A.Y.: 2022-23 **Sub:** System Fundamentals

Experiment 3

(CPU Scheduling – Preemptive algorithm)

Aim: Implement CPU Scheduling – Preemptive Algorithm.

Theory: CPU scheduling is the task of selecting a waiting process from the ready queue and allocating the CPU to it. The CPU is allocated to the selected process by the dispatcher (Itis the module that gives control of the CPU to the processes by short-term scheduler). Scheduling is a fundamental operating system function. In a single processor system, onlyone process can run at a time; any other process must wait until the CPU is free and can berescheduled. The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization.

Types of scheduling based on decision mode:

1. **Non-Preemptive:** Under this scheduling, once the CPU has been allocatedto a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.
2. **Preemptive:** Under this scheduling, once the CPU has been allocated to aprocess, the process does not keep the CPU but can be utilized by some other process. This incurs a cost associated with access to shared data. It also affects the design of the operating system kernel.

Scheduling Criteria:

1. **CPU utilization:** It can range from 0-100%. In a real system, it ranges shouldrange from 40-90%.
2. **Throughput:** Number of processes that are completed per unit time.
3. **Turnaround time:** How long a process takes to execute. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O
4. **Waiting time:** It is the sum of the periods spent waiting in the ready queue.
5. **Response time:** Time from the submission of a request until the first response is produced. It is desirable to maximize CPU utilization and Throughput and minimize Turnaround time, waiting time and Response time.

PREEMPTIVE SCHEDULING ALGORITHMS:

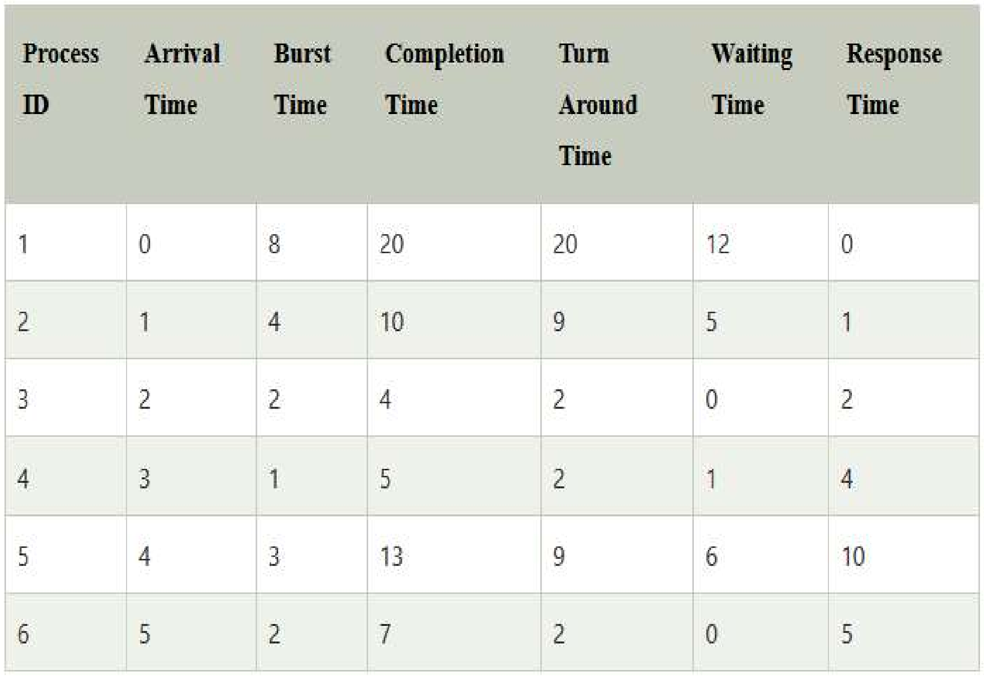
1. Shortest Remaining Time First
2. Round Robin Scheduling

1. Shortest Remaining Time First

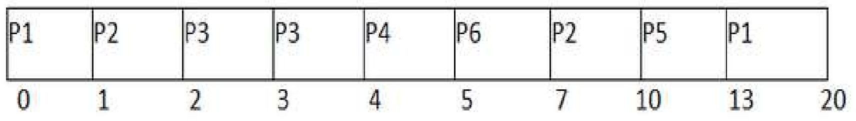
In SRTF, the execution of the process can be stopped after certain amount of time. At the arrival of every process, the short term scheduler schedules the process with the least remaining burst time among the list of available processes and the running process.

2. **Round Robin Scheduling:**
   - It is designed especially for time-sharing systems.
   - It is like FCFS, but preemption is added to switch between processes.
   - A time quantum is defined.
   - The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum. If a process's CPU burst exceeds 1 time quantum, that process is preempted and is put back in the ready queue.

<u>Example</u>:



Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time	Response Time
1	0	8	20	20	12	0
2	1	4	10	9	5	1
3	2	2	4	2	0	2
4	3	1	5	2	1	4
5	4	3	13	9	6	10
6	5	2	7	2	0	5



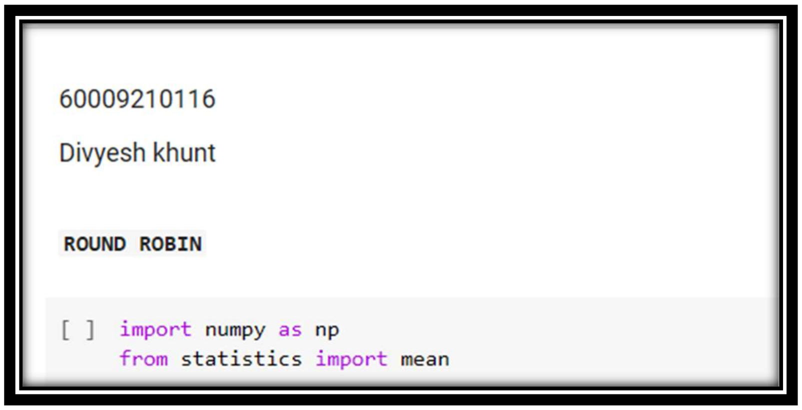
Avg Waiting Time = 24/6

A.Y.: 2022-23 **Sub:** System Fundamentals

Lab Assignments to complete in this session

1. Consider the set of 6 processes arrived at zero instance and burst time are 7, 5, 3, 1, 2, 1. If the CPU scheduling policy is shortest remaining time first, calculate the average waiting time.
2. Consider the set of 3 processes arrived at zero instance and burst time are 9, 4, 9 respectively. If theCPU scheduling policy is Round Robin with time quantum=3ms, calculate the average waiting time and average turnaround time.

A.Y.: 2022-23 **Sub:** System Fundamentals

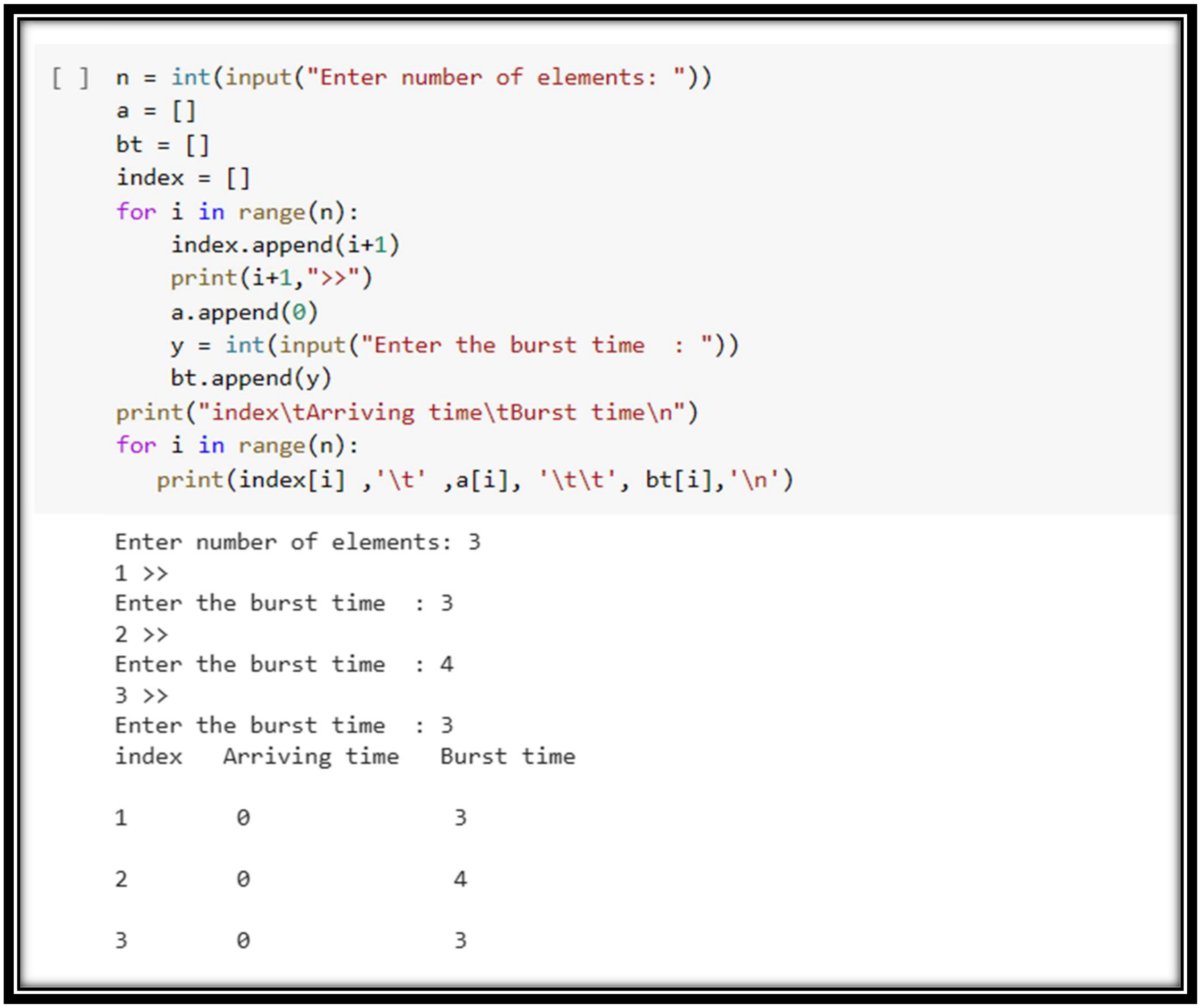


60009210116

Divyesh khunt

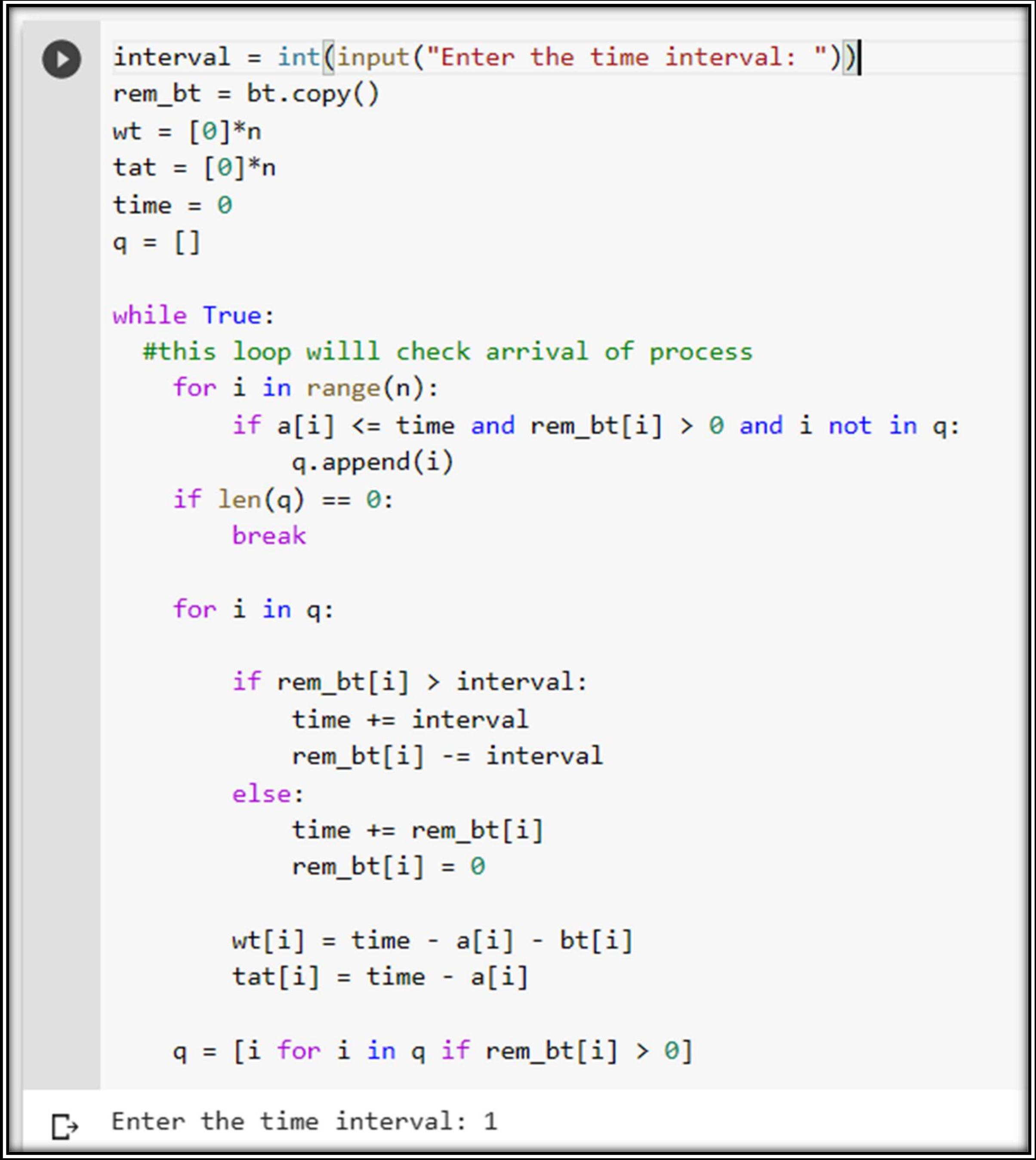
ROUND ROBIN

```
[ ]  import numpy as np
     from statistics import mean
```



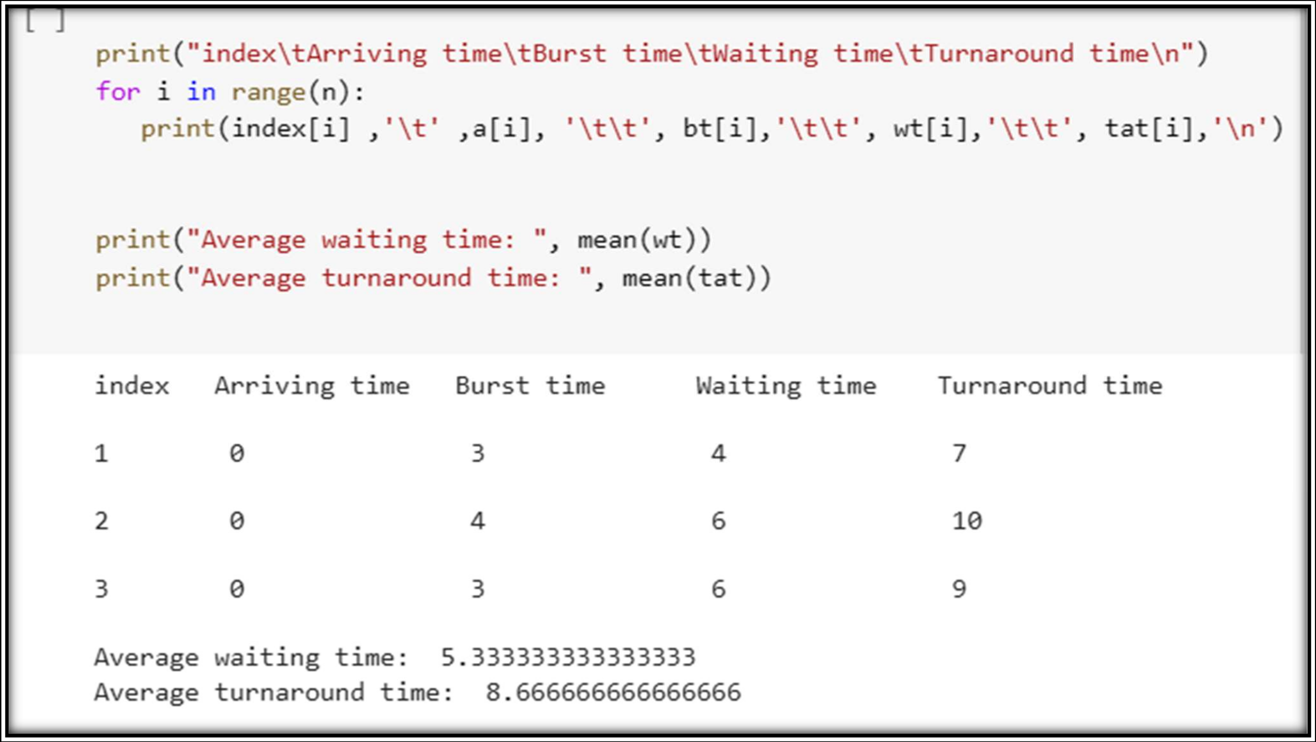
```
[ ]  n = int(input("Enter number of elements: "))
     a = []
     bt = []
     index = []
     for i in range(n):
         index.append(i+1)
         print(i+1,">>")
         a.append(0)
         y = int(input("Enter the burst time  : "))
         bt.append(y)
     print("index\tArriving time\tBurst time\n")
     for i in range(n):
        print(index[i] ,'\t' ,a[i], '\t\t', bt[i],'\n')
```

```
     Enter number of elements: 3
     1 >>
     Enter the burst time  : 3
     2 >>
     Enter the burst time  : 4
     3 >>
     Enter the burst time  : 3
     index   Arriving time   Burst time

     1        0               3

     2        0               4

     3        0               3
```



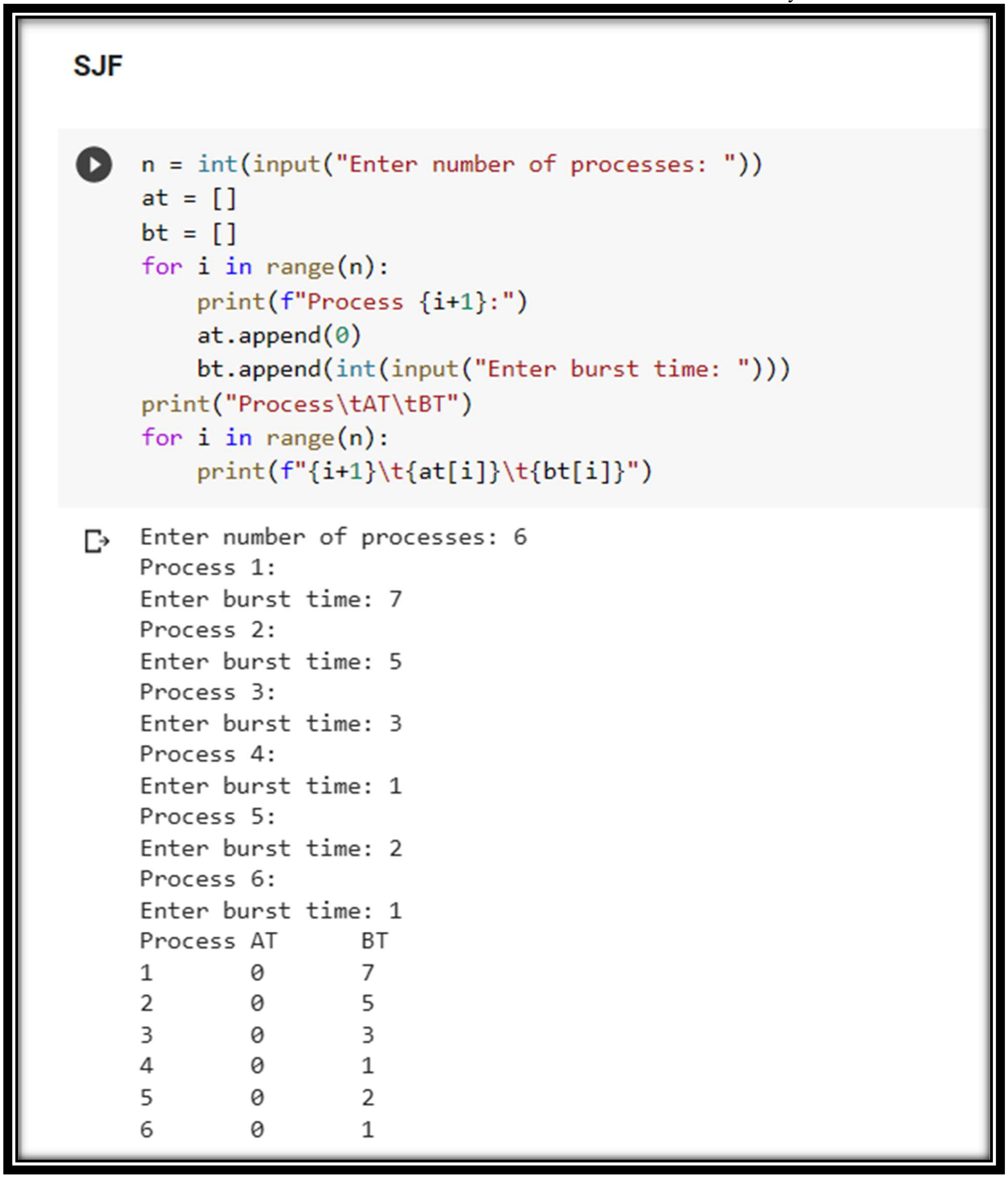
```
interval = int(input("Enter the time interval: "))
rem_bt = bt.copy()
wt = [0]*n
tat = [0]*n
time = 0
q = []

while True:
  #this loop willl check arrival of process
    for i in range(n):
        if a[i] <= time and rem_bt[i] > 0 and i not in q:
            q.append(i)
    if len(q) == 0:
        break

    for i in q:

        if rem_bt[i] > interval:
            time += interval
            rem_bt[i] -= interval
        else:
            time += rem_bt[i]
            rem_bt[i] = 0

        wt[i] = time - a[i] - bt[i]
        tat[i] = time - a[i]

    q = [i for i in q if rem_bt[i] > 0]
```

```
Enter the time interval: 1
```



```
[ ]
    print("index\tArriving time\tBurst time\tWaiting time\tTurnaround time\n")
    for i in range(n):
       print(index[i] ,'\t' ,a[i], '\t\t', bt[i],'\t\t', wt[i],'\t\t', tat[i],'\n')


    print("Average waiting time: ", mean(wt))
    print("Average turnaround time: ", mean(tat))
```

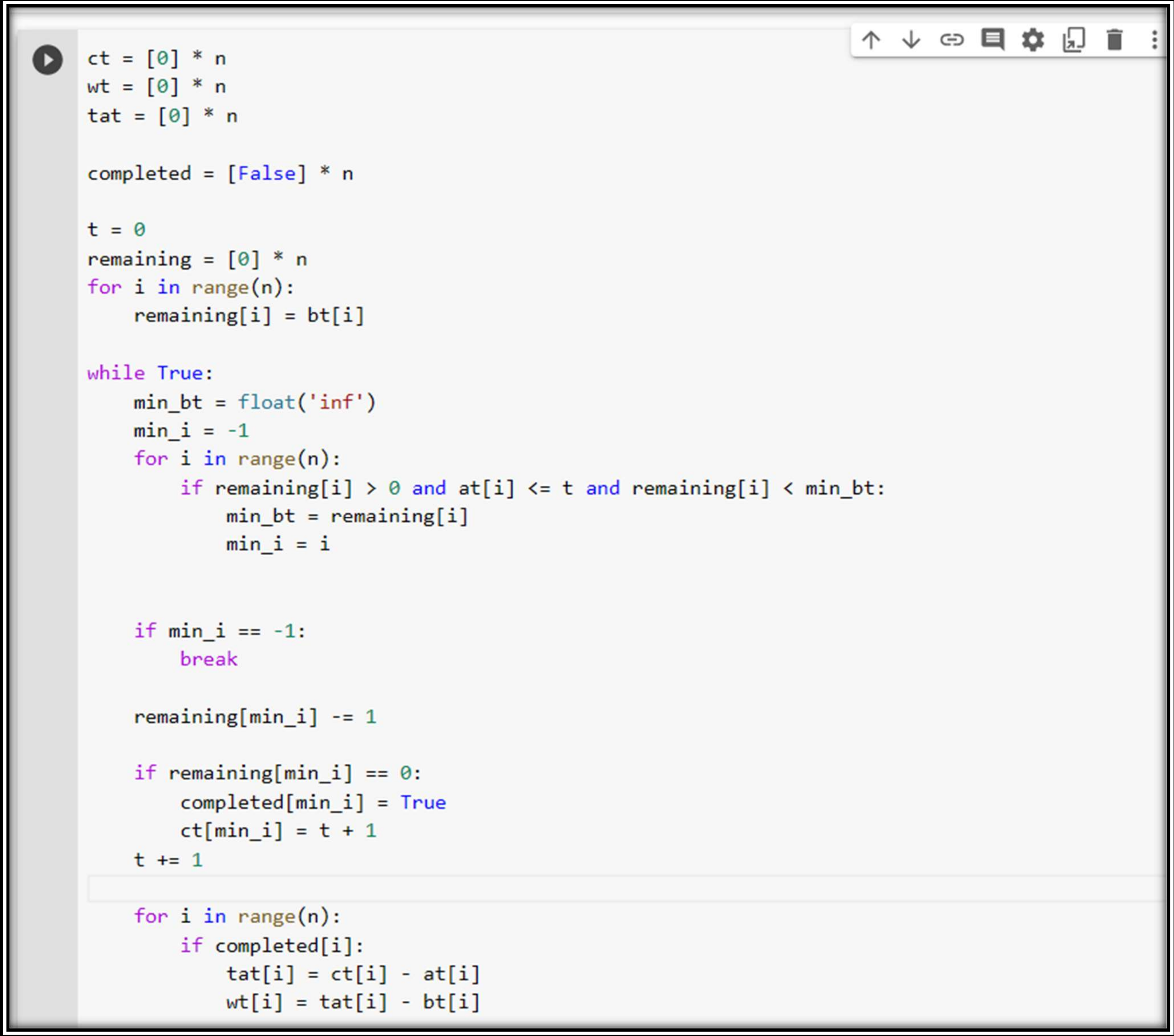
```
index   Arriving time   Burst time      Waiting time    Turnaround time

1        0               3               4               7

2        0               4               6               10

3        0               3               6               9

Average waiting time:  5.333333333333333
Average turnaround time:  8.666666666666666
```



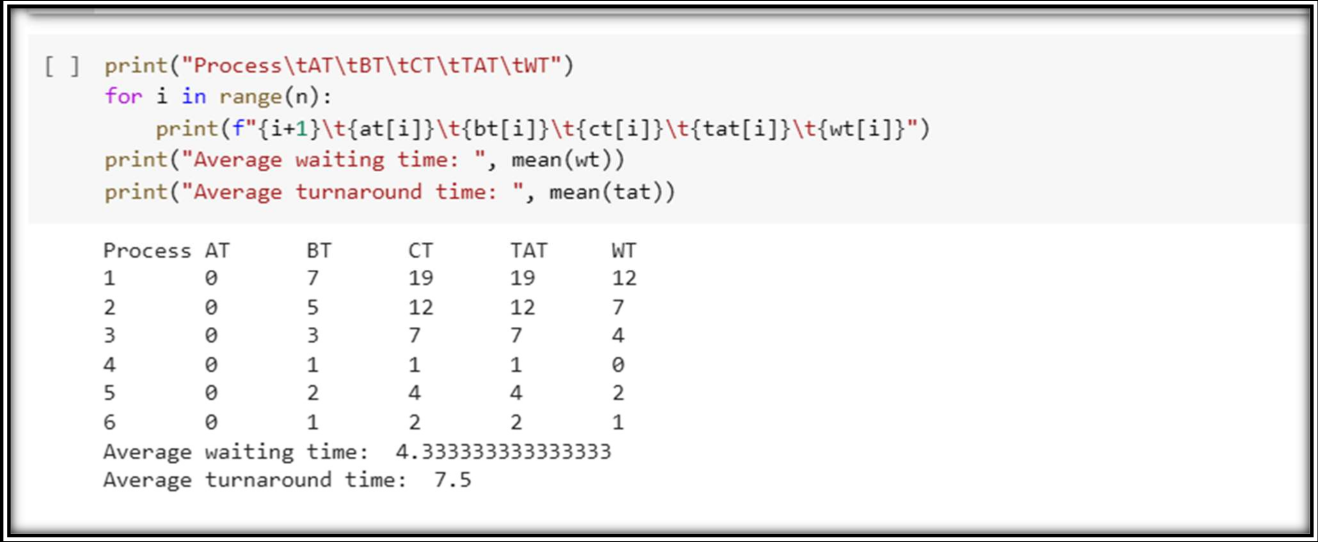
SJF

```python
n = int(input("Enter number of processes: "))
at = []
bt = []
for i in range(n):
    print(f"Process {i+1}:")
    at.append(0)
    bt.append(int(input("Enter burst time: ")))
print("Process\tAT\tBT")
for i in range(n):
    print(f"{i+1}\t{at[i]}\t{bt[i]}")
```

```
Enter number of processes: 6
Process 1:
Enter burst time: 7
Process 2:
Enter burst time: 5
Process 3:
Enter burst time: 3
Process 4:
Enter burst time: 1
Process 5:
Enter burst time: 2
Process 6:
Enter burst time: 1
Process AT      BT
1       0       7
2       0       5
3       0       3
4       0       1
5       0       2
6       0       1
```

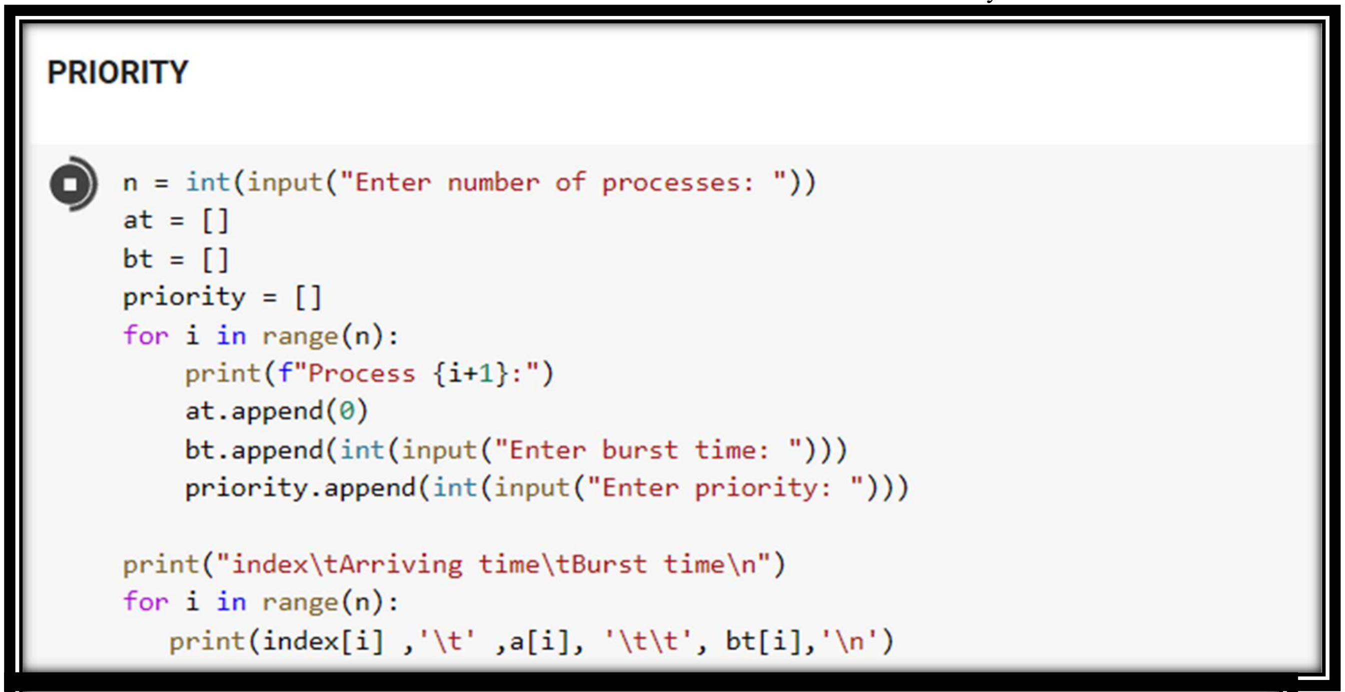


```
ct = [0] * n
wt = [0] * n
tat = [0] * n

completed = [False] * n

t = 0
remaining = [0] * n
for i in range(n):
    remaining[i] = bt[i]

while True:
    min_bt = float('inf')
    min_i = -1
    for i in range(n):
        if remaining[i] > 0 and at[i] <= t and remaining[i] < min_bt:
            min_bt = remaining[i]
            min_i = i


    if min_i == -1:
        break

    remaining[min_i] -= 1

    if remaining[min_i] == 0:
        completed[min_i] = True
        ct[min_i] = t + 1
    t += 1

    for i in range(n):
        if completed[i]:
            tat[i] = ct[i] - at[i]
            wt[i] = tat[i] - bt[i]
```



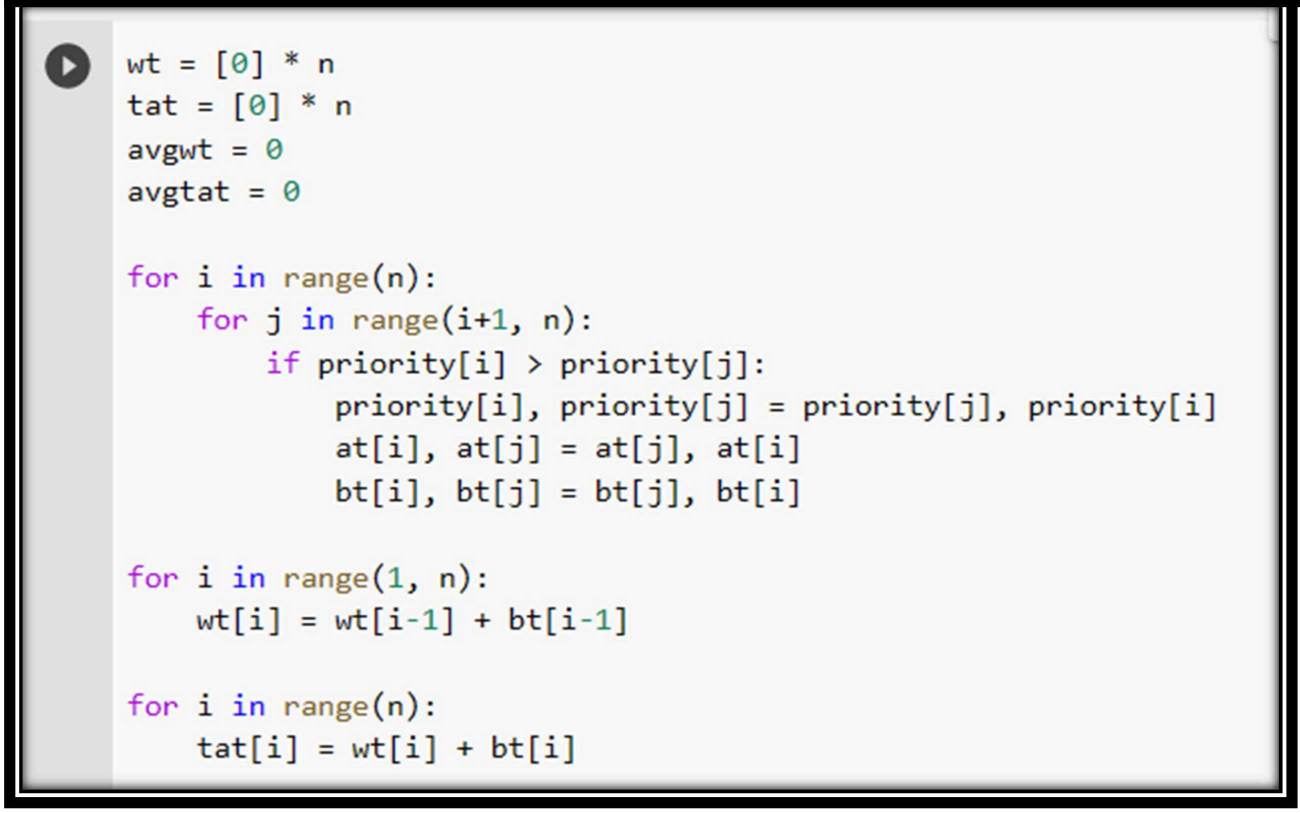
```
[ ] print("Process\tAT\tBT\tCT\tTAT\tWT")
    for i in range(n):
        print(f"{i+1}\t{at[i]}\t{bt[i]}\t{ct[i]}\t{tat[i]}\t{wt[i]}")
    print("Average waiting time: ", mean(wt))
    print("Average turnaround time: ", mean(tat))
```

```
Process AT      BT      CT      TAT     WT
1       0       7       19      19      12
2       0       5       12      12      7
3       0       3       7       7       4
4       0       1       1       1       0
5       0       2       4       4       2
6       0       1       2       2       1
Average waiting time:  4.333333333333333
Average turnaround time:  7.5
```

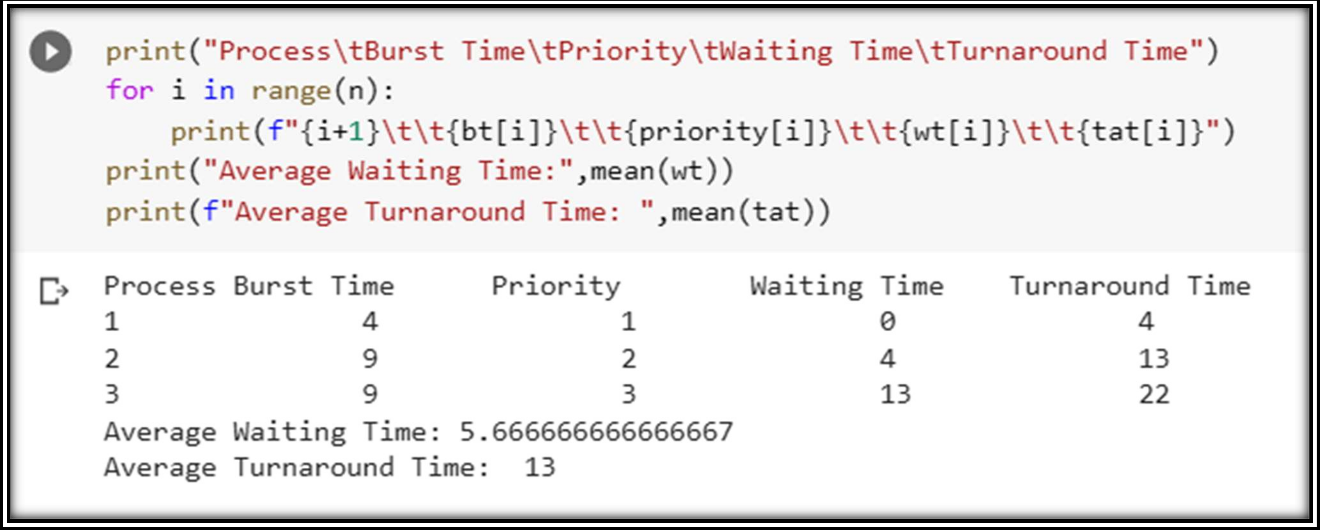


PRIORITY

```
n = int(input("Enter number of processes: "))
at = []
bt = []
priority = []
for i in range(n):
    print(f"Process {i+1}:")
    at.append(0)
    bt.append(int(input("Enter burst time: ")))
    priority.append(int(input("Enter priority: ")))

print("index\tArriving time\tBurst time\n")
for i in range(n):
   print(index[i] ,'\t' ,a[i], '\t\t', bt[i],'\n')
```



```
wt = [0] * n
tat = [0] * n
avgwt = 0
avgtat = 0

for i in range(n):
    for j in range(i+1, n):
        if priority[i] > priority[j]:
            priority[i], priority[j] = priority[j], priority[i]
            at[i], at[j] = at[j], at[i]
            bt[i], bt[j] = bt[j], bt[i]

for i in range(1, n):
    wt[i] = wt[i-1] + bt[i-1]

for i in range(n):
    tat[i] = wt[i] + bt[i]
```



```
print("Process\tBurst Time\tPriority\tWaiting Time\tTurnaround Time")
for i in range(n):
    print(f"{i+1}\t\t{bt[i]}\t\t{priority[i]}\t\t{wt[i]}\t\t{tat[i]}")
print("Average Waiting Time:",mean(wt))
print(f"Average Turnaround Time: ",mean(tat))
```

```
Process Burst Time      Priority        Waiting Time    Turnaround Time
1               4               1               0               4
2               9               2               4               13
3               9               3               13              22
Average Waiting Time: 5.666666666666667
Average Turnaround Time:  13
```