UIT2512---Operating Systems Practices Lab

2) Implementation of Round Robin CPU Scheduling Algorithm in Python

Name: Vasundhara.B

Roll no: 3122 21 5002 119

AIM:

To implement the Round Robin CPU Scheduling Algorithm in Python, illustrating the allocation of CPU time to processes in a circular fashion with a fixed time quantum.

Description:

This algorithm employs a fixed time quantum, allowing each process to execute for a specified time slice before moving on to the next process. This implementation in Python will simulate the round-robin scheduling of processes, showcasing how it ensures fairness among processes by preventing any single process from monopolizing the CPU for extended periods.

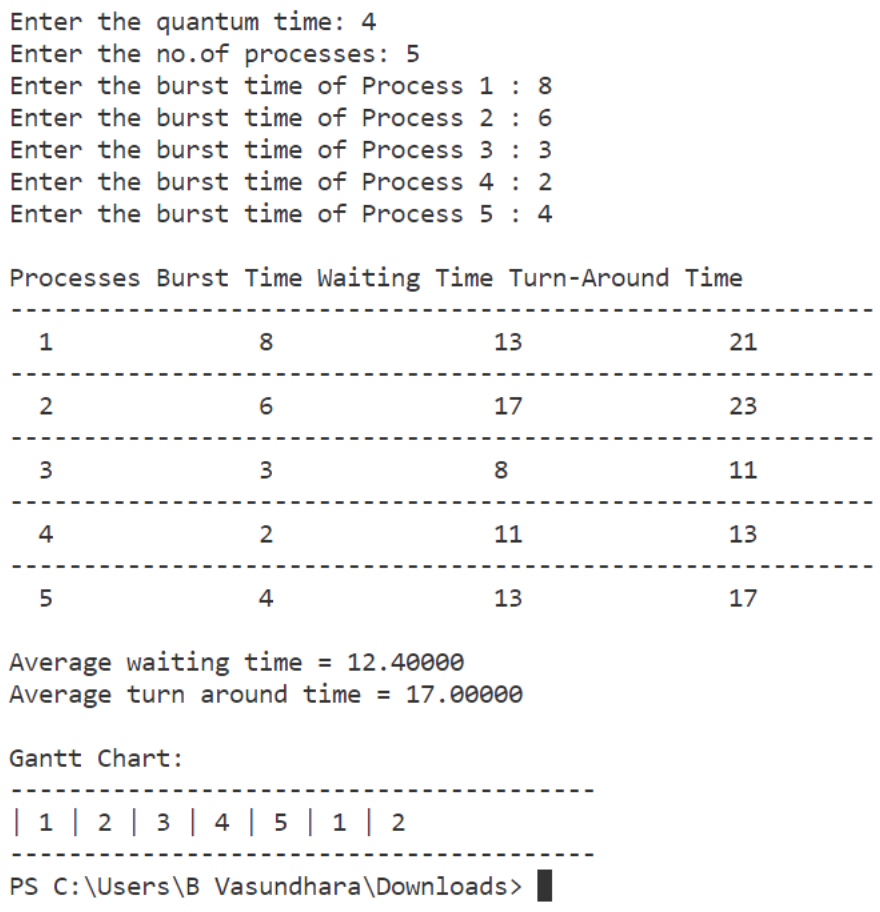
CODE:

```
def findWaitingTime(processes, n, bt, wt, quantum):
    rem_bt = [0] * n
    for i in range(n):
        rem_bt[i] = bt[i]

    t = 0  # Current time
    gantt_chart = []  # List to store the Gantt chart

    while True:
        done = True
        for i in range(n):
            if rem_bt[i] > 0:
                done = False  # There is a pending process
                if rem_bt[i] > quantum:
                    t += quantum
                    rem_bt[i] -= quantum
                else:
                    t += rem_bt[i]
                    wt[i] = t - bt[i]
                    rem_bt[i] = 0
```

```python
                # Add the process to the Gantt chart
                gantt_chart.append(processes[i])

        if done:
            break
    return gantt_chart

def findTurnAroundTime(processes, n, bt, wt, tat):
    for i in range(n):
        tat[i] = bt[i] + wt[i]


def findavgTime(processes, n, bt, quantum):
    wt = [0] * n
    tat = [0] * n

    gantt_chart = findWaitingTime(processes, n, bt, wt, quantum)

    findTurnAroundTime(processes, n, bt, wt, tat)

    print()
    print("Processes Burst Time Waiting",
          "Time Turn-Around Time")
    total_wt = 0
    total_tat = 0
    for i in range(n):
        total_wt = total_wt + wt[i]
        total_tat = total_tat + tat[i]
        print("-----------------------------------------------------------")
        print(" ", processes[i], "\t\t", bt[i],
              "\t\t", wt[i], "\t\t", tat[i])

    print("\nAverage waiting time = %.5f " % (total_wt / n))
    print("Average turn around time = %.5f " % (total_tat / n))

    # Print Gantt Chart
    print("\nGantt Chart:")
    print("-" * 40)
    prev_process = gantt_chart[0]
    for process in gantt_chart:

        print("|", process, end=" ")
    print()
    print("-" * 40)

# Driver code
if __name__ == "__main__":
    # Process id's
    proc = []
    burst_time=[]
```

```python
quantum = int(input("Enter the quantum time: "))
n = int(input("Enter the no.of processes: "))
for i in range(n):
    proc.append(i+1)
    bt=int(input(f"Enter the burst time of Process {i+1} : "))
    burst_time.append(bt)

# Time quantum

findavgTime(proc, n, burst_time, quantum)
```

OUTPUT:



```
Enter the quantum time: 4
Enter the no.of processes: 5
Enter the burst time of Process 1 : 8
Enter the burst time of Process 2 : 6
Enter the burst time of Process 3 : 3
Enter the burst time of Process 4 : 2
Enter the burst time of Process 5 : 4

Processes Burst Time Waiting Time Turn-Around Time
-----------------------------------------------------------
  1              8              13              21
-----------------------------------------------------------
  2              6              17              23
-----------------------------------------------------------
  3              3              8               11
-----------------------------------------------------------
  4              2              11              13
-----------------------------------------------------------
  5              4              13              17

Average waiting time = 12.40000
Average turn around time = 17.00000

Gantt Chart:
----------------------------------------
| 1 | 2 | 3 | 4 | 5 | 1 | 2
----------------------------------------
PS C:\Users\B Vasundhara\Downloads>
```