Usman Institute of Technology

Department of Computer Science Fall 2022

Name: Muhammad Waleed

Roll no: 20B-115-SE

Course: Operating Systems (CS312)

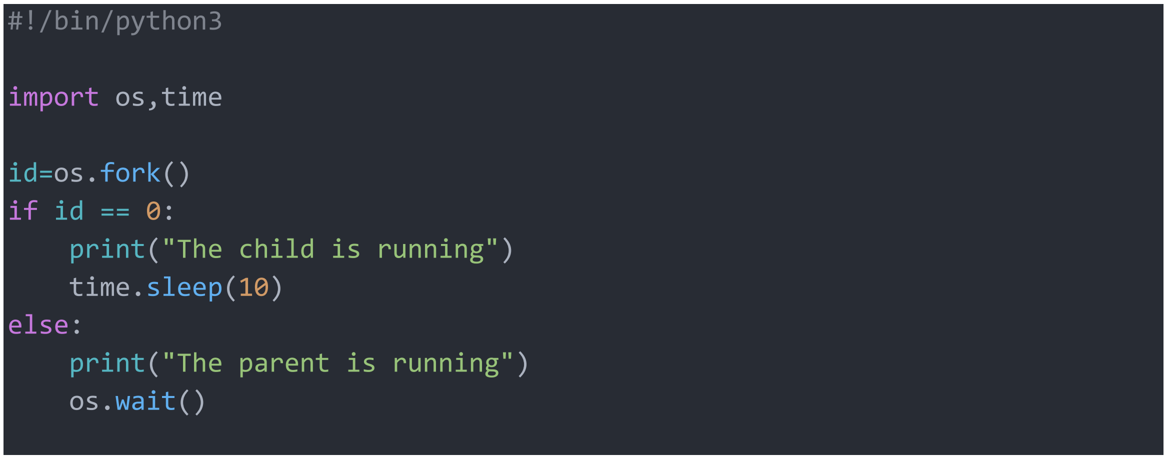
Course Instructor: Ma’am Shabina Mushtaq

Date: 17-Nov-2022

Muhammad Waleed
20b-115-se
Operating Systems
Lab#06

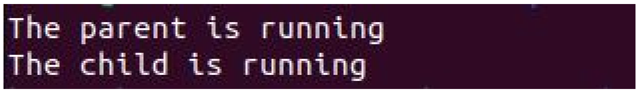
Lab Tasks:

1. Using a Linux system, write a program that forks a child process that ultimately becomes a zombie process. This zombie process must remain in the system for at least 10 seconds.



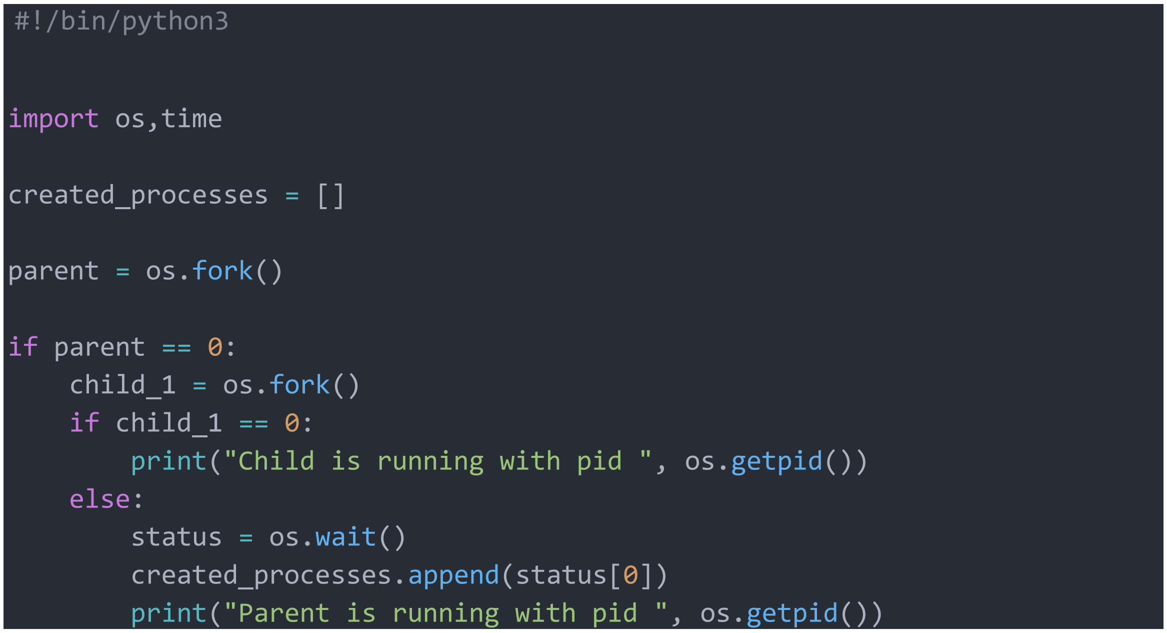
```
#!/bin/python3

import os,time

id=os.fork()
if id == 0:
    print("The child is running")
    time.sleep(10)
else:
    print("The parent is running")
    os.wait()
```

Output:



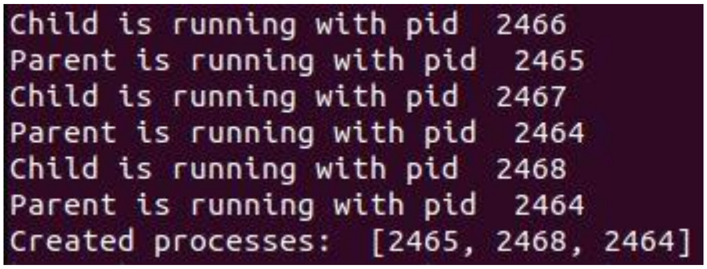
```
The parent is running
The child is running
```

2. Write a program that creates a child process which further creates its two child processes. Store the process id of each process in an array called Created Processes. Also display the process id of the terminated child to understand the hierarchy of termination of each child process.



```
#!/bin/python3


import os,time

created_processes = []

parent = os.fork()

if parent == 0:
    child_1 = os.fork()
    if child_1 == 0:
        print("Child is running with pid ", os.getpid())
    else:
        status = os.wait()
        created_processes.append(status[0])
        print("Parent is running with pid ", os.getpid())
        child_2 = os.fork()
        if child_2 == 0:
            print("Child is running with pid ", os.getpid())
        else:
            status = os.wait()
            created_processes.append(status[0])
else:
    status = os.wait()
    created_processes.append(status[0])
    print("Parent is running with pid ", os.getpid())
    child_3 = os.fork()
    if child_3 == 0:
        print("Child is running with pid ", os.getpid())
    else:
        status = os.wait()
        created_processes.append(status[0])
        print("Parent is running with pid ", os.getpid())
        created_processes.append(os.getpid())
        print("Created processes: ", created_processes)
```

Output:

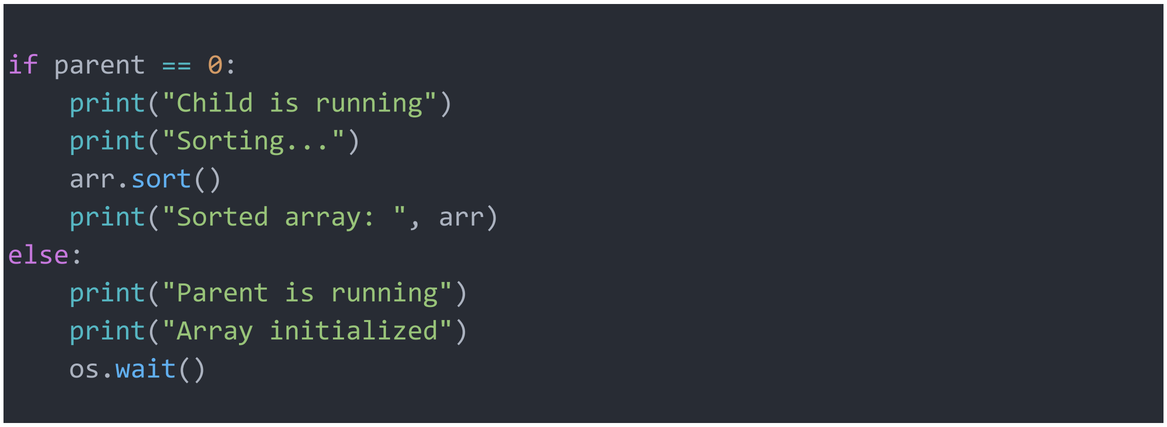


```
Child is running with pid  2466
Parent is running with pid  2465
Child is running with pid  2467
Parent is running with pid  2464
Child is running with pid  2468
Parent is running with pid  2464
Created processes:  [2465, 2468, 2464]
```

3. Write a program in which a parent process will initialize an array, and child process will sort this array. Use wait() and sleep() methods to achieve the synchronization such that parent process should run first.

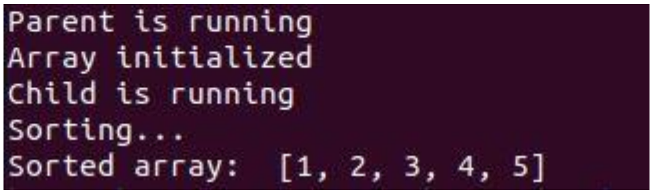
```
#!/bin/python3

import os, time

arr = [1, 3, 2, 5, 4]
parent = os.fork()
```

Muhammad Waleed
20b-115-se
Operating Systems
Lab#06



```
if parent == 0:
    print("Child is running")
    print("Sorting...")
    arr.sort()
    print("Sorted array: ", arr)
else:
    print("Parent is running")
    print("Array initialized")
    os.wait()
```

Output:



```
Parent is running
Array initialized
Child is running
Sorting...
Sorted array:  [1, 2, 3, 4, 5]
```

Name: Muhammad Waleed

Roll no: 20B-115-SE

Course: Operating Systems (CS312)

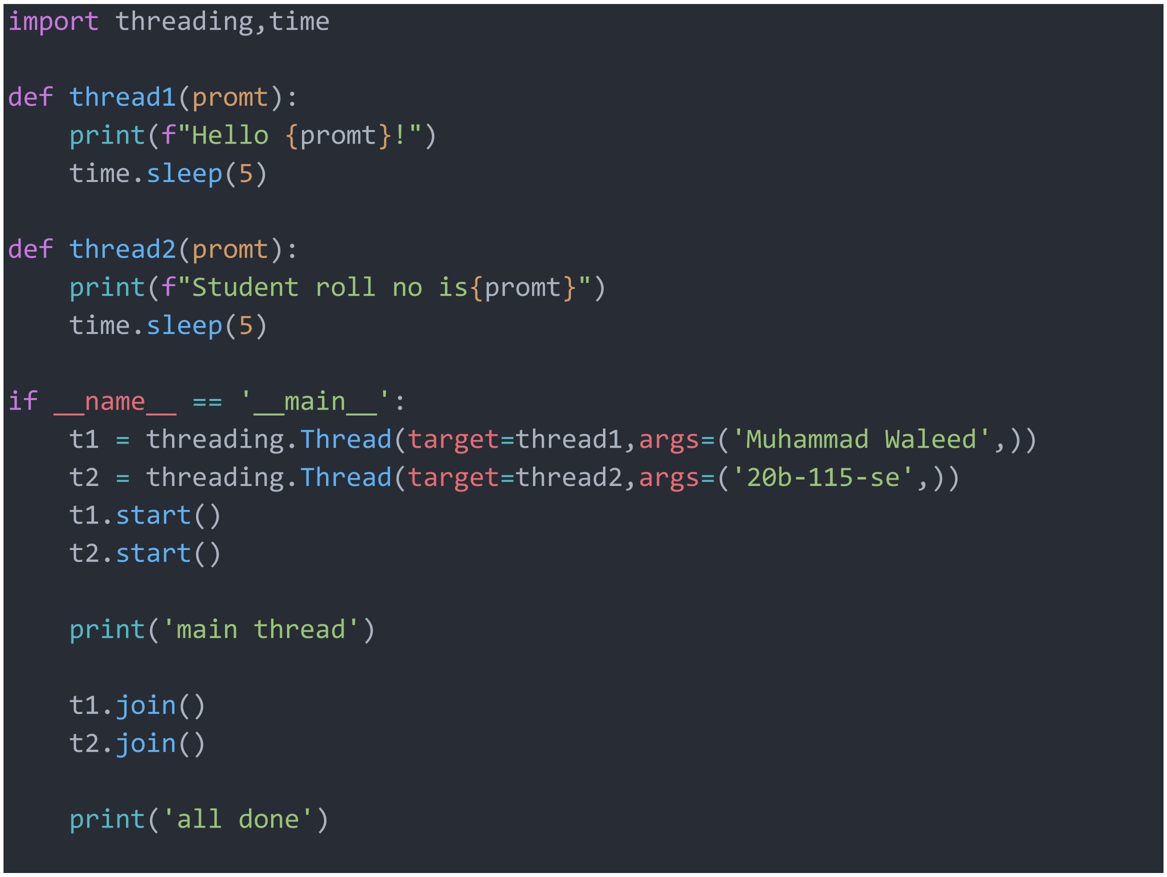
Course Instructor: Ma’am Shabina Mushtaq

Date: 10-Nov-2022

Muhammad Waleed
20b-115-se
SE-B
OS Lab#07
Ma'am Shabina Mushtaq

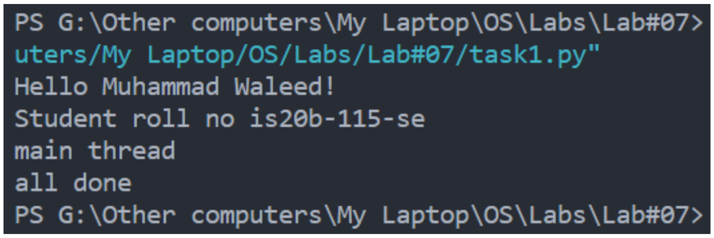
Lab Tasks:

1. Modify Example 1 to display strings via two independent threads: thread1: "Hello ! StudentName___", thread 2: "Student roll no is :__________"



```python
import threading,time

def thread1(promt):
    print(f"Hello {promt}!")
    time.sleep(5)

def thread2(promt):
    print(f"Student roll no is{promt}")
    time.sleep(5)

if __name__ == '__main__':
    t1 = threading.Thread(target=thread1,args=('Muhammad Waleed',))
    t2 = threading.Thread(target=thread2,args=('20b-115-se',))
    t1.start()
    t2.start()

    print('main thread')

    t1.join()
    t2.join()

    print('all done')
```

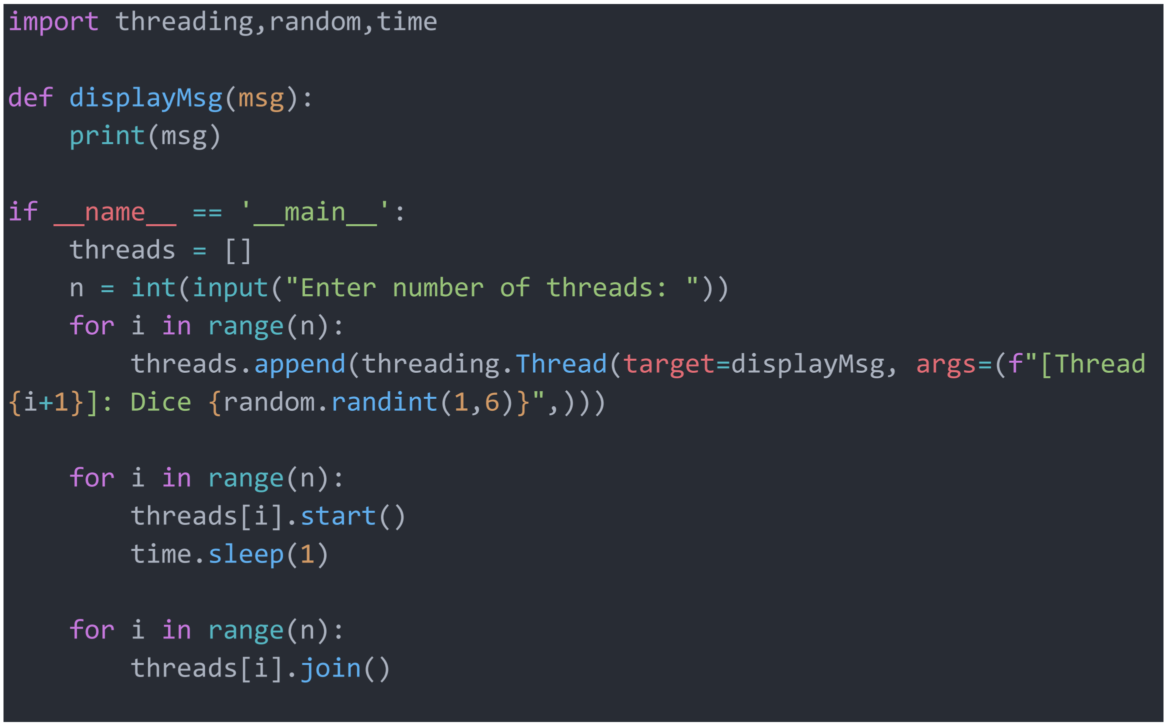
Output:



```
PS G:\Other computers\My Laptop\OS\Labs\Lab#07>
uters/My Laptop/OS/Labs/Lab#07/task1.py"
Hello Muhammad Waleed!
Student roll no is20b-115-se
main thread
all done
PS G:\Other computers\My Laptop\OS\Labs\Lab#07>
```

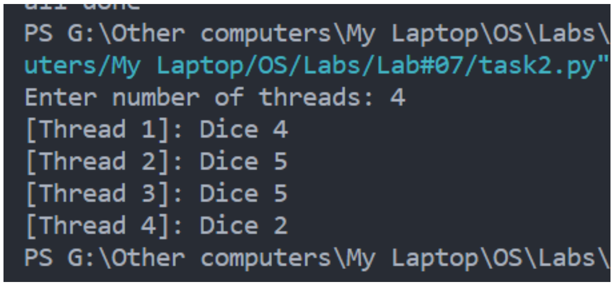
Muhammad Waleed
20b-115-se
SE-B
OS Lab#07
Ma’am Shabina Mushtaq

2. Create threads message as many times as user wants to create threads by using array of threads and loop. Threads should display message that is passed through argument.



```
import threading,random,time

def displayMsg(msg):
    print(msg)

if __name__ == '__main__':
    threads = []
    n = int(input("Enter number of threads: "))
    for i in range(n):
        threads.append(threading.Thread(target=displayMsg, args=(f"[Thread
{i+1}]: Dice {random.randint(1,6)}",)))

    for i in range(n):
        threads[i].start()
        time.sleep(1)

    for i in range(n):
        threads[i].join()
```

Output:



```
all done
PS G:\Other computers\My Laptop\OS\Labs\
uters/My Laptop/OS/Labs/Lab#07/task2.py"
Enter number of threads: 4
[Thread 1]: Dice 4
[Thread 2]: Dice 5
[Thread 3]: Dice 5
[Thread 4]: Dice 2
PS G:\Other computers\My Laptop\OS\Labs\
```

Usman Institute of Technology

Department of Computer Science Fall 2022

Name: Muhammad Waleed

Roll no: 20B-115-SE

Course: Operating Systems (CS312)

Course Instructor: Ma'am Shabina Mushtaq

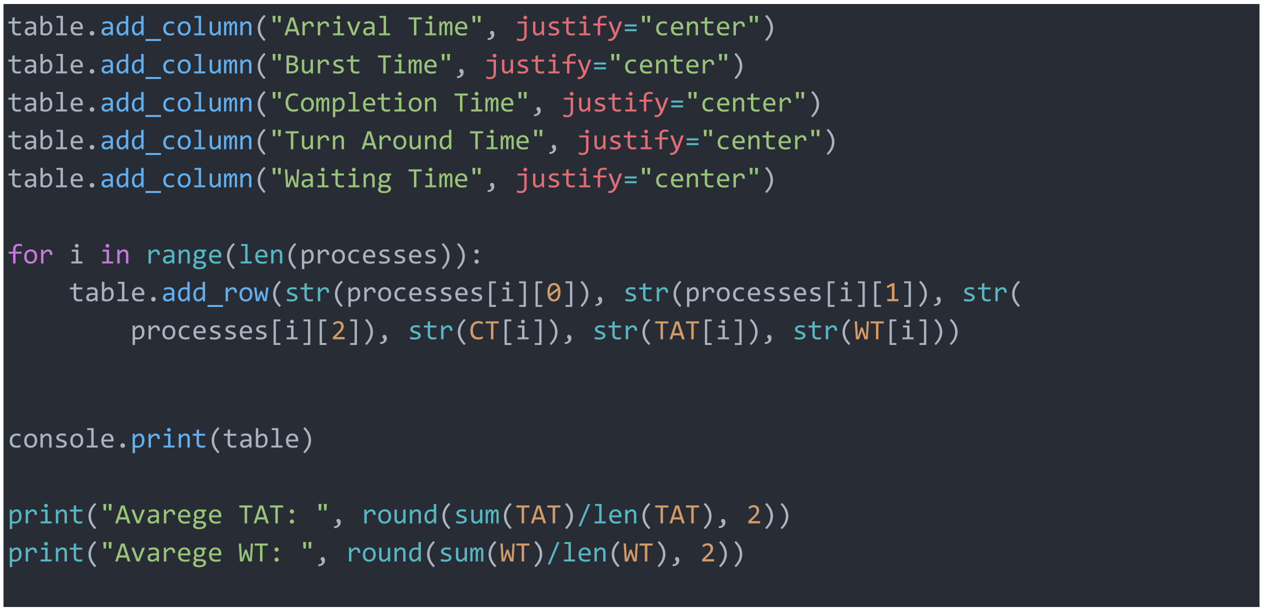
Date: 24-Nov-2022

Muhammad Waleed
20b-115-se
SE-B
OS Lab#08
Ma’am Shabina Mushtaq

FCFS (with arrival time 0):

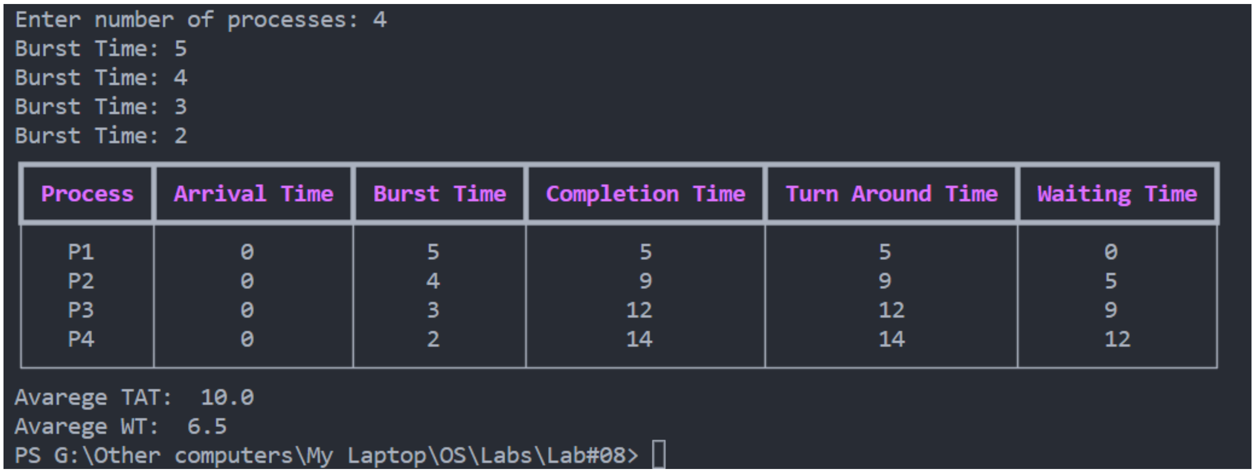
```
import os
try:
    from rich.console import Console
    from rich.table import Table
except ModuleNotFoundError:
    os.system("pip install rich")
    from rich.console import Console
    from rich.table import Table

console = Console()
table = Table(show_header=True, header_style="bold magenta")
os.system("cls")

nprocess = int(input("Enter number of processes: "))
processes = []
CT = []
TAT = []
WT = []
for i in range(nprocess):
    b = int(input("Burst Time: "))
    processes.append(["P"+str(i+1), 0, b])

# sort According to arrival time
processes.sort(key=lambda x: x[1])
# Calculting Completion time
for i in range(len(processes)):
    if i == 0:
        CT.append(processes[i][2])
    else:
        CT.append(CT[i-1]+processes[i][2])

# Calculation Turn Around Time
for i in range(len(processes)):
    TAT.append(CT[i]-processes[i][1])

# Calculation Waiting Time
for i in range(len(processes)):
    WT.append(TAT[i]-processes[i][2])

table.add_column("Process", justify="center")
```

Muhammad Waleed
20b-115-se
SE-B
OS Lab#08
Ma’am Shabina Mushtaq



```python
table.add_column("Arrival Time", justify="center")
table.add_column("Burst Time", justify="center")
table.add_column("Completion Time", justify="center")
table.add_column("Turn Around Time", justify="center")
table.add_column("Waiting Time", justify="center")

for i in range(len(processes)):
    table.add_row(str(processes[i][0]), str(processes[i][1]), str(
        processes[i][2]), str(CT[i]), str(TAT[i]), str(WT[i]))


console.print(table)

print("Avarege TAT: ", round(sum(TAT)/len(TAT), 2))
print("Avarege WT: ", round(sum(WT)/len(WT), 2))
```

Output:



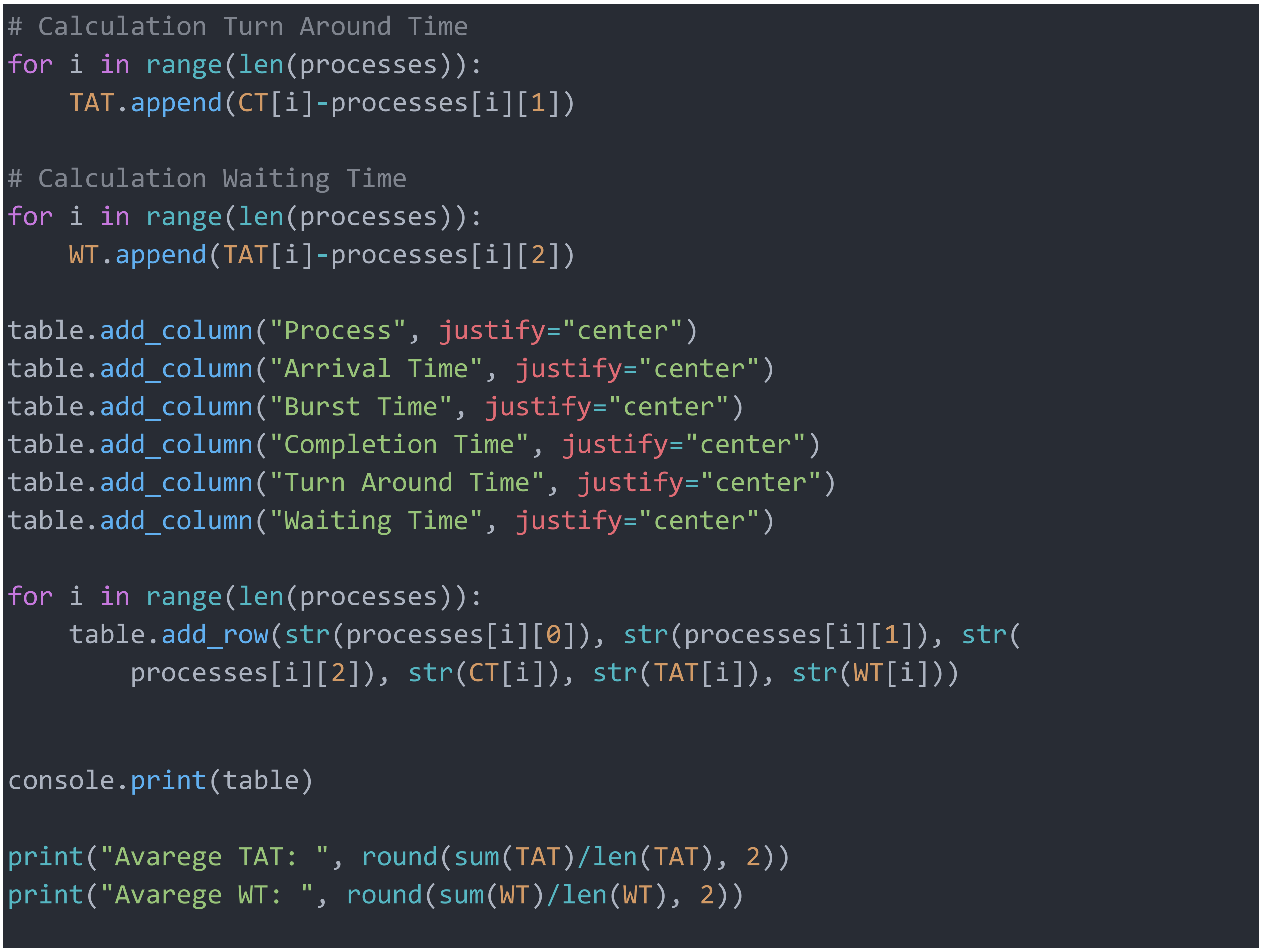
```
Enter number of processes: 4
Burst Time: 5
Burst Time: 4
Burst Time: 3
Burst Time: 2
┏━━━━━━━━━┳━━━━━━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━┓
┃ Process ┃ Arrival Time ┃ Burst Time ┃ Completion Time ┃ Turn Around Time ┃ Waiting Time ┃
┡━━━━━━━━━╇━━━━━━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━┩
│   P1    │      0       │     5      │        5        │        5         │      0       │
│   P2    │      0       │     4      │        9        │        9         │      5       │
│   P3    │      0       │     3      │       12        │        12        │      9       │
│   P4    │      0       │     2      │       14        │        14        │      12      │
└─────────┴──────────────┴────────────┴─────────────────┴──────────────────┴──────────────┘
Avarege TAT:  10.0
Avarege WT:  6.5
PS G:\Other computers\My Laptop\OS\Labs\Lab#08> 
```

Muhammad Waleed
20b-115-se
SE-B
OS Lab#08
Ma’am Shabina Mushtaq

SJF (with arrival time 0):

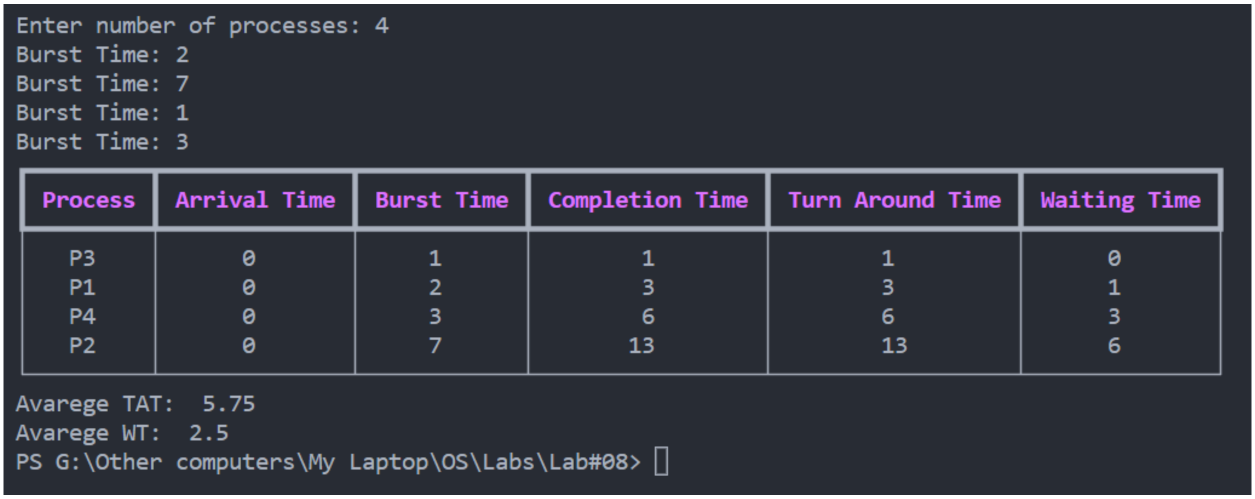
```
import os
try:
    from rich.console import Console
    from rich.table import Table
except ModuleNotFoundError:
    os.system("pip install rich")
    from rich.console import Console
    from rich.table import Table

console = Console()
table = Table(show_header=True, header_style="bold magenta")
os.system("cls")

nprocess = int(input("Enter number of processes: "))
processes = []
CT = []
TAT = []
WT = []
for i in range(nprocess):
    b = int(input("Burst Time: "))
    processes.append(["P"+str(i+1), 0, b])

# sort According to burst time
processes.sort(key=lambda x: x[2])

# Calculting Completion time
for i in range(len(processes)):
    if i == 0:
        if processes[i][1] > 0:
            state_idle = processes[i][1]
            CT.append(processes[i][2]+state_idle)
        else:
            CT.append(processes[i][2])
    else:
        if CT[i-1] < processes[i][1]:
            idle_state = processes[i][1] - CT[i-1]
            CT.append(CT[i-1]+processes[i][2]+idle_state)
        else:
            CT.append(CT[i-1]+processes[i][2])
```

Muhammad Waleed
20b-115-se
SE-B
OS Lab#08
Ma'am Shabina Mushtaq



```
# Calculation Turn Around Time
for i in range(len(processes)):
    TAT.append(CT[i]-processes[i][1])

# Calculation Waiting Time
for i in range(len(processes)):
    WT.append(TAT[i]-processes[i][2])

table.add_column("Process", justify="center")
table.add_column("Arrival Time", justify="center")
table.add_column("Burst Time", justify="center")
table.add_column("Completion Time", justify="center")
table.add_column("Turn Around Time", justify="center")
table.add_column("Waiting Time", justify="center")

for i in range(len(processes)):
    table.add_row(str(processes[i][0]), str(processes[i][1]), str(
        processes[i][2]), str(CT[i]), str(TAT[i]), str(WT[i]))


console.print(table)

print("Avarege TAT: ", round(sum(TAT)/len(TAT), 2))
print("Avarege WT: ", round(sum(WT)/len(WT), 2))
```

Output:



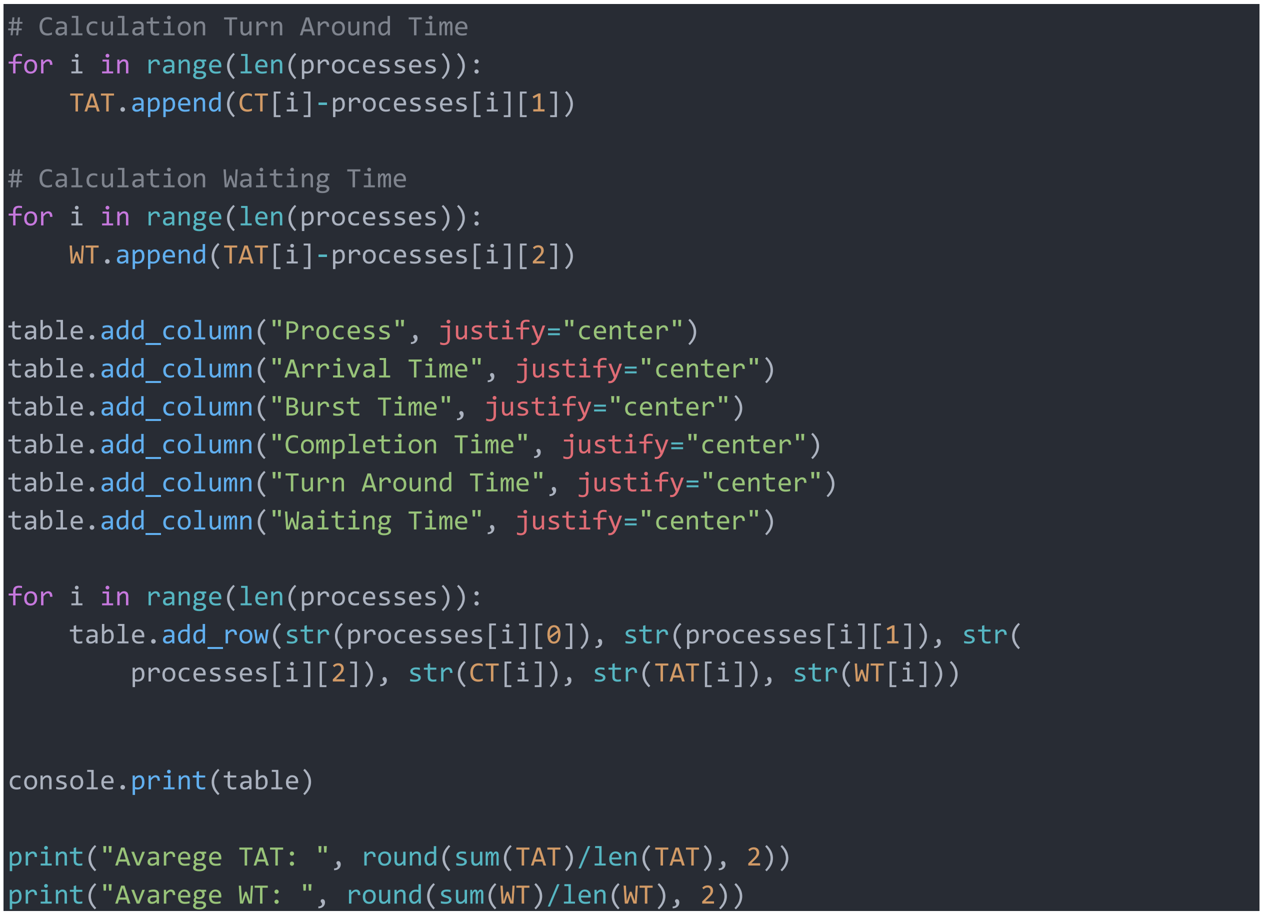
```
Enter number of processes: 4
Burst Time: 2
Burst Time: 7
Burst Time: 1
Burst Time: 3
┏━━━━━━━━━┳━━━━━━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━┓
┃ Process ┃ Arrival Time ┃ Burst Time ┃ Completion Time ┃ Turn Around Time ┃ Waiting Time ┃
┡━━━━━━━━━╇━━━━━━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━┩
│   P3    │      0       │     1      │        1        │        1         │      0       │
│   P1    │      0       │     2      │        3        │        3         │      1       │
│   P4    │      0       │     3      │        6        │        6         │      3       │
│   P2    │      0       │     7      │       13        │        13        │      6       │
└─────────┴──────────────┴────────────┴─────────────────┴──────────────────┴──────────────┘
Avarege TAT:  5.75
Avarege WT:  2.5
PS G:\Other computers\My Laptop\OS\Labs\Lab#08> 
```

Muhammad Waleed
20b-115-se
SE-B
OS Lab#08
Ma'am Shabina Mushtaq

Modified FCFS for different arrival time and idleness:

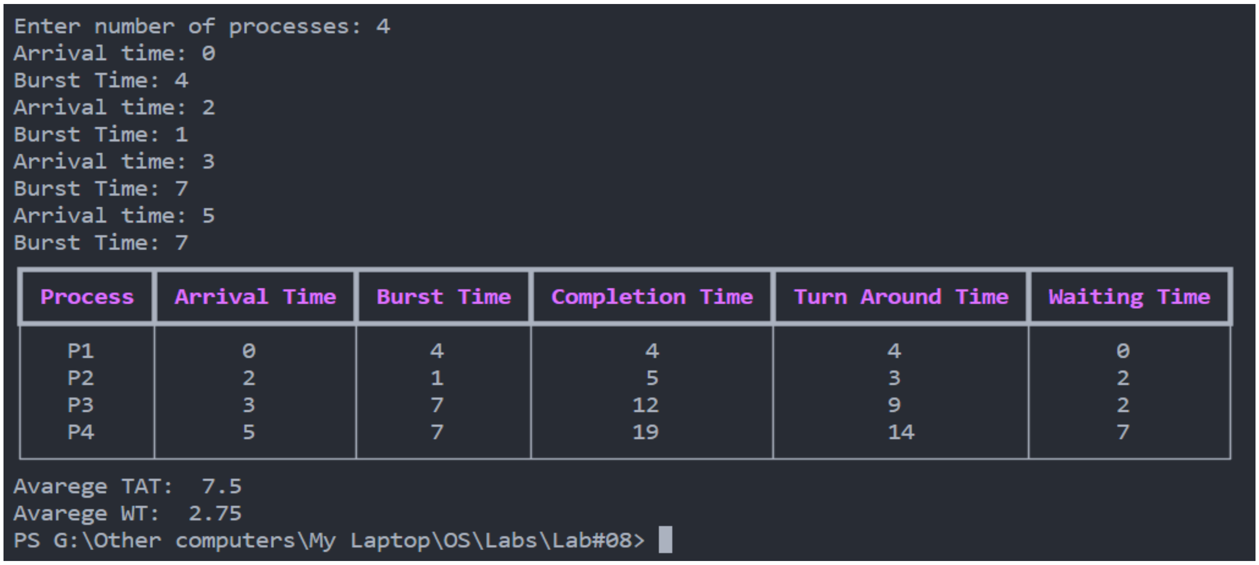
```
import os
try:
    from rich.console import Console
    from rich.table import Table
except ModuleNotFoundError:
    os.system("pip install rich")
    from rich.console import Console
    from rich.table import Table

console = Console()
table = Table(show_header=True, header_style="bold magenta")
os.system("cls")

nprocess = int(input("Enter number of processes: "))
processes = []
CT = []
TAT = []
WT = []
for i in range(nprocess):
    a = int(input("Arrival time: "))
    b = int(input("Burst Time: "))
    processes.append(["P"+str(i+1), a, b])

# sort According to arrival time
processes.sort(key=lambda x: x[1])
# Calculting Completion time
for i in range(len(processes)):
    if i == 0:
        if processes[i][1] > 0:
            state_idle = processes[i][1]
            CT.append(processes[i][2]+state_idle)
        else:
            CT.append(processes[i][2])
    else:
        if CT[i-1] < processes[i][1]:
            idle_state = processes[i][1] - CT[i-1]
            CT.append(CT[i-1]+processes[i][2]+idle_state)
        else:
            CT.append(CT[i-1]+processes[i][2])
```

Muhammad Waleed
20b-115-se
SE-B
OS Lab#08
Ma'am Shabina Mushtaq



```
# Calculation Turn Around Time
for i in range(len(processes)):
    TAT.append(CT[i]-processes[i][1])

# Calculation Waiting Time
for i in range(len(processes)):
    WT.append(TAT[i]-processes[i][2])

table.add_column("Process", justify="center")
table.add_column("Arrival Time", justify="center")
table.add_column("Burst Time", justify="center")
table.add_column("Completion Time", justify="center")
table.add_column("Turn Around Time", justify="center")
table.add_column("Waiting Time", justify="center")

for i in range(len(processes)):
    table.add_row(str(processes[i][0]), str(processes[i][1]), str(
        processes[i][2]), str(CT[i]), str(TAT[i]), str(WT[i]))


console.print(table)

print("Avarege TAT: ", round(sum(TAT)/len(TAT), 2))
print("Avarege WT: ", round(sum(WT)/len(WT), 2))
```

Output:



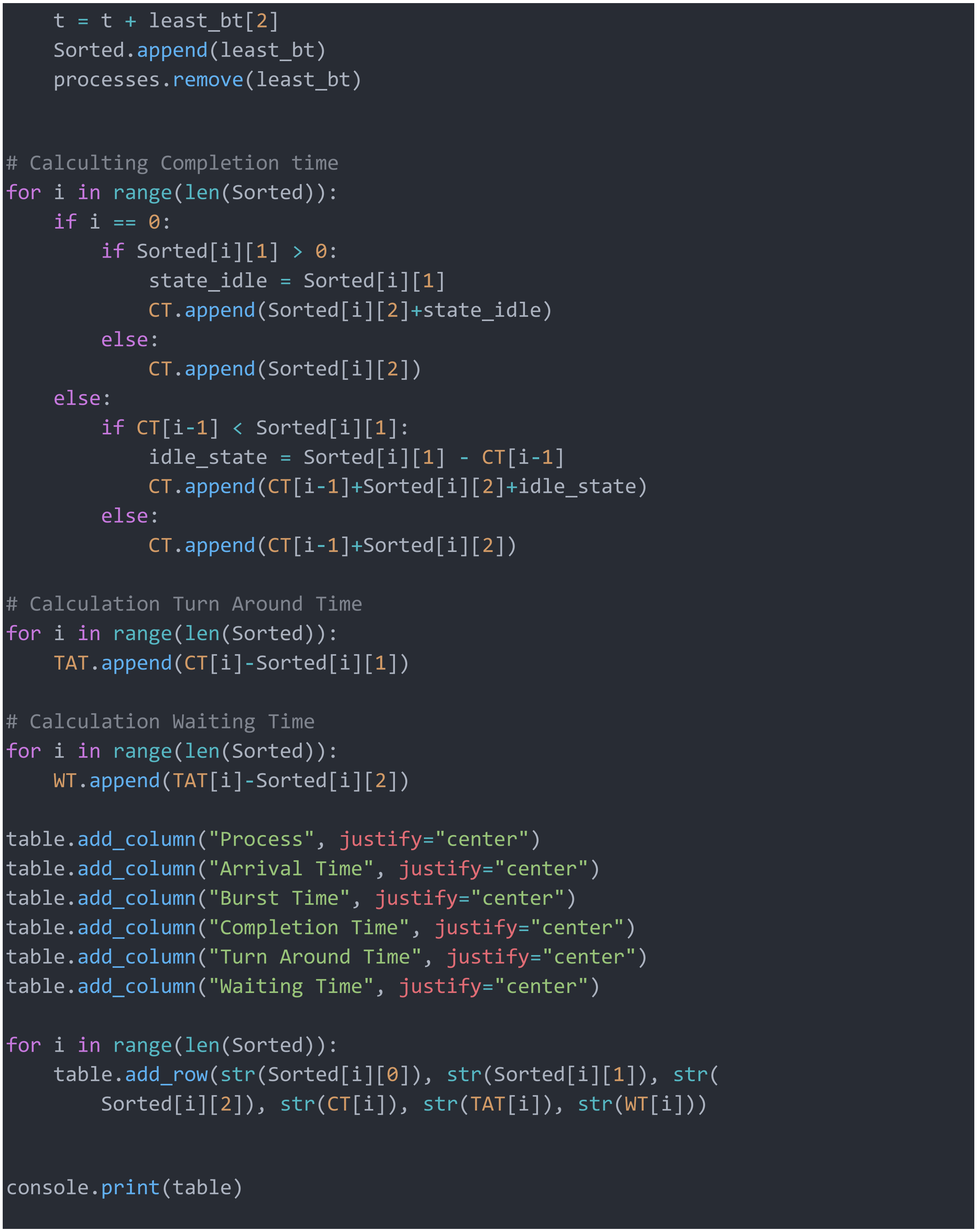
```
Enter number of processes: 4
Arrival time: 0
Burst Time: 4
Arrival time: 2
Burst Time: 1
Arrival time: 3
Burst Time: 7
Arrival time: 5
Burst Time: 7
┏━━━━━━━━━┳━━━━━━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━┓
┃ Process ┃ Arrival Time ┃ Burst Time ┃ Completion Time ┃ Turn Around Time ┃ Waiting Time ┃
┡━━━━━━━━━╇━━━━━━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━┩
│   P1    │      0       │     4      │        4        │        4         │      0       │
│   P2    │      2       │     1      │        5        │        3         │      2       │
│   P3    │      3       │     7      │       12        │        9         │      2       │
│   P4    │      5       │     7      │       19        │        14        │      7       │
└─────────┴──────────────┴────────────┴─────────────────┴──────────────────┴──────────────┘
Avarege TAT:  7.5
Avarege WT:  2.75
PS G:\Other computers\My Laptop\OS\Labs\Lab#08> 
```

Muhammad Waleed
20b-115-se
SE-B
OS Lab#08
Ma’am Shabina Mushtaq

SJF (with different arrival time):

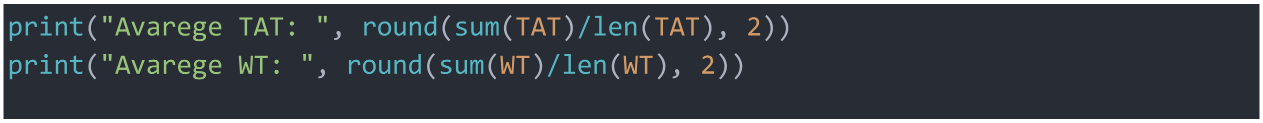
```
import os
try:
    from rich.console import Console
    from rich.table import Table
except ModuleNotFoundError:
    os.system("pip install rich")
    from rich.console import Console
    from rich.table import Table

console = Console()
table = Table(show_header=True, header_style="bold magenta")
os.system("cls")

nprocess = int(input("Enter number of processes: "))
processes = []
Sorted = []
CT = []
TAT = []
WT = []
for i in range(nprocess):
    a = int(input("Arrival time: "))
    b = int(input("Burst Time: "))
    processes.append(["P"+str(i+1), a, b])

n = len(processes)
# arranging
t = min(processes, key=lambda x: x[1])
t = t[1]
for i in range(n):
    reach_pro = []
    flag = True
    while flag == True:
        for j in range(len(processes)):
            if processes[j][1] <= t:
                reach_pro.append(processes[j])
        if len(reach_pro) == 0:
            t += 1
        else:
            flag = False
    least_bt = min(reach_pro, key=lambda x: x[2])
```

Muhammad Waleed
20b-115-se
SE-B
OS Lab#08
Ma’am Shabina Mushtaq



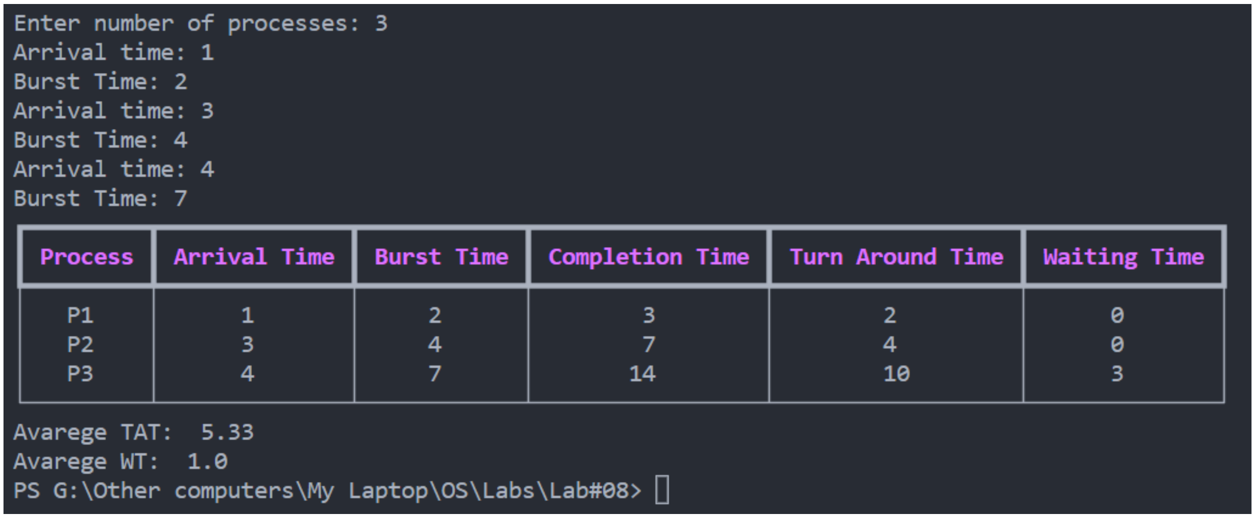
```
    t = t + least_bt[2]
    Sorted.append(least_bt)
    processes.remove(least_bt)


# Calculting Completion time
for i in range(len(Sorted)):
    if i == 0:
        if Sorted[i][1] > 0:
            state_idle = Sorted[i][1]
            CT.append(Sorted[i][2]+state_idle)
        else:
            CT.append(Sorted[i][2])
    else:
        if CT[i-1] < Sorted[i][1]:
            idle_state = Sorted[i][1] - CT[i-1]
            CT.append(CT[i-1]+Sorted[i][2]+idle_state)
        else:
            CT.append(CT[i-1]+Sorted[i][2])

# Calculation Turn Around Time
for i in range(len(Sorted)):
    TAT.append(CT[i]-Sorted[i][1])

# Calculation Waiting Time
for i in range(len(Sorted)):
    WT.append(TAT[i]-Sorted[i][2])

table.add_column("Process", justify="center")
table.add_column("Arrival Time", justify="center")
table.add_column("Burst Time", justify="center")
table.add_column("Completion Time", justify="center")
table.add_column("Turn Around Time", justify="center")
table.add_column("Waiting Time", justify="center")

for i in range(len(Sorted)):
    table.add_row(str(Sorted[i][0]), str(Sorted[i][1]), str(
        Sorted[i][2]), str(CT[i]), str(TAT[i]), str(WT[i]))


console.print(table)
```

Muhammad Waleed
20b-115-se
SE-B
OS Lab#08
Ma’am Shabina Mushtaq



```python
print("Avarege TAT: ", round(sum(TAT)/len(TAT), 2))
print("Avarege WT: ", round(sum(WT)/len(WT), 2))
```

Output:



```
Enter number of processes: 3
Arrival time: 1
Burst Time: 2
Arrival time: 3
Burst Time: 4
Arrival time: 4
Burst Time: 7
| Process | Arrival Time | Burst Time | Completion Time | Turn Around Time | Waiting Time |

|   P1    |      1       |     2      |        3        |        2         |      0       |
|   P2    |      3       |     4      |        7        |        4         |      0       |
|   P3    |      4       |     7      |       14        |       10         |      3       |

Avarege TAT:  5.33
Avarege WT:  1.0
PS G:\Other computers\My Laptop\OS\Labs\Lab#08> 
```

Usman Institute of Technology

Department of Computer Science Fall 2022

Name: Muhammad Waleed

Roll no: 20B-115-SE

Course: Operating Systems (CS312)

Course Instructor: Ma’am Shabina Mushtaq

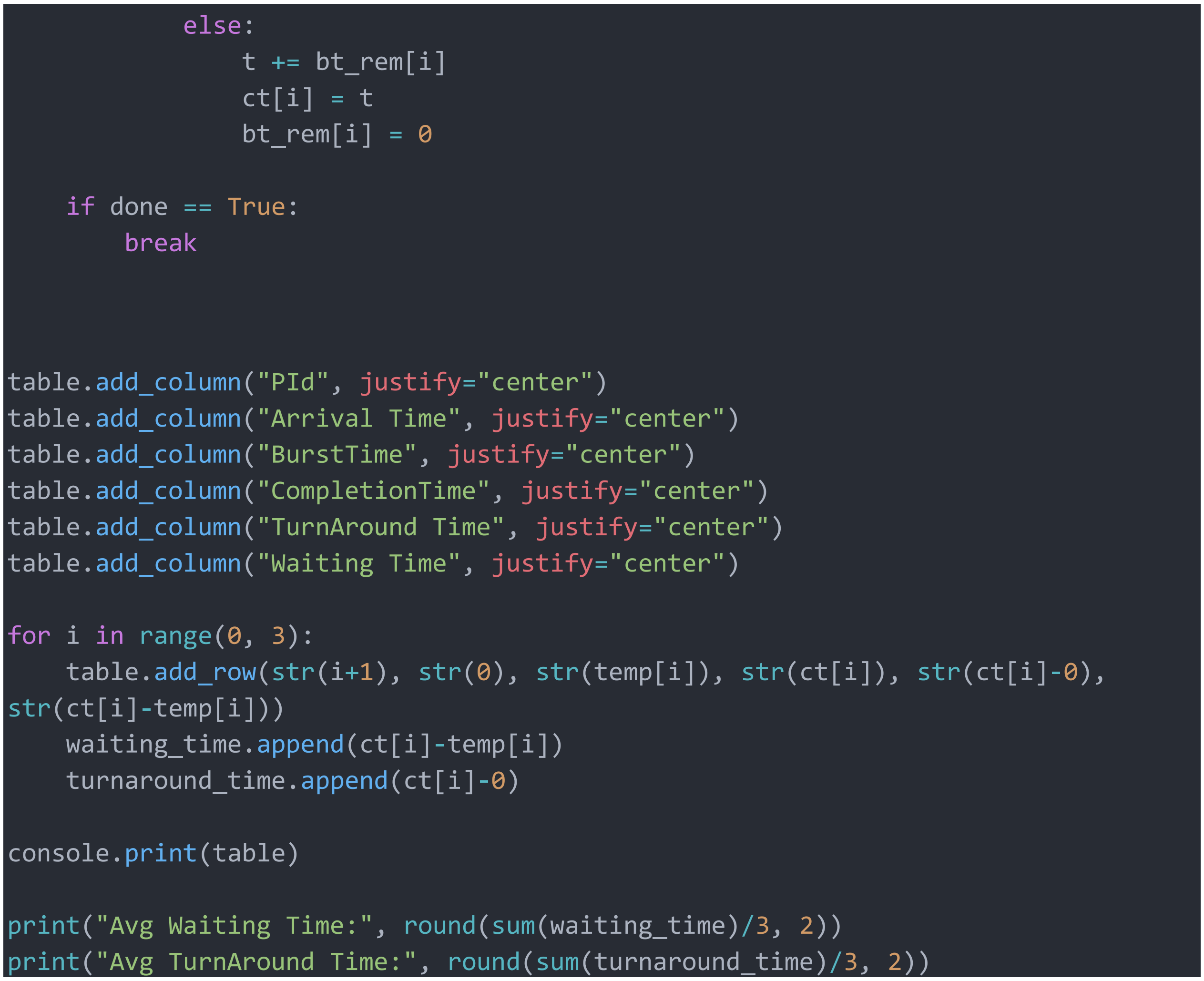
Date: 1-Dec-2022

Muhammad Waleed
20b-115-se
SE-B
OS Lab#09
Ma’am Shabina Mushtaq

Round Robins:

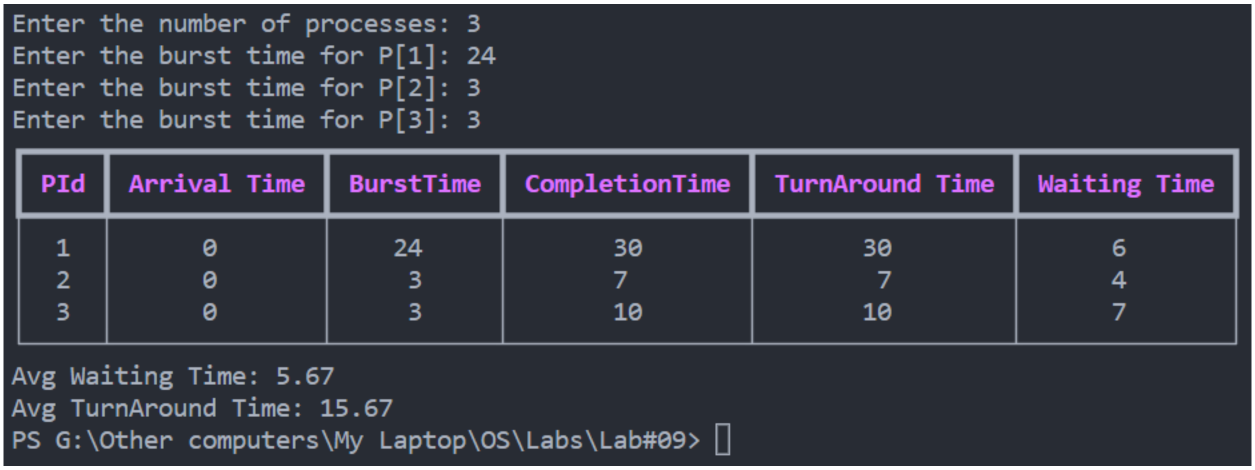
```
import os

try:
    from rich.console import Console
    from rich.table import Table
except ImportError:
    os.system("pip install rich")
    from rich.console import Console
    from rich.table import Table

console = Console()
table = Table(show_header=True, header_style="bold magenta")

os.system("cls")

q = 4 # Quantum Time
t = 0 # Current Time

nprocess = int(input("Enter the number of processes: "))
bt_rem = [] # Burst Time Remaining

for i in range(nprocess):
    bt = int(input("Enter the burst time for P[{}]: ".format(i+1)))
    bt_rem.append(bt)

ct = [0 for i in range(nprocess)]

temp = bt_rem.copy()

waiting_time = []
turnaround_time = []

while 1:
    done = True
    for i in range(0, 3):
        if bt_rem[i] > 0:
            done = False
            if bt_rem[i] > q:
                t += q
                bt_rem[i] -= q
```

Muhammad Waleed
20b-115-se
SE-B
OS Lab#09
Ma'am Shabina Mushtaq



```
            else:
                t += bt_rem[i]
                ct[i] = t
                bt_rem[i] = 0

    if done == True:
        break



table.add_column("PId", justify="center")
table.add_column("Arrival Time", justify="center")
table.add_column("BurstTime", justify="center")
table.add_column("CompletionTime", justify="center")
table.add_column("TurnAround Time", justify="center")
table.add_column("Waiting Time", justify="center")

for i in range(0, 3):
    table.add_row(str(i+1), str(0), str(temp[i]), str(ct[i]), str(ct[i]-0),
str(ct[i]-temp[i]))
    waiting_time.append(ct[i]-temp[i])
    turnaround_time.append(ct[i]-0)

console.print(table)

print("Avg Waiting Time:", round(sum(waiting_time)/3, 2))
print("Avg TurnAround Time:", round(sum(turnaround_time)/3, 2))
```

Output:



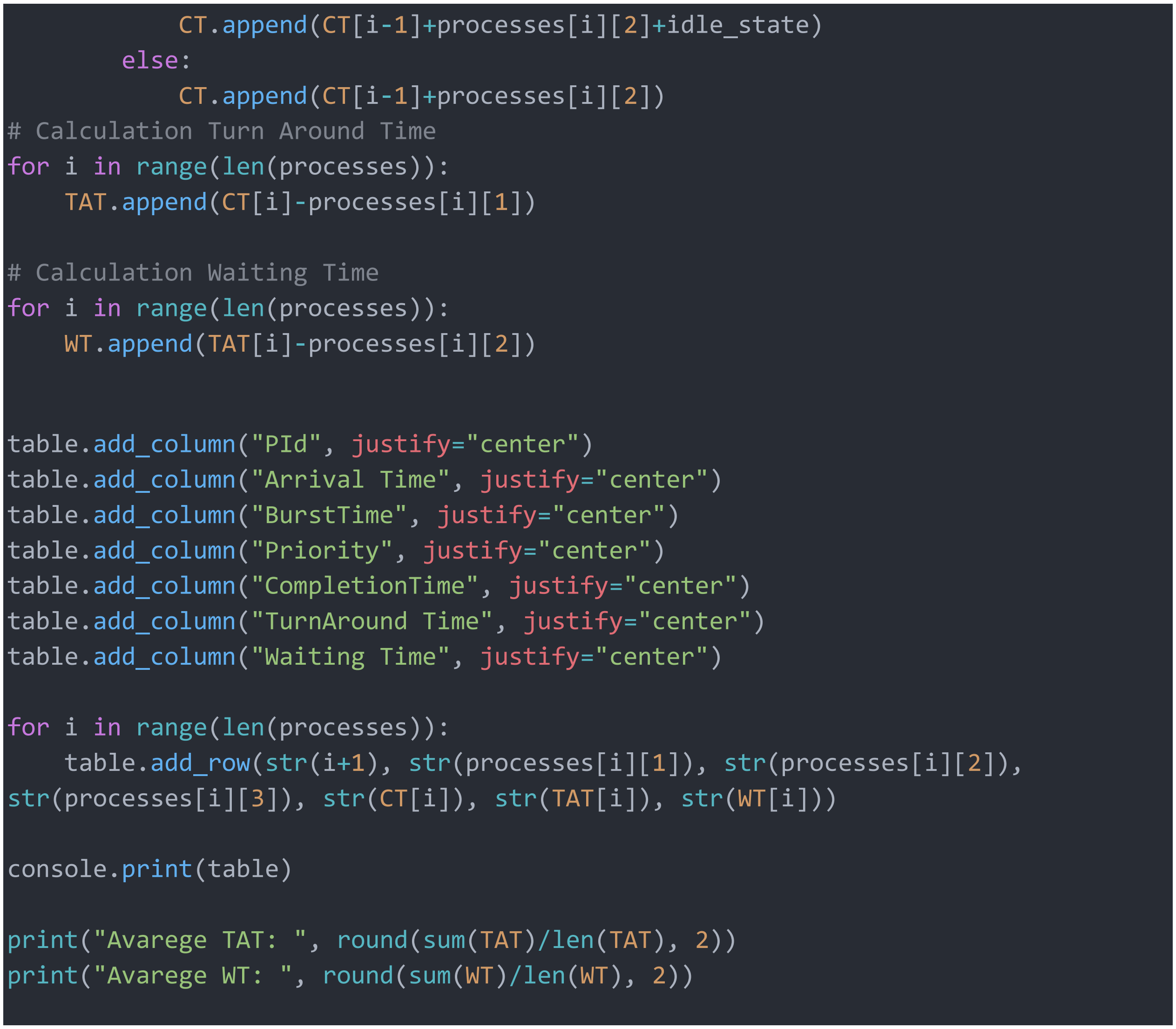
```
Enter the number of processes: 3
Enter the burst time for P[1]: 24
Enter the burst time for P[2]: 3
Enter the burst time for P[3]: 3
┏━━━━━┳━━━━━━━━━━━━━━┳━━━━━━━━━━━┳━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━┓
┃ PId ┃ Arrival Time ┃ BurstTime ┃ CompletionTime ┃ TurnAround Time ┃ Waiting Time ┃
┡━━━━━╇━━━━━━━━━━━━━━╇━━━━━━━━━━━╇━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━┩
│  1  │      0       │    24     │       30       │       30        │      6       │
│  2  │      0       │     3     │       7        │        7        │      4       │
│  3  │      0       │     3     │       10       │       10        │      7       │
└─────┴──────────────┴───────────┴────────────────┴─────────────────┴──────────────┘
Avg Waiting Time: 5.67
Avg TurnAround Time: 15.67
PS G:\Other computers\My Laptop\OS\Labs\Lab#09> 
```

Muhammad Waleed
20b-115-se
SE-B
OS Lab#09
Ma’am Shabina Mushtaq

Priority Algorithm:

```
import os

try:
    from rich.console import Console
    from rich.table import Table
except ImportError:
    os.system("pip install rich")
    from rich.console import Console
    from rich.table import Table

console = Console()
table = Table(show_header=True, header_style="bold magenta")

os.system("cls")
n = int(input("Enter the number of processes: "))
processes = []
CT = []
TAT = []
WT = []
for i in range(n):
    b = int(input("Burst Time: "))
    pr = int(input("Priority no: "))
    processes.append(["P"+str(i+1), 0, b, pr])

# sort According to prioriry
processes.sort(key=lambda x: x[3])
# Calculting Completion time
for i in range(len(processes)):
    if i == 0:
        if processes[i][1] > 0:
            state_idle = processes[i][1]
            CT.append(processes[i][2]+state_idle)
        else:
            CT.append(processes[i][2])
    else:
        if CT[i-1] < processes[i][1]:
            idle_state = processes[i][1] - CT[i-1]
```

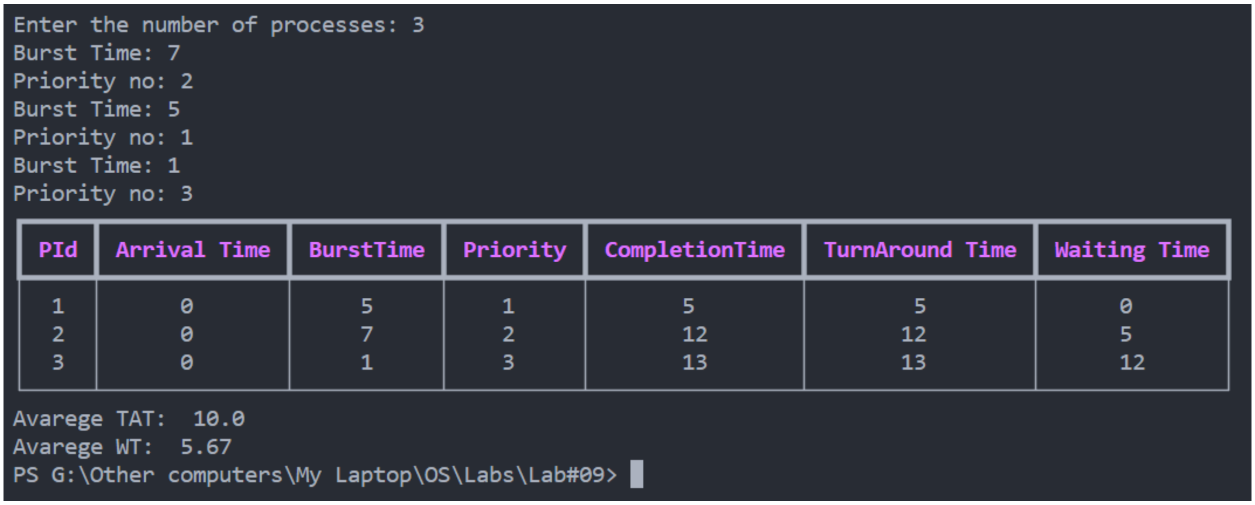
Muhammad Waleed
20b-115-se
SE-B
OS Lab#09
Ma’am Shabina Mushtaq



```
            CT.append(CT[i-1]+processes[i][2]+idle_state)
        else:
            CT.append(CT[i-1]+processes[i][2])
# Calculation Turn Around Time
for i in range(len(processes)):
    TAT.append(CT[i]-processes[i][1])

# Calculation Waiting Time
for i in range(len(processes)):
    WT.append(TAT[i]-processes[i][2])


table.add_column("PId", justify="center")
table.add_column("Arrival Time", justify="center")
table.add_column("BurstTime", justify="center")
table.add_column("Priority", justify="center")
table.add_column("CompletionTime", justify="center")
table.add_column("TurnAround Time", justify="center")
table.add_column("Waiting Time", justify="center")

for i in range(len(processes)):
    table.add_row(str(i+1), str(processes[i][1]), str(processes[i][2]),
str(processes[i][3]), str(CT[i]), str(TAT[i]), str(WT[i]))

console.print(table)

print("Avarege TAT: ", round(sum(TAT)/len(TAT), 2))
print("Avarege WT: ", round(sum(WT)/len(WT), 2))
```

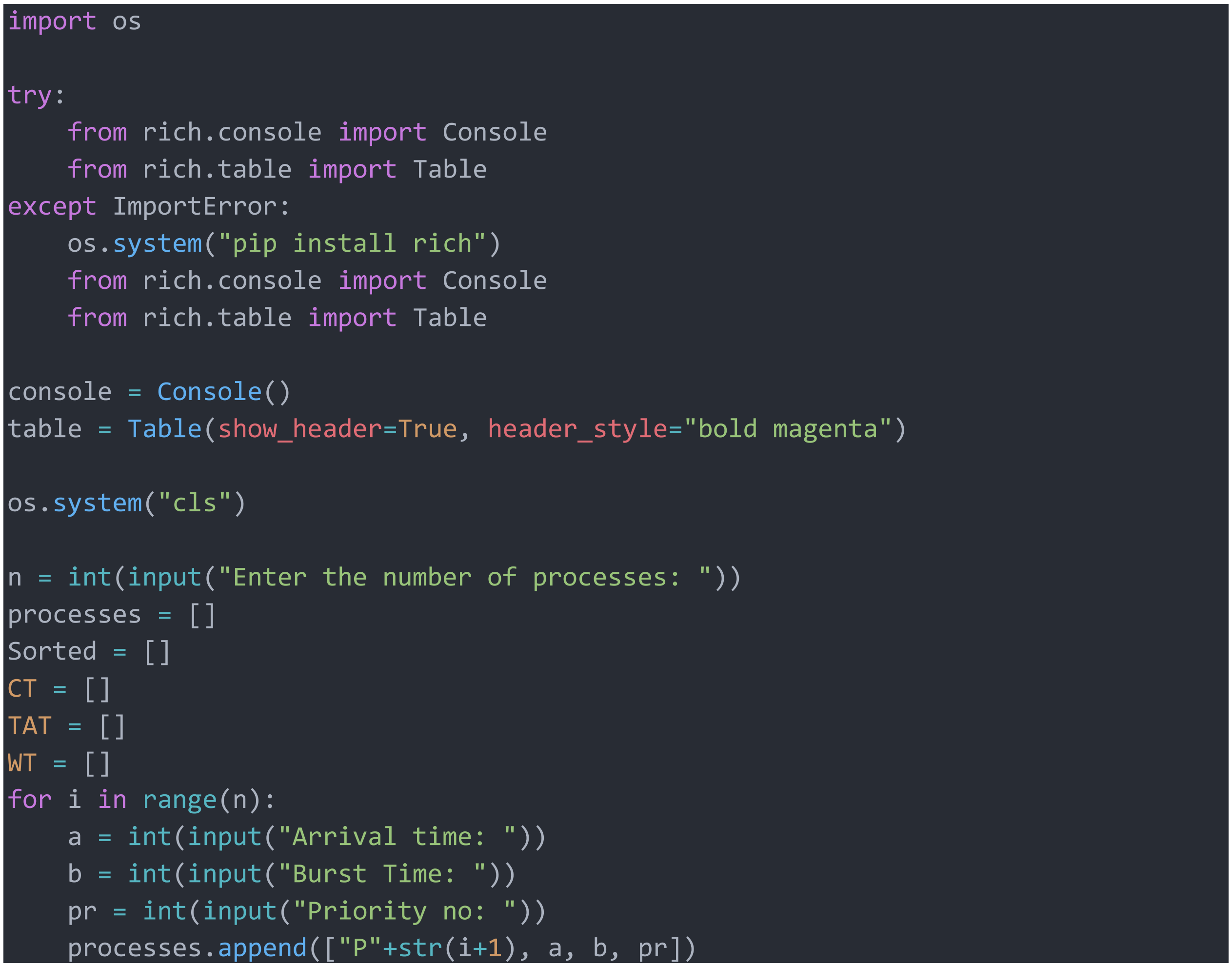
Output:

Muhammad Waleed
20b-115-se
SE-B
OS Lab#09
Ma’am Shabina Mushtaq



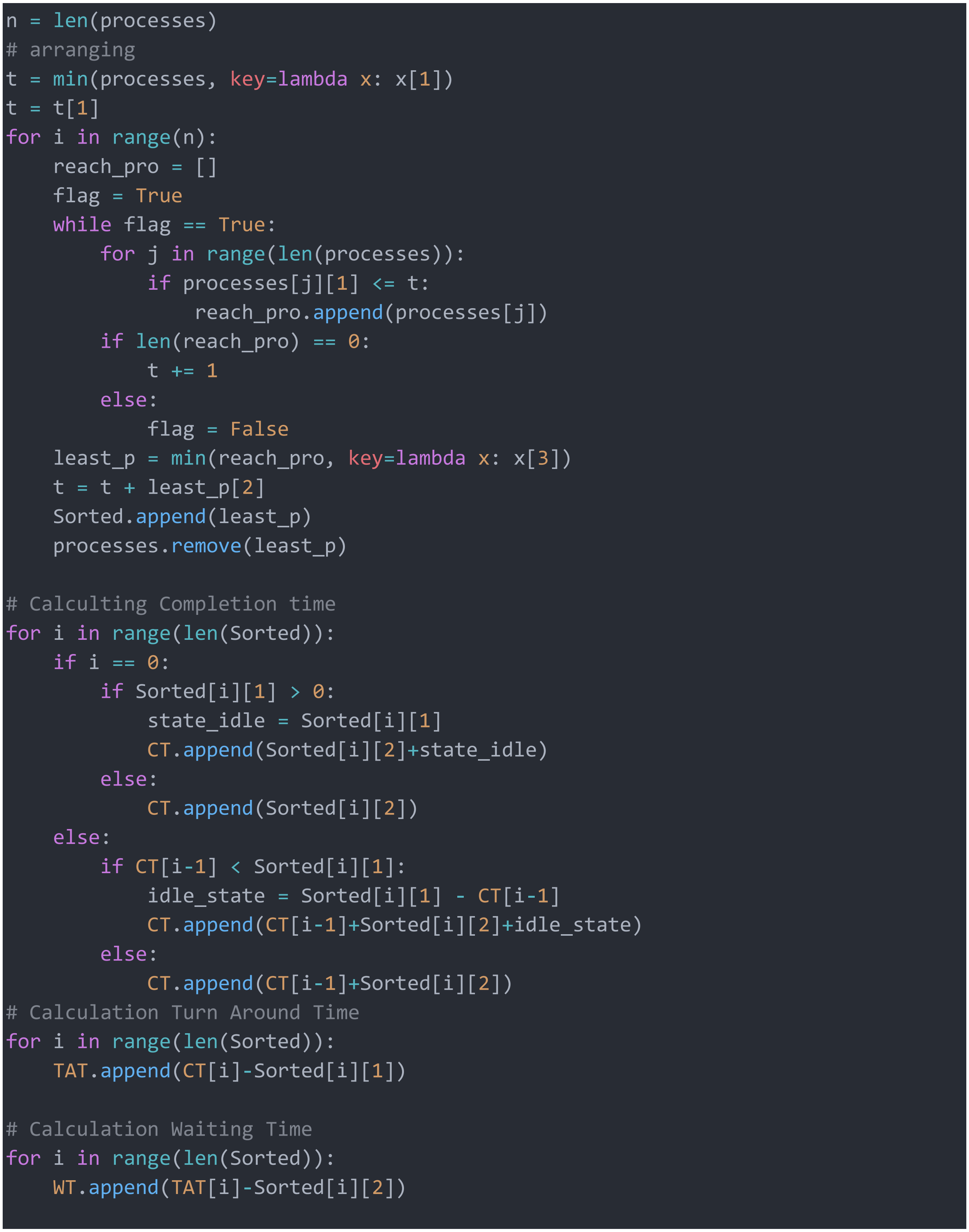
```
Enter the number of processes: 3
Burst Time: 7
Priority no: 2
Burst Time: 5
Priority no: 1
Burst Time: 1
Priority no: 3
┏━━━━━┳━━━━━━━━━━━━━━┳━━━━━━━━━━━┳━━━━━━━━━━┳━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━┓
┃ PId ┃ Arrival Time ┃ BurstTime ┃ Priority ┃ CompletionTime ┃ TurnAround Time ┃ Waiting Time ┃
┡━━━━━╇━━━━━━━━━━━━━━╇━━━━━━━━━━━╇━━━━━━━━━━╇━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━┩
│  1  │      0       │     5     │    1     │       5        │        5        │      0       │
│  2  │      0       │     7     │    2     │       12       │       12        │      5       │
│  3  │      0       │     1     │    3     │       13       │       13        │      12      │
└─────┴──────────────┴───────────┴──────────┴────────────────┴─────────────────┴──────────────┘
Avarege TAT:  10.0
Avarege WT:  5.67
PS G:\Other computers\My Laptop\OS\Labs\Lab#09> █
```

Priority Algorithm (with different arrival time):



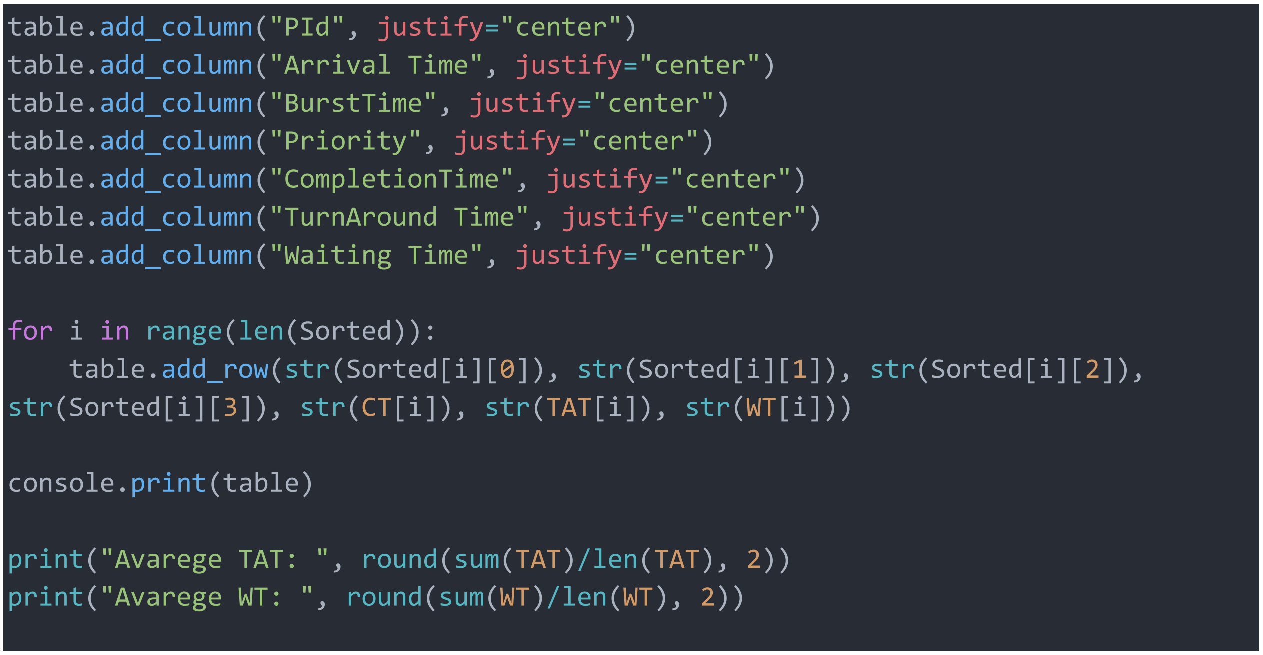
```python
import os

try:
    from rich.console import Console
    from rich.table import Table
except ImportError:
    os.system("pip install rich")
    from rich.console import Console
    from rich.table import Table

console = Console()
table = Table(show_header=True, header_style="bold magenta")

os.system("cls")

n = int(input("Enter the number of processes: "))
processes = []
Sorted = []
CT = []
TAT = []
WT = []
for i in range(n):
    a = int(input("Arrival time: "))
    b = int(input("Burst Time: "))
    pr = int(input("Priority no: "))
    processes.append(["P"+str(i+1), a, b, pr])
```

Muhammad Waleed
20b-115-se
SE-B
OS Lab#09
Ma’am Shabina Mushtaq



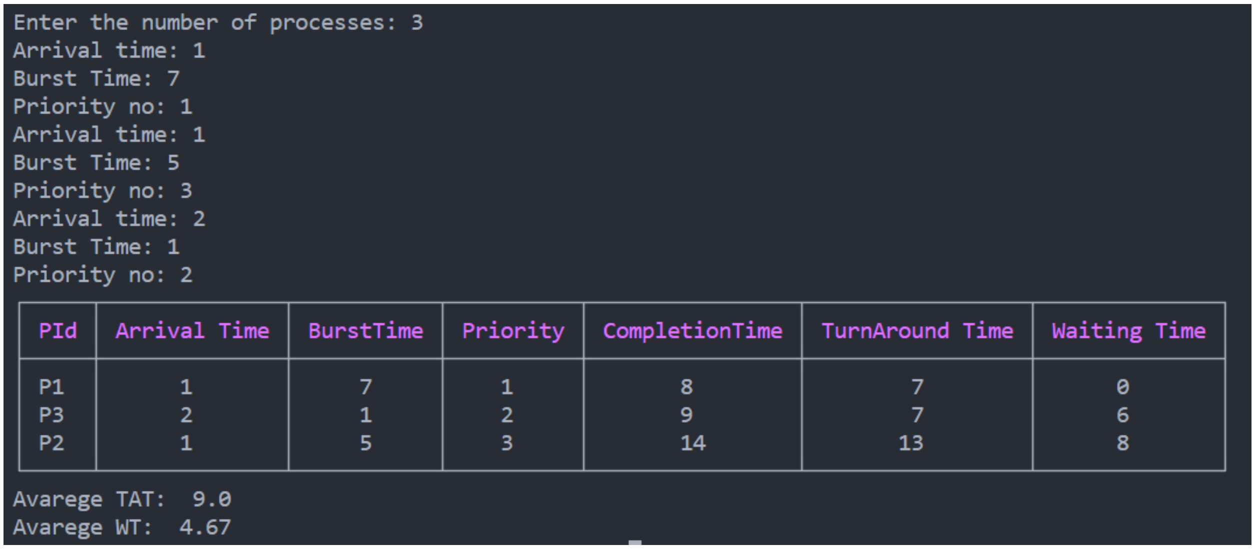
```
n = len(processes)
# arranging
t = min(processes, key=lambda x: x[1])
t = t[1]
for i in range(n):
    reach_pro = []
    flag = True
    while flag == True:
        for j in range(len(processes)):
            if processes[j][1] <= t:
                reach_pro.append(processes[j])
        if len(reach_pro) == 0:
            t += 1
        else:
            flag = False
    least_p = min(reach_pro, key=lambda x: x[3])
    t = t + least_p[2]
    Sorted.append(least_p)
    processes.remove(least_p)

# Calculting Completion time
for i in range(len(Sorted)):
    if i == 0:
        if Sorted[i][1] > 0:
            state_idle = Sorted[i][1]
            CT.append(Sorted[i][2]+state_idle)
        else:
            CT.append(Sorted[i][2])
    else:
        if CT[i-1] < Sorted[i][1]:
            idle_state = Sorted[i][1] - CT[i-1]
            CT.append(CT[i-1]+Sorted[i][2]+idle_state)
        else:
            CT.append(CT[i-1]+Sorted[i][2])
# Calculation Turn Around Time
for i in range(len(Sorted)):
    TAT.append(CT[i]-Sorted[i][1])

# Calculation Waiting Time
for i in range(len(Sorted)):
    WT.append(TAT[i]-Sorted[i][2])
```

Muhammad Waleed
20b-115-se
SE-B
OS Lab#09
Ma’am Shabina Mushtaq



```
table.add_column("PId", justify="center")
table.add_column("Arrival Time", justify="center")
table.add_column("BurstTime", justify="center")
table.add_column("Priority", justify="center")
table.add_column("CompletionTime", justify="center")
table.add_column("TurnAround Time", justify="center")
table.add_column("Waiting Time", justify="center")

for i in range(len(Sorted)):
    table.add_row(str(Sorted[i][0]), str(Sorted[i][1]), str(Sorted[i][2]),
str(Sorted[i][3]), str(CT[i]), str(TAT[i]), str(WT[i]))

console.print(table)

print("Avarege TAT: ", round(sum(TAT)/len(TAT), 2))
print("Avarege WT: ", round(sum(WT)/len(WT), 2))
```

Output:



```
Enter the number of processes: 3
Arrival time: 1
Burst Time: 7
Priority no: 1
Arrival time: 1
Burst Time: 5
Priority no: 3
Arrival time: 2
Burst Time: 1
Priority no: 2
┌─────┬──────────────┬───────────┬──────────┬────────────────┬─────────────────┬──────────────┐
│ PId │ Arrival Time │ BurstTime │ Priority │ CompletionTime │ TurnAround Time │ Waiting Time │
├─────┼──────────────┼───────────┼──────────┼────────────────┼─────────────────┼──────────────┤
│ P1  │      1       │     7     │    1     │       8        │        7        │      0       │
│ P3  │      2       │     1     │    2     │       9        │        7        │      6       │
│ P2  │      1       │     5     │    3     │       14       │       13        │      8       │
└─────┴──────────────┴───────────┴──────────┴────────────────┴─────────────────┴──────────────┘
Avarege TAT:  9.0
Avarege WT:  4.67
```

Usman Institute of Technology

Department of Computer Science Fall 2022

Name: Muhammad Waleed

Roll no: 20B-115-SE

Course: Operating Systems (CS312)

Course Instructor: Ma’am Shabina Mushtaq

Date: 22-Dec-2022

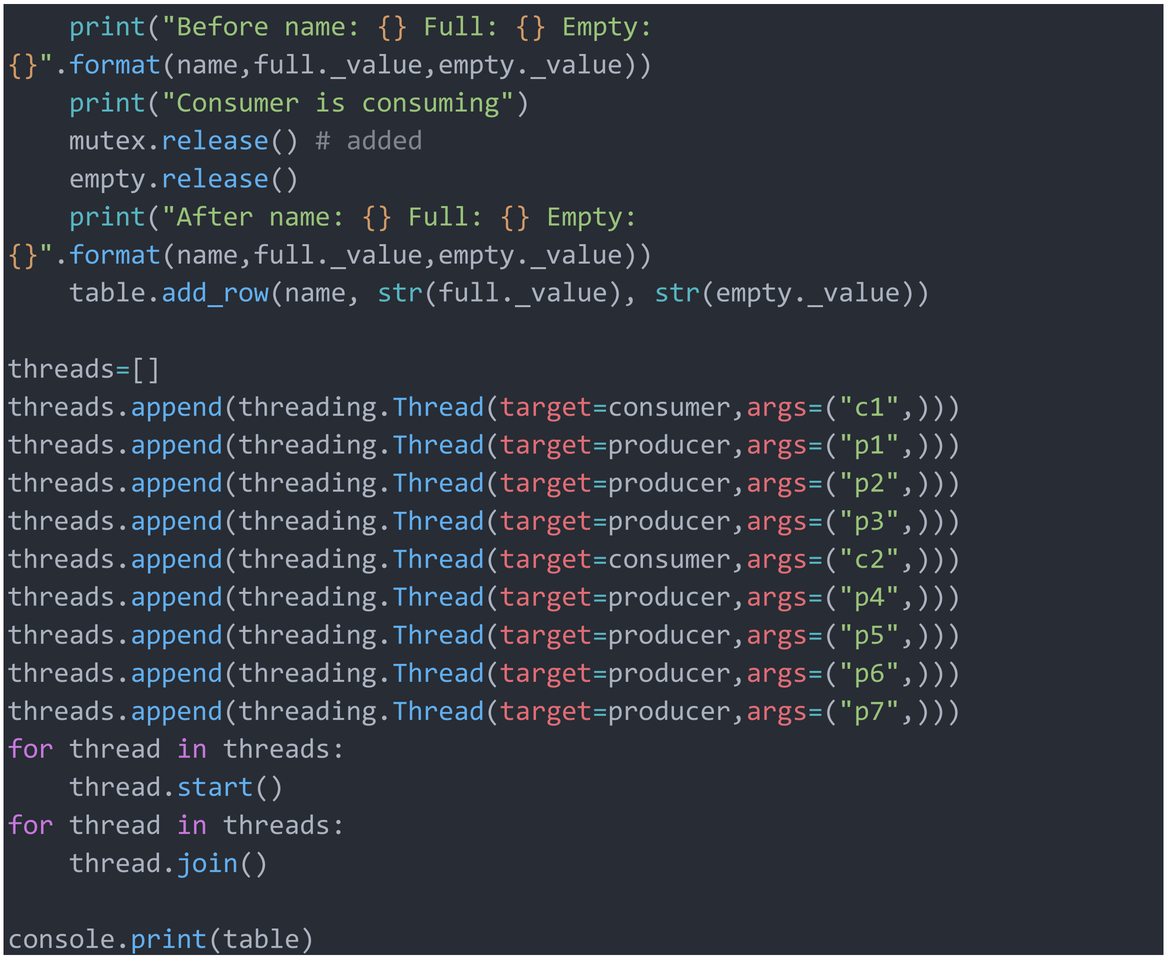
Muhammad Waleed
20b-115-se
SE-B
OS Lab#10
Ma’am Shabina Mushtaq

Lab Tasks:

1. Write a python program that demonstrates the synchronization of Consumer producer Bounded Buffer Problem using semaphores.

```python
import threading,os

try:
    from rich.console import Console
    from rich.table import Table
except ImportError:
    os.system("pip install rich")
    from rich.console import Console
    from rich.table import Table

console = Console()
table = Table(show_header=True, header_style="bold magenta")

buf = []
empty = threading.Semaphore(5)
full = threading.Semaphore(0)
mutex = threading.Lock()

table.add_column("Name", style="dim", width=12)
table.add_column("Full", style="dim", width=12)
table.add_column("Empty", style="dim", width=12)


def producer(name):
    empty.acquire()
    mutex.acquire() # added
    print("Before name: {} Full: {} Empty:
{}".format(name,full._value,empty._value))
    print("Producer is producing")
    mutex.release() # added
    full.release()
    print("After name: {} Full: {} Empty:
{}".format(name,full._value,empty._value))
    table.add_row(name, str(full._value), str(empty._value))

def consumer(name):
    full.acquire()
    mutex.acquire() # added
```

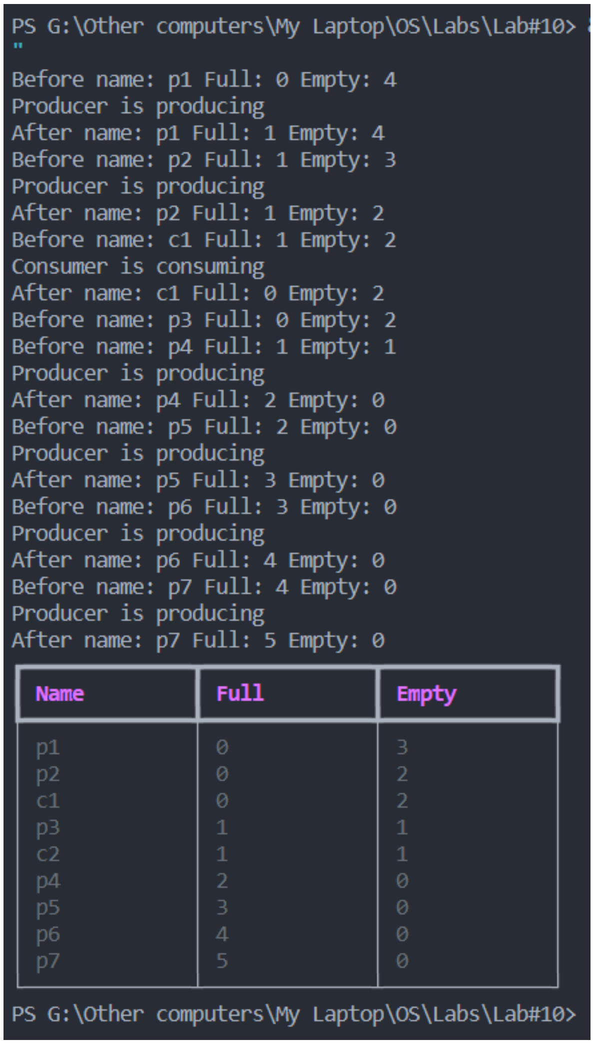
Muhammad Waleed
20b-115-se
SE-B
OS Lab#10
Ma’am Shabina Mushtaq



```
    print("Before name: {} Full: {} Empty:
{}".format(name,full._value,empty._value))
    print("Consumer is consuming")
    mutex.release() # added
    empty.release()
    print("After name: {} Full: {} Empty:
{}".format(name,full._value,empty._value))
    table.add_row(name, str(full._value), str(empty._value))

threads=[]
threads.append(threading.Thread(target=consumer,args=("c1",)))
threads.append(threading.Thread(target=producer,args=("p1",)))
threads.append(threading.Thread(target=producer,args=("p2",)))
threads.append(threading.Thread(target=producer,args=("p3",)))
threads.append(threading.Thread(target=consumer,args=("c2",)))
threads.append(threading.Thread(target=producer,args=("p4",)))
threads.append(threading.Thread(target=producer,args=("p5",)))
threads.append(threading.Thread(target=producer,args=("p6",)))
threads.append(threading.Thread(target=producer,args=("p7",)))
for thread in threads:
    thread.start()
for thread in threads:
    thread.join()

console.print(table)
```

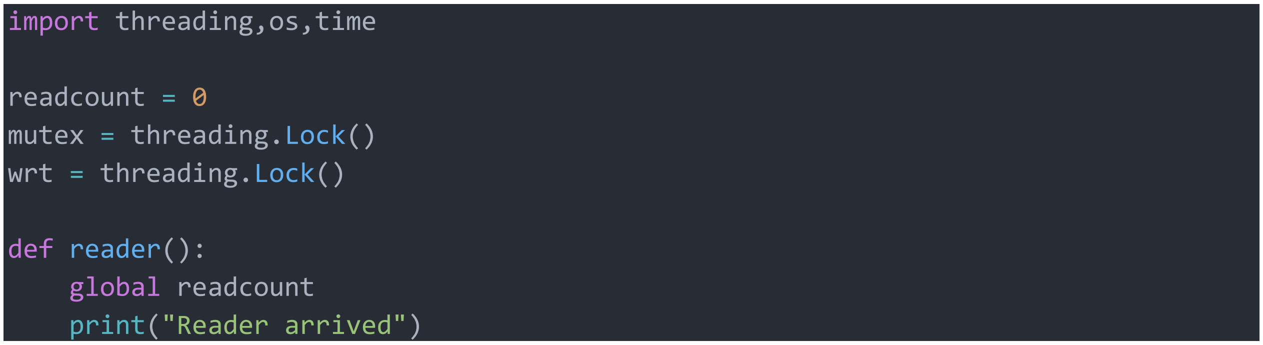
Muhammad Waleed
20b-115-se
SE-B
OS Lab#10
Ma'am Shabina Mushtaq

Output:



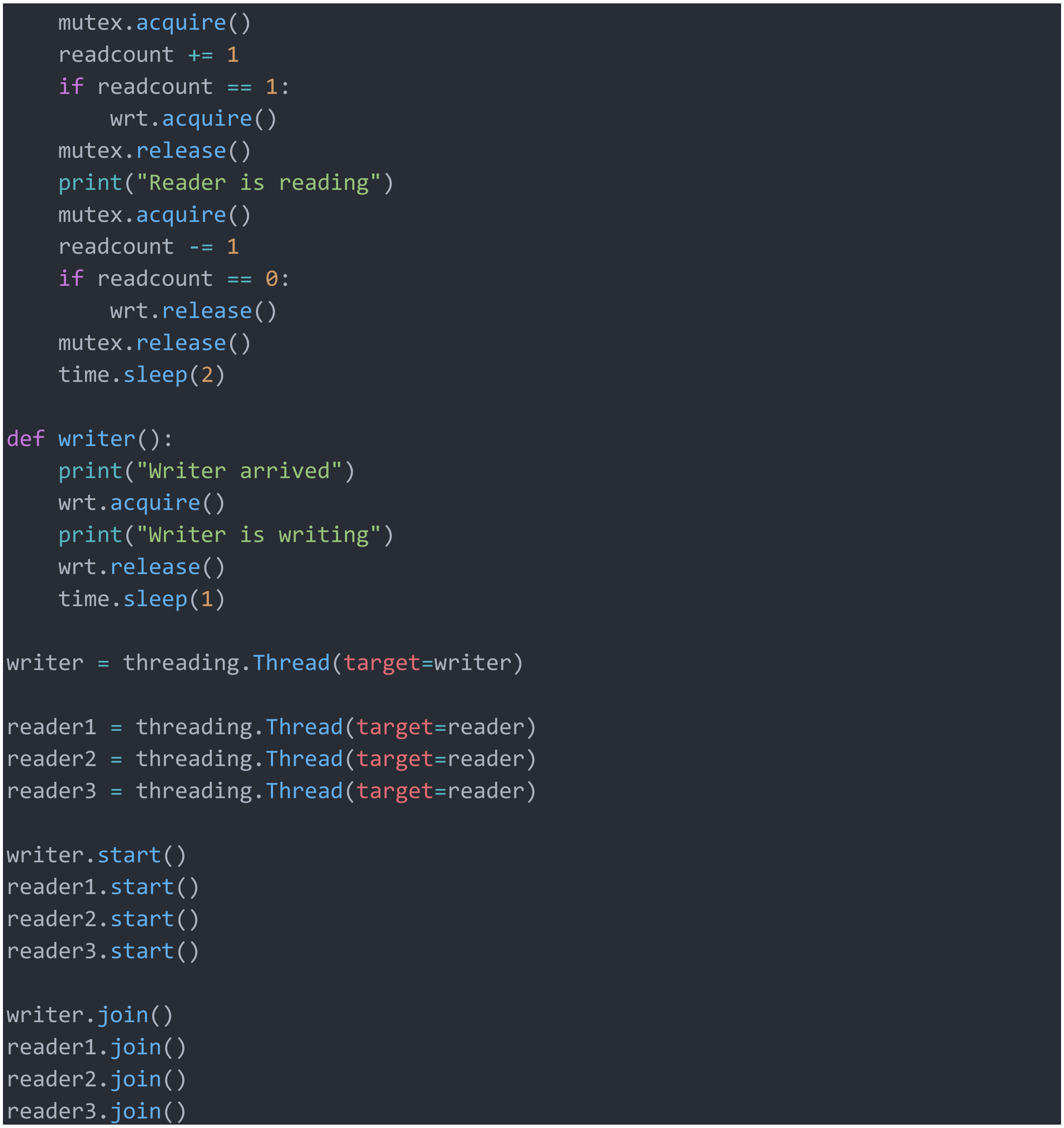
```
PS G:\Other computers\My Laptop\OS\Labs\Lab#10>
"

Before name: p1 Full: 0 Empty: 4
Producer is producing
After name: p1 Full: 1 Empty: 4
Before name: p2 Full: 1 Empty: 3
Producer is producing
After name: p2 Full: 1 Empty: 2
Before name: c1 Full: 1 Empty: 2
Consumer is consuming
After name: c1 Full: 0 Empty: 2
Before name: p3 Full: 0 Empty: 2
Before name: p4 Full: 1 Empty: 1
Producer is producing
After name: p4 Full: 2 Empty: 0
Before name: p5 Full: 2 Empty: 0
Producer is producing
After name: p5 Full: 3 Empty: 0
Before name: p6 Full: 3 Empty: 0
Producer is producing
After name: p6 Full: 4 Empty: 0
Before name: p7 Full: 4 Empty: 0
Producer is producing
After name: p7 Full: 5 Empty: 0
┌──────────────┬──────────────┬──────────────┐
│ Name         │ Full         │ Empty        │
├──────────────┼──────────────┼──────────────┤
│ p1           │ 0            │ 3            │
│ p2           │ 0            │ 2            │
│ c1           │ 0            │ 2            │
│ p3           │ 1            │ 1            │
│ c2           │ 1            │ 1            │
│ p4           │ 2            │ 0            │
│ p5           │ 3            │ 0            │
│ p6           │ 4            │ 0            │
│ p7           │ 5            │ 0            │
└──────────────┴──────────────┴──────────────┘
PS G:\Other computers\My Laptop\OS\Labs\Lab#10>
```

2. Write a python program that demonstrates the synchronization of Readers and Writer Problem using semaphores.



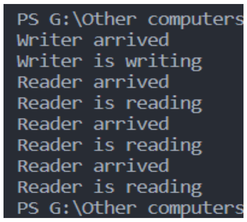
```python
import threading,os,time

readcount = 0
mutex = threading.Lock()
wrt = threading.Lock()

def reader():
    global readcount
    print("Reader arrived")
```

Muhammad Waleed
20b-115-se
SE-B
OS Lab#10
Ma’am Shabina Mushtaq



```
    mutex.acquire()
    readcount += 1
    if readcount == 1:
        wrt.acquire()
    mutex.release()
    print("Reader is reading")
    mutex.acquire()
    readcount -= 1
    if readcount == 0:
        wrt.release()
    mutex.release()
    time.sleep(2)

def writer():
    print("Writer arrived")
    wrt.acquire()
    print("Writer is writing")
    wrt.release()
    time.sleep(1)

writer = threading.Thread(target=writer)

reader1 = threading.Thread(target=reader)
reader2 = threading.Thread(target=reader)
reader3 = threading.Thread(target=reader)

writer.start()
reader1.start()
reader2.start()
reader3.start()

writer.join()
reader1.join()
reader2.join()
reader3.join()
```

Output:



```
PS G:\Other computers
Writer arrived
Writer is writing
Reader arrived
Reader is reading
Reader arrived
Reader is reading
Reader arrived
Reader is reading
PS G:\Other computers
```