Name: Abhinav Rajesh	**Roll No:** 03
Exp. No: 7	**Date:** 24/02/2022

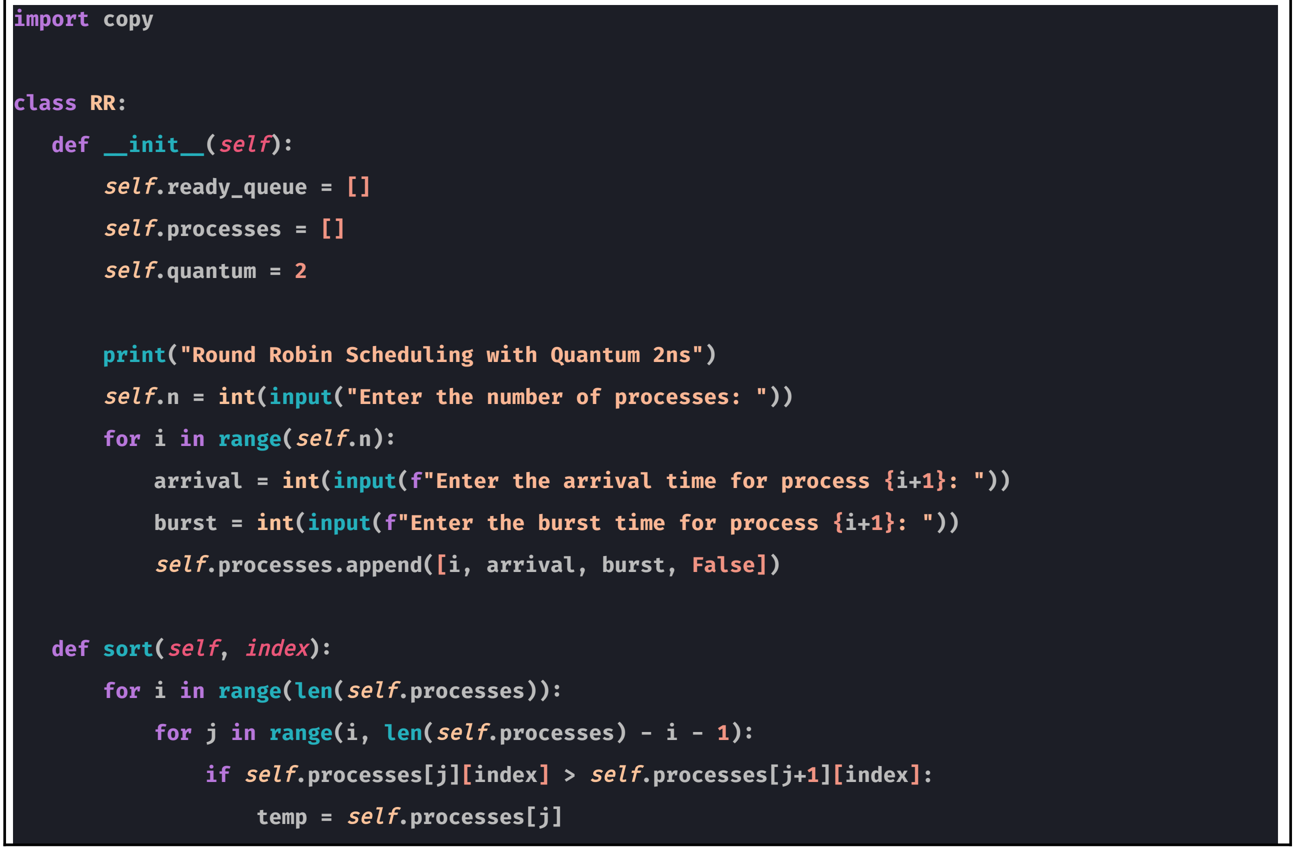
SJF

Aim: Write a program to implement Round Robin Scheduling with arrival time (quantum 2 ns).

Algorithm:

1. START
2. Define Class RR
    a. Initialize constructor and get input from the user about number of processes and the burst and arrival time for these processes
    b. Create function sort to sort the processes according to the parameter provided(arrival time, process number, etc.)
    c. Create a function roundRobinSchedule to perform function required for RR algorithm, like finding the processes ready queue, etc
    d. Create utility function setProcessArrival to set the value in ready queue when a process has arrived
    e. Create function displayTable to print the computed data in the required table format
3. Create main function
    a. Create an object “rr” of the class “RR”
    b. Call the methods sort, roundRobinScheduling and displayTable in this order
4. STOP

Program

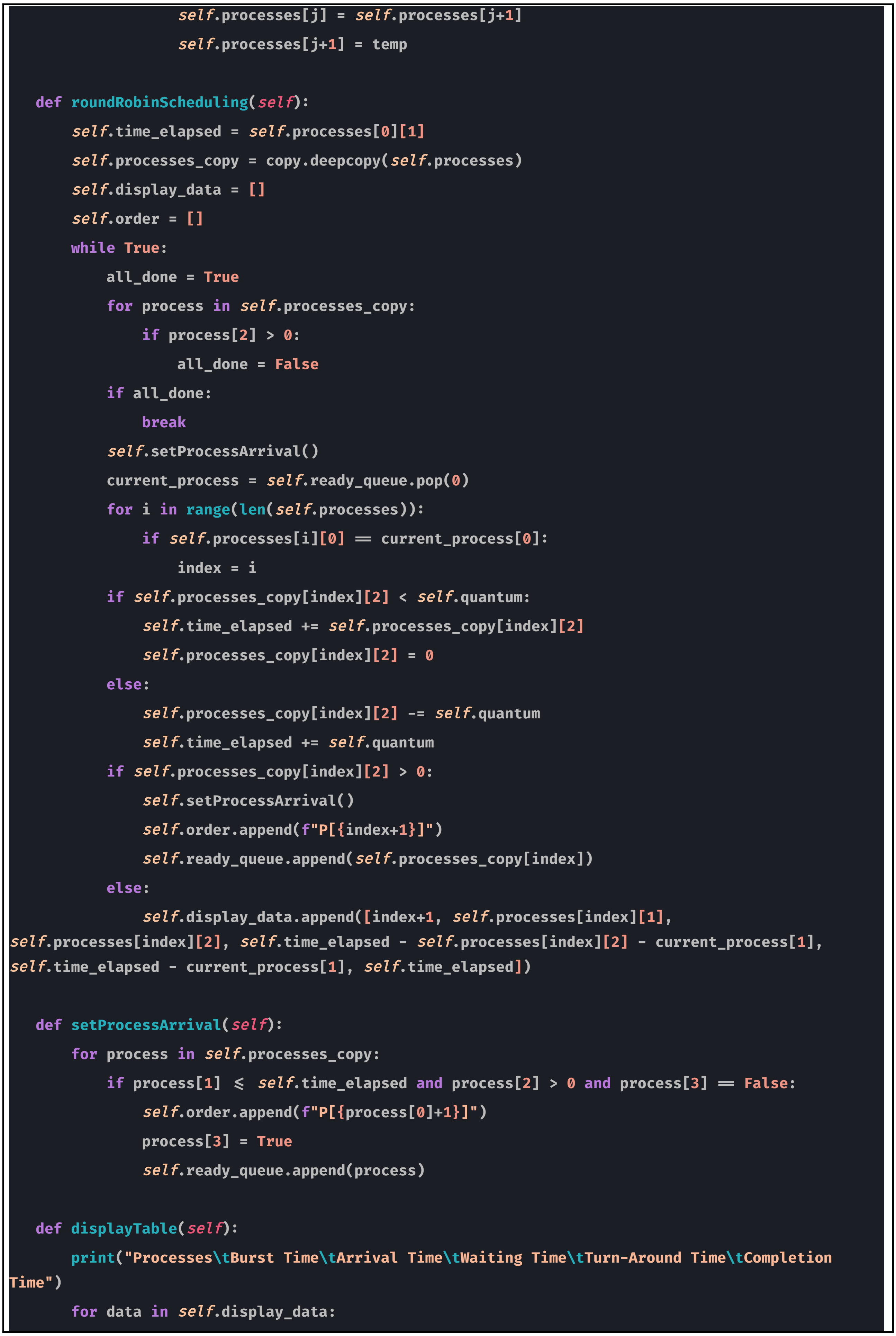


```
import copy

class RR:
   def __init__(self):
       self.ready_queue = []
       self.processes = []
       self.quantum = 2

       print("Round Robin Scheduling with Quantum 2ns")
       self.n = int(input("Enter the number of processes: "))
       for i in range(self.n):
           arrival = int(input(f"Enter the arrival time for process {i+1}: "))
           burst = int(input(f"Enter the burst time for process {i+1}: "))
           self.processes.append([i, arrival, burst, False])

   def sort(self, index):
       for i in range(len(self.processes)):
           for j in range(i, len(self.processes) - i - 1):
               if self.processes[j][index] > self.processes[j+1][index]:
                   temp = self.processes[j]
```

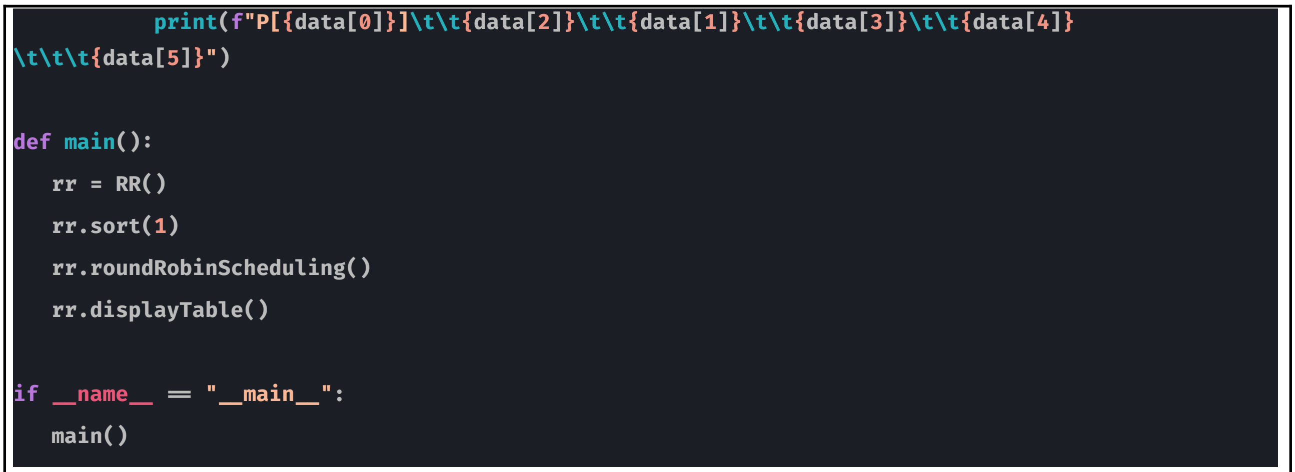


```
                    self.processes[j] = self.processes[j+1]
                    self.processes[j+1] = temp

    def roundRobinScheduling(self):
        self.time_elapsed = self.processes[0][1]
        self.processes_copy = copy.deepcopy(self.processes)
        self.display_data = []
        self.order = []
        while True:
            all_done = True
            for process in self.processes_copy:
                if process[2] > 0:
                    all_done = False
            if all_done:
                break
            self.setProcessArrival()
            current_process = self.ready_queue.pop(0)
            for i in range(len(self.processes)):
                if self.processes[i][0] == current_process[0]:
                    index = i
            if self.processes_copy[index][2] < self.quantum:
                self.time_elapsed += self.processes_copy[index][2]
                self.processes_copy[index][2] = 0
            else:
                self.processes_copy[index][2] -= self.quantum
                self.time_elapsed += self.quantum
            if self.processes_copy[index][2] > 0:
                self.setProcessArrival()
                self.order.append(f"P[{index+1}]")
                self.ready_queue.append(self.processes_copy[index])
            else:
                self.display_data.append([index+1, self.processes[index][1],
self.processes[index][2], self.time_elapsed - self.processes[index][2] - current_process[1],
self.time_elapsed - current_process[1], self.time_elapsed])

    def setProcessArrival(self):
        for process in self.processes_copy:
            if process[1] <= self.time_elapsed and process[2] > 0 and process[3] == False:
                self.order.append(f"P[{process[0]+1}]")
                process[3] = True
                self.ready_queue.append(process)

    def displayTable(self):
        print("Processes\tBurst Time\tArrival Time\tWaiting Time\tTurn-Around Time\tCompletion
Time")
        for data in self.display_data:
```

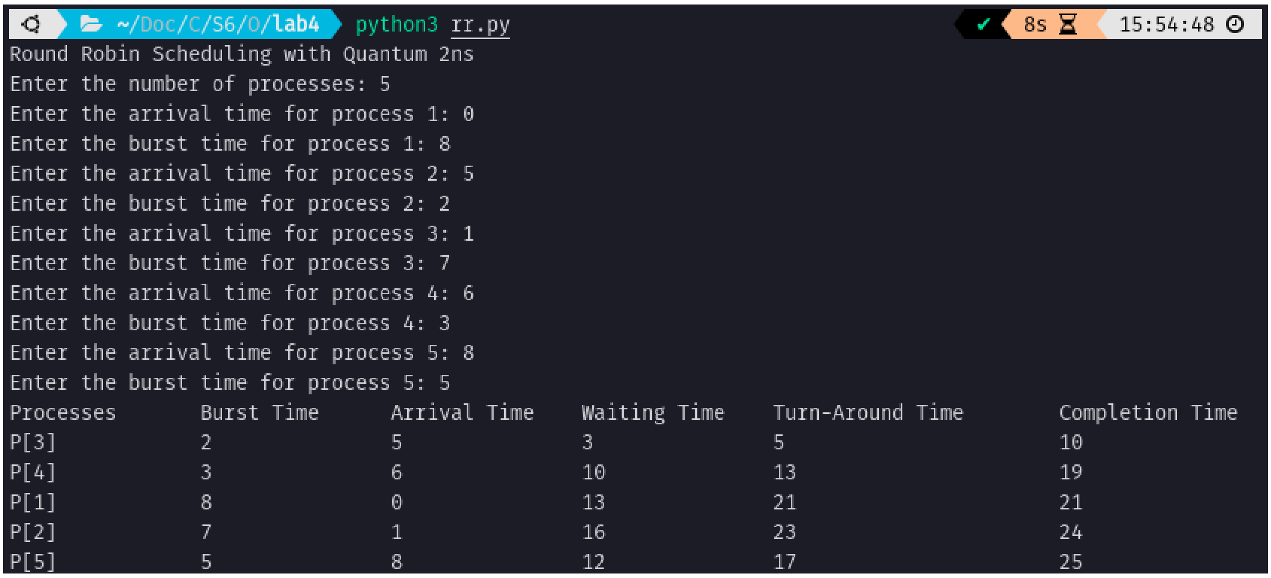


```
           print(f"P[{data[0]}]\t\t{data[2]}\t\t{data[1]}\t\t{data[3]}\t\t{data[4]}
\t\t\t{data[5]}")

def main():
   rr = RR()
   rr.sort(1)
   rr.roundRobinScheduling()
   rr.displayTable()

if __name__ == "__main__":
   main()
```

Output



```
~/Doc/C/S6/O/lab4  python3 rr.py                                        ✔  8s ⌛  15:54:48
Round Robin Scheduling with Quantum 2ns
Enter the number of processes: 5
Enter the arrival time for process 1: 0
Enter the burst time for process 1: 8
Enter the arrival time for process 2: 5
Enter the burst time for process 2: 2
Enter the arrival time for process 3: 1
Enter the burst time for process 3: 7
Enter the arrival time for process 4: 6
Enter the burst time for process 4: 3
Enter the arrival time for process 5: 8
Enter the burst time for process 5: 5
Processes       Burst Time      Arrival Time    Waiting Time    Turn-Around Time        Completion Time
P[3]            2               5               3               5                       10
P[4]            3               6               10              13                      19
P[1]            8               0               13              21                      21
P[2]            7               1               16              23                      24
P[5]            5               8               12              17                      25
```

Result:

Python Program to implement RR scheduling is compiled and executed successfully

Remarks: