FCFS :-

```
def findWaitingTime(processes, n, bt, wt):
    wt[0] = 0  # Waiting time for the first process is 0

    for i in range(1, n):
        wt[i] = bt[i - 1] + wt[i - 1]  # Calculate waiting time for each process
def findTurnAroundTime(processes, n, bt, wt, tat):
    for i in range(n):
        tat[i] = bt[i] + wt[i]  # Calculate turnaround time for each process
def findavgTime(processes, n, bt):
    wt = [0] * n  # Initialize waiting time list
    tat = [0] * n  # Initialize turnaround time list
    total_wt = 0
    total_tat = 0
    findWaitingTime(processes, n, bt, wt)
    findTurnAroundTime(processes, n, bt, wt, tat)

    print("Processes Burst time Waiting time Turn around time")
    for i in range(n):
        total_wt += wt[i]
        total_tat += tat[i]
        print(f" {i + 1}\t\t{bt[i]}\t {wt[i]}\t\t {tat[i]}")
    print("Average waiting time = " + str(total_wt / n))
    print("Average turn around time = " + str(total_tat / n))
if __name__ == "__main__":
    # Process IDs
    processes = [1, 2, 3]
    n = len(processes)
    # Burst time of all processes
    burst_time = [10, 5, 8]
    findavgTime(processes, n, burst_time)
```

OUTPUT :-

```
[Running] python -u "c:\Users\Admin\Desktop\LP1\FCFS.py"
Processes Burst time Waiting time Turn around time
 1      10   0       10
 2      5    10      15
 3      8    15      23
Average waiting time = 8.333333333333334
Average turn around time = 16.0

[Done] exited with code=0 in 0.061 seconds
```

SJF (Pre-emptive):-

```
def main():
    # Taking the number of processes
    n = int(input("Enter number of processes: "))
    # Matrix for storing Process ID, Burst Time, Waiting Time & Turnaround Time
    A = [[0 for _ in range(4)] for _ in range(100)]
    total, avg_wt, avg_tat = 0, 0, 0
    print("Enter Burst Time:")
    # User input for Burst Time and assigning Process ID
    for i in range(n):
        A[i][1] = int(input(f"P{i + 1}: "))  # Burst Time
        A[i][0] = i + 1  # Process ID
    # Sorting processes according to their Burst Time
    for i in range(n):
        index = i
        for j in range(i + 1, n):
            if A[j][1] < A[index][1]:
                index = j

        # Swap Burst Times
        A[i][1], A[index][1] = A[index][1], A[i][1]
        # Swap Process IDs
        A[i][0], A[index][0] = A[index][0], A[i][0]
    # Initialize waiting time for the first process
    A[0][2] = 0
    # Calculate waiting times
    for i in range(1, n):
        A[i][2] = sum(A[j][1] for j in range(i))  # Waiting time
        total += A[i][2]
    avg_wt = total / n
    total = 0
```

```
    # Calculation of Turnaround Time and printing the data
    print("P\t BT\t WT\t TAT")
    for i in range(n):
        A[i][3] = A[i][1] + A[i][2]  # Turnaround time
        total += A[i][3]
        print(f"P{A[i][0]}\t {A[i][1]}\t {A[i][2]}\t {A[i][3]}")
    avg_tat = total / n
    print(f"Average Waiting Time = {avg_wt}")
    print(f"Average Turnaround Time = {avg_tat}")
if __name__ == "__main__":
    main()
```

OUTPUT :-

```
PS C:\Users\Admin> & C:/Users/Admin/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Admin/Desktop/LP1/SJF.py
Enter number of processes: 4
Enter Burst Time:
P1: 1
P2: 2
P3: 2
P4: 1
P        BT      WT      TAT
P1       1       0       1
P4       1       1       2
P3       2       2       4
P2       2       4       6
Average Waiting Time = 1.75
Average Turnaround Time = 3.25
```

PRIORITY (Non-Pre-emptive) :-

```
def main():
    # Taking the number of processes
    n = int(input("Enter the number of processes: "))
    # Arrays to store process details
    p = [0] * 10  # Process IDs
    pp = [0] * 10  # Priorities
    bt = [0] * 10  # Burst Times
    w = [0] * 10  # Waiting Times
    t = [0] * 10  # Turnaround Times
    print("Enter the burst time and priority for each process:")
    for i in range(n):
        print(f"Process[{i + 1}]")
        bt[i] = int(input("Burst Time: "))  # Input Burst Time
        pp[i] = int(input("Priority: "))    # Input Priority
        p[i] = i + 1  # Assigning Process ID

    # Sorting processes based on priority (higher number indicates higher priority)
    for i in range(n - 1):
        for j in range(i + 1, n):
            if pp[i] < pp[j]:
                # Swap priorities and corresponding burst times and IDs
                pp[i], pp[j] = pp[j], pp[i]
                bt[i], bt[j] = bt[j], bt[i]
                p[i], p[j] = p[j], p[i]

    # Initialize waiting time and turnaround time
    w[0] = 0
    awt = 0
    t[0] = bt[0]
```

```
    atat = t[0]
    # Calculate waiting and turnaround times
    for i in range(1, n):
        w[i] = t[i - 1]  # Waiting time
        awt += w[i]
        t[i] = w[i] + bt[i]  # Turnaround time
        atat += t[i]

    # Print the process details
    print("Process \t Burst time \t Wait time \t TAT \t Priority ")
    for i in range(n):
        print(f"{p[i]}\t\t{bt[i]}\t\t{w[i]}\t\t{t[i]}\t\t{pp[i]}")

    # Calculate and print average waiting time and turnaround time
    awt /= n
    atat /= n
    print(f"Average Wait time: {awt}")
    print(f"Average TAT: {atat}")
if __name__ == "__main__":
    main()
```

OUTPUT :-

```
PS C:\Users\Admin> & C:/Users/Admin/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Admin/Desktop/LP1/priority.py
Enter the number of processes: 4
Enter the burst time and priority for each process:
Process[1]
Burst Time: 1
Priority: 1
Process[2]
Burst Time: 2
Priority: 2
Process[3]
Burst Time: 3
Priority: 2
Process[4]
Burst Time: 4
Priority: 3
Process          Burst time      Wait time       TAT     Priority
4               4               0               4               3
3               3               4               7               2
2               2               7               9               2
1               1               9               10              1
Average Wait time: 5.0
Average TAT: 7.5
```

ROUND ROBIN (Pre-emptive):-

```
def findWaitingTime(processes, n, bt, wt, quantum):
    rem_bt = [0] * n  # Remaining burst times

    # Copy the burst time into rem_bt[]
    for i in range(n):
        rem_bt[i] = bt[i]
    t = 0  # Current time

    while True:
        done = True

        for i in range(n):
            if rem_bt[i] > 0:  # There is a pending process
                done = False  # Mark as not done

                if rem_bt[i] > quantum:
                    t += quantum
                    rem_bt[i] -= quantum
                else:
                    t += rem_bt[i]
                    wt[i] = t - bt[i]  # Waiting time
                    rem_bt[i] = 0  # Process finished
        # If all processes are done
        if done:
            break
def findTurnAroundTime(processes, n, bt, wt, tat):
    # Calculating turnaround time
    for i in range(n):
        tat[i] = bt[i] + wt[i]  # TAT = BT + WT
```

```
def findavgTime(processes, n, bt, quantum):
    wt = [0] * n  # Waiting time
    tat = [0] * n  # Turnaround time
    findWaitingTime(processes, n, bt, wt, quantum)
    findTurnAroundTime(processes, n, bt, wt, tat)
    # Display processes along with all details
    print("Processes Burst Time  Waiting Time Turn-Around Time")
    total_wt = 0
    total_tat = 0
    for i in range(n):
        total_wt += wt[i]
        total_tat += tat[i]
        print(f" {i + 1}\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))
# Driver code
if __name__ == "__main__":
    # Process IDs
    proc = [1, 2, 3]
    n = 3
    # Burst time of all processes
    burst_time = [10, 5, 8]
    # Time quantum
    quantum = 2
    findavgTime(proc, n, burst_time, quantum)
```

OUTPUT :-

```
PS C:\Users\Admin> & C:/Users/Admin/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Admin/Desktop/LP1/RoundRobin_pre.py
Processes Burst Time  Waiting Time Turn-Around Time
 1              10             13             23
 2              5              10             15
 3              8              13             21

Average waiting time = 12.00000
Average turn around time = 19.66667
```