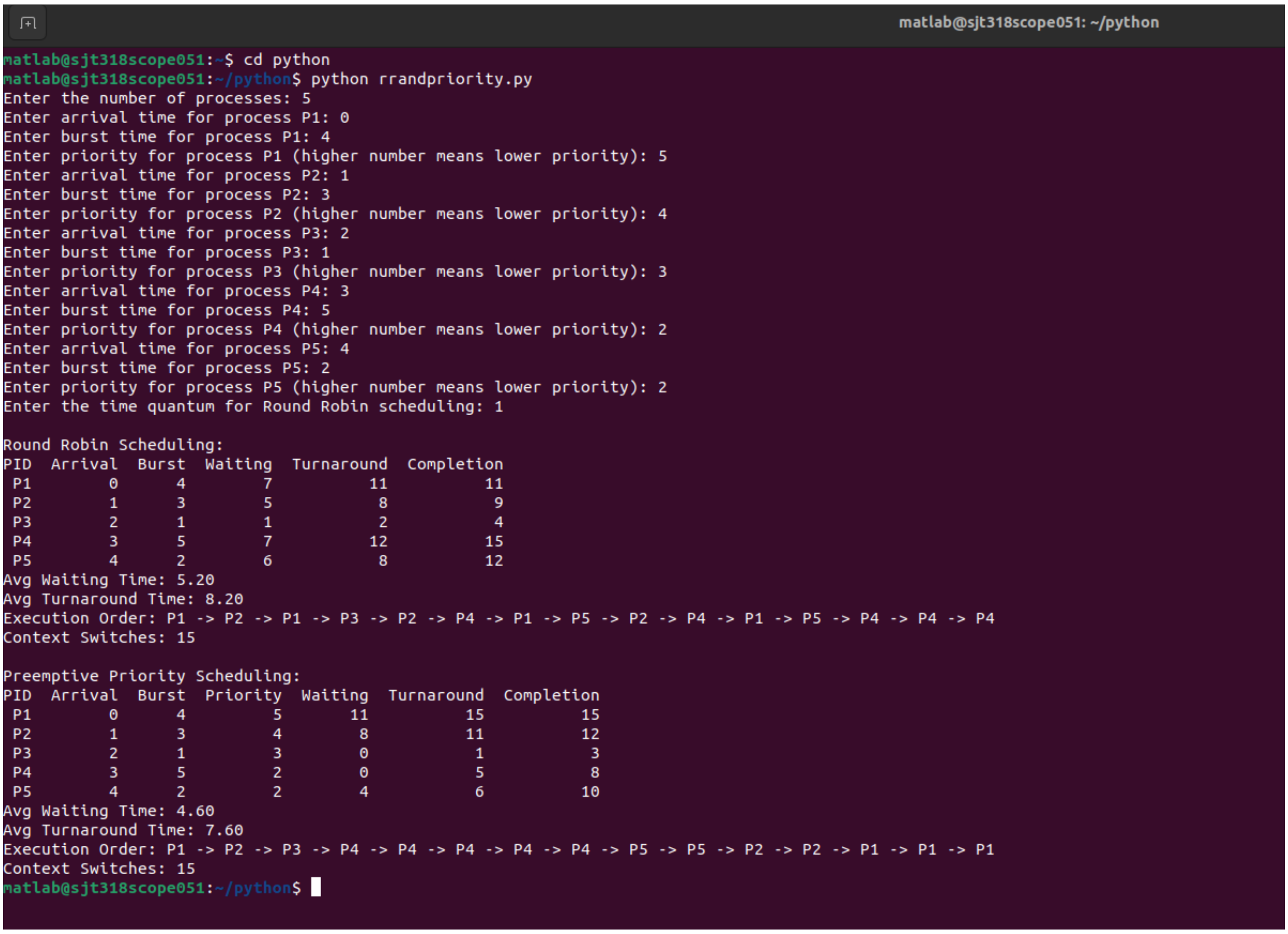


```
matlab@sjt318scope051:~$ cd python
matlab@sjt318scope051:~/python$ python rrandpriority.py
Enter the number of processes: 5
Enter arrival time for process P1: 0
Enter burst time for process P1: 5
Enter priority for process P1 (higher number means lower priority): 4
Enter arrival time for process P2: 1
Enter burst time for process P2: 3
Enter priority for process P2 (higher number means lower priority): 2
Enter arrival time for process P3: 2
Enter burst time for process P3: 1
Enter priority for process P3 (higher number means lower priority): 3
Enter arrival time for process P4: 3
Enter burst time for process P4: 2
Enter priority for process P4 (higher number means lower priority): 1
Enter arrival time for process P5: 4
Enter burst time for process P5: 3
Enter priority for process P5 (higher number means lower priority): 5
Enter the time quantum for Round Robin scheduling: 2

Round Robin Scheduling:
PID  Arrival  Burst  Waiting  Turnaround  Completion
 P1        0      5        8          13          13
 P2        1      3        8          11          12
 P3        2      1        2           3           5
 P4        3      2        4           6           9
 P5        4      3        7          10          14
Avg Waiting Time: 5.80
Avg Turnaround Time: 8.60
Execution Order: P1 -> P2 -> P3 -> P1 -> P4 -> P5 -> P2 -> P1 -> P5
Context Switches: 9

Preemptive Priority Scheduling:
PID  Arrival  Burst  Priority  Waiting  Turnaround  Completion
 P1        0      5         4        6          11          11
 P2        1      3         2        2           5           6
 P3        2      1         3        4           5           7
 P4        3      2         1        0           2           5
 P5        4      3         5        7          10          14
Avg Waiting Time: 3.80
Avg Turnaround Time: 6.60
Execution Order: P1 -> P2 -> P2 -> P4 -> P4 -> P2 -> P3 -> P1 -> P1 -> P1 -> P1 -> P5 -> P5 -> P5
Context Switches: 14
matlab@sjt318scope051:~/python$ 
```

Test Case 2:-



```
matlab@sjt318scope051:~$ cd python
matlab@sjt318scope051:~/python$ python rrandpriority.py
Enter the number of processes: 5
Enter arrival time for process P1: 0
Enter burst time for process P1: 4
Enter priority for process P1 (higher number means lower priority): 5
Enter arrival time for process P2: 1
Enter burst time for process P2: 3
Enter priority for process P2 (higher number means lower priority): 4
Enter arrival time for process P3: 2
Enter burst time for process P3: 1
Enter priority for process P3 (higher number means lower priority): 3
Enter arrival time for process P4: 3
Enter burst time for process P4: 5
Enter priority for process P4 (higher number means lower priority): 2
Enter arrival time for process P5: 4
Enter burst time for process P5: 2
Enter priority for process P5 (higher number means lower priority): 2
Enter the time quantum for Round Robin scheduling: 1

Round Robin Scheduling:
PID  Arrival  Burst  Waiting  Turnaround  Completion
 P1        0      4        7          11          11
 P2        1      3        5           8           9
 P3        2      1        1           2           4
 P4        3      5        7          12          15
 P5        4      2        6           8          12
Avg Waiting Time: 5.20
Avg Turnaround Time: 8.20
Execution Order: P1 -> P2 -> P1 -> P3 -> P2 -> P4 -> P1 -> P5 -> P2 -> P4 -> P1 -> P5 -> P4 -> P4 -> P4
Context Switches: 15

Preemptive Priority Scheduling:
PID  Arrival  Burst  Priority  Waiting  Turnaround  Completion
 P1        0      4         5       11          15          15
 P2        1      3         4        8          11          12
 P3        2      1         3        0           1           3
 P4        3      5         2        0           5           8
 P5        4      2         2        4           6          10
Avg Waiting Time: 4.60
Avg Turnaround Time: 7.60
Execution Order: P1 -> P2 -> P3 -> P4 -> P4 -> P4 -> P4 -> P4 -> P5 -> P5 -> P2 -> P2 -> P1 -> P1 -> P1
Context Switches: 15
matlab@sjt318scope051:~/python$ 
```