**SVKM’S NMIMS**

**MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT& ENGINEERING**

**(Campus Name)**

Academic Year: 2020-2021

# **Practical 4 – Program to Demonstrate the Round Robin Algorithm**

|  |  |  |
| --- | --- | --- |
| Name: Jal Bafana | Roll\_No: K005 | SAP-ID:70102300054 |
| Btech-cys |  |  |

Dear all,

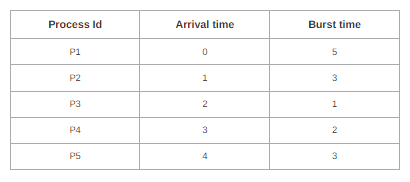
Kindly complete the following task with your name in output file also attach the C program with the file.

Find the Turnaround time and Average Turnaround time.

Find the Waiting time and Average Waiting time.

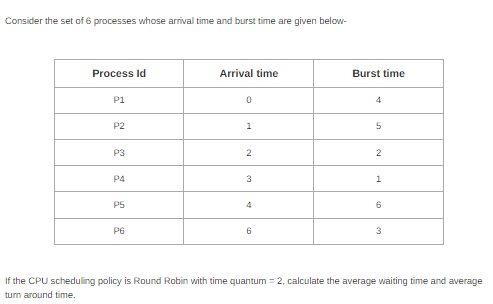
1. Completion Time: Time at which process completes its execution.
2. Turn Around Time: Time Difference between completion time and arrival time. Turn Around Time = Completion Time – Arrival Time
3. Waiting Time(W.T): Time Difference between turn around time and burst time.   
   Waiting Time = Turn Around Time – Burst Time

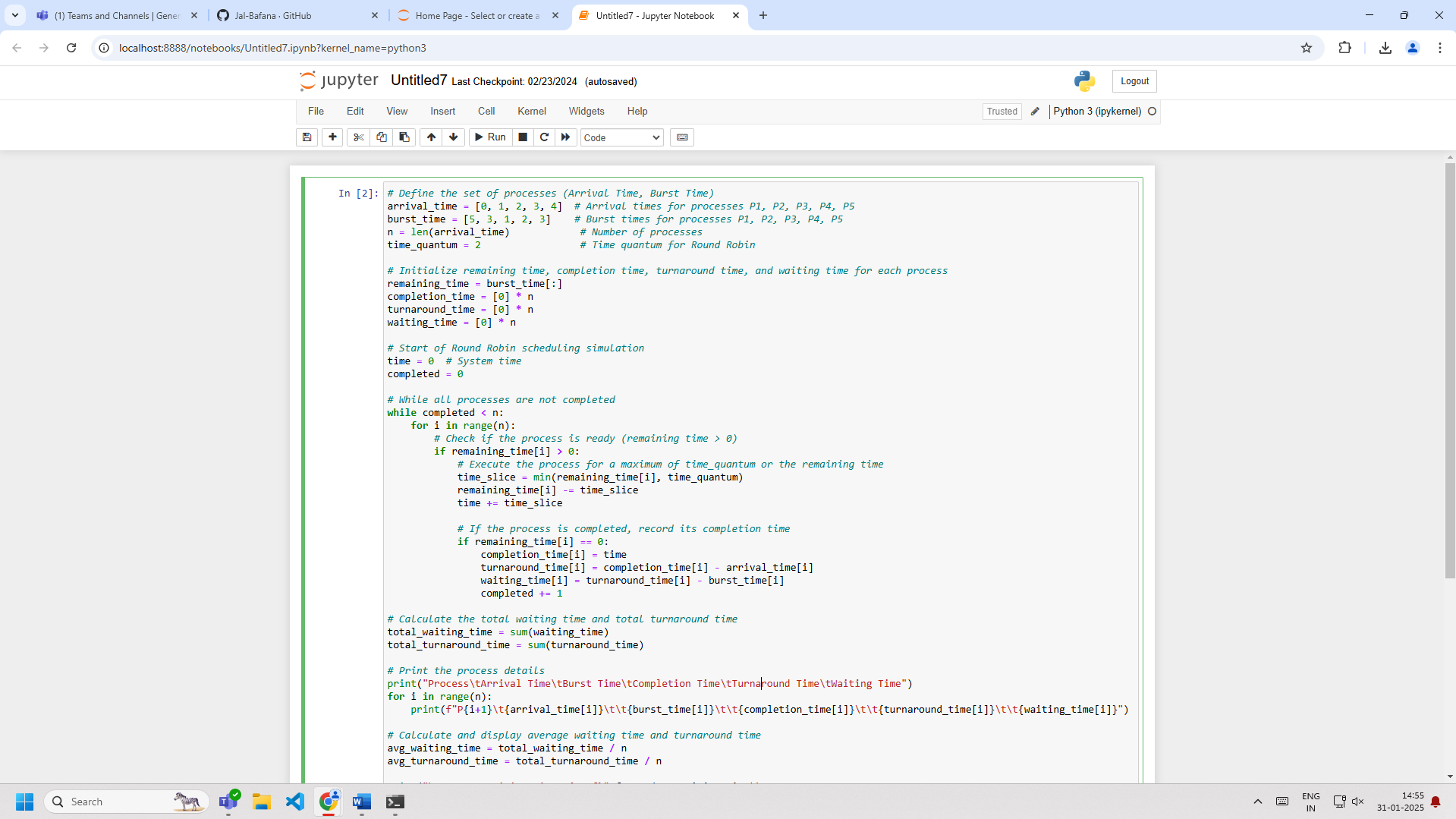
Problem – 1: Consider the set of 5 processes whose arrival time and burst time are given below-



If the CPU scheduling policy is Round Robin with time quantum = 2 unit, calculate the average waiting time and average turnaround time

Problem 2:





# Define the set of processes (Arrival Time, Burst Time)

arrival\_time = [0, 1, 2, 3, 4] # Arrival times for processes P1, P2, P3, P4, P5

burst\_time = [5, 3, 1, 2, 3] # Burst times for processes P1, P2, P3, P4, P5

n = len(arrival\_time) # Number of processes

time\_quantum = 2 # Time quantum for Round Robin

# Initialize remaining time, completion time, turnaround time, and waiting time for each process

remaining\_time = burst\_time[:]

completion\_time = [0] \* n

turnaround\_time = [0] \* n

waiting\_time = [0] \* n

# Start of Round Robin scheduling simulation

time = 0 # System time

completed = 0

# While all processes are not completed

while completed < n:

for i in range(n):

# Check if the process is ready (remaining time > 0)

if remaining\_time[i] > 0:

# Execute the process for a maximum of time\_quantum or the remaining time

time\_slice = min(remaining\_time[i], time\_quantum)

remaining\_time[i] -= time\_slice

time += time\_slice

# If the process is completed, record its completion time

if remaining\_time[i] == 0:

completion\_time[i] = time

turnaround\_time[i] = completion\_time[i] - arrival\_time[i]

waiting\_time[i] = turnaround\_time[i] - burst\_time[i]

completed += 1

# Calculate the total waiting time and total turnaround time

total\_waiting\_time = sum(waiting\_time)

total\_turnaround\_time = sum(turnaround\_time)

# Print the process details

print("Process\tArrival Time\tBurst Time\tCompletion Time\tTurnaround Time\tWaiting Time")

for i in range(n):

print(f"P{i+1}\t{arrival\_time[i]}\t\t{burst\_time[i]}\t\t{completion\_time[i]}\t\t{turnaround\_time[i]}\t\t{waiting\_time[i]}")

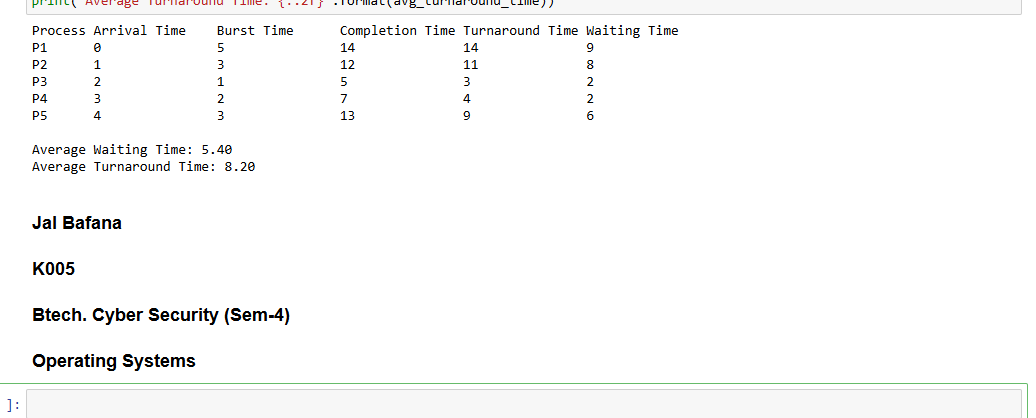
# Calculate and display average waiting time and turnaround time

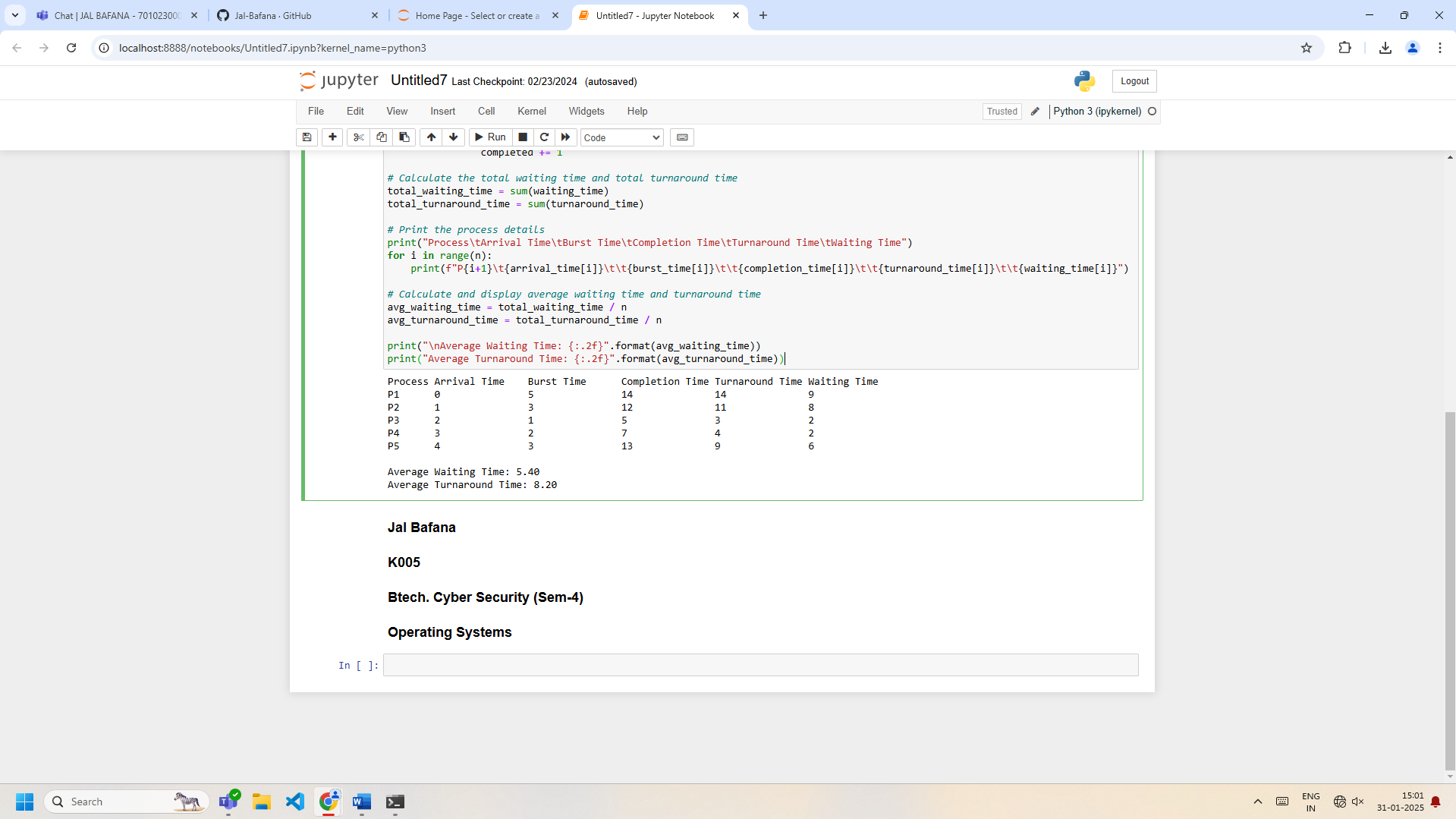
avg\_waiting\_time = total\_waiting\_time / n

avg\_turnaround\_time = total\_turnaround\_time / n

print("\nAverage Waiting Time: {:.2f}".format(avg\_waiting\_time))

print("Average Turnaround Time: {:.2f}".format(avg\_turnaround\_time))





## **Conclusion: -**

* The **Round Robin Scheduling Algorithm** ensures fairness by giving each process a fixed time slice (quantum) for execution.
* This method is very effective in time-sharing systems, where responsiveness and equal sharing of CPU time among processes are important.
* However, **Round Robin** can be inefficient with a large number of processes or very large time quantums, as some processes may need less CPU time but end up being scheduled multiple times unnecessarily.