Benha University
Benha Faculty of Engineering
Electrical Engineering Department

Implementation of SJF (Shortest Job First) Scheduling Algorithms (Non-Preemptive and Preemptive)

By

Student/ Kareem Gamal Essa Ahmed
Student/ Kareem Mohamed Abd-ElMoneam

4th year, Electrical Engineering Department, Communication and Computer

Supervised By:

Dr/ Tamer Omar

Benha 2025

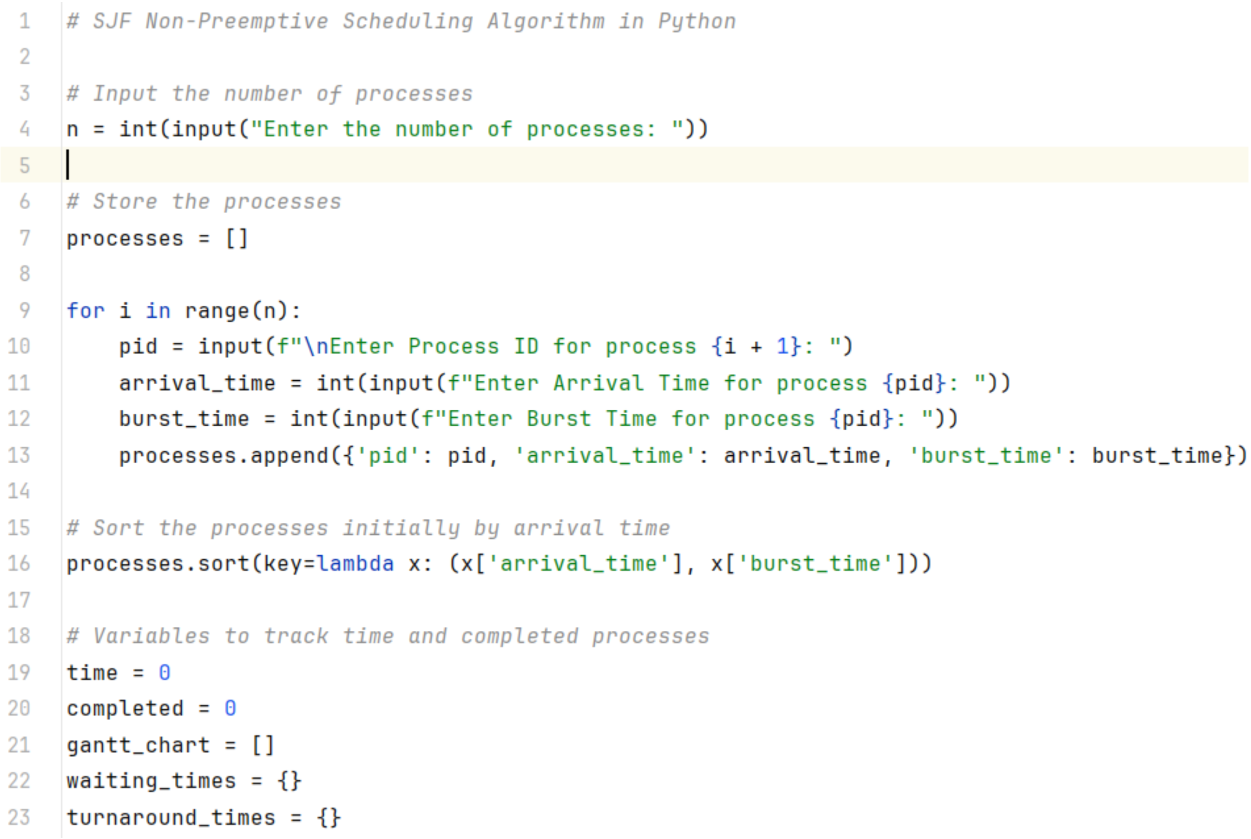
1. Introduction:

In this report, we implement and compare the two types of Shortest Job First (SJF) Scheduling Algorithms:

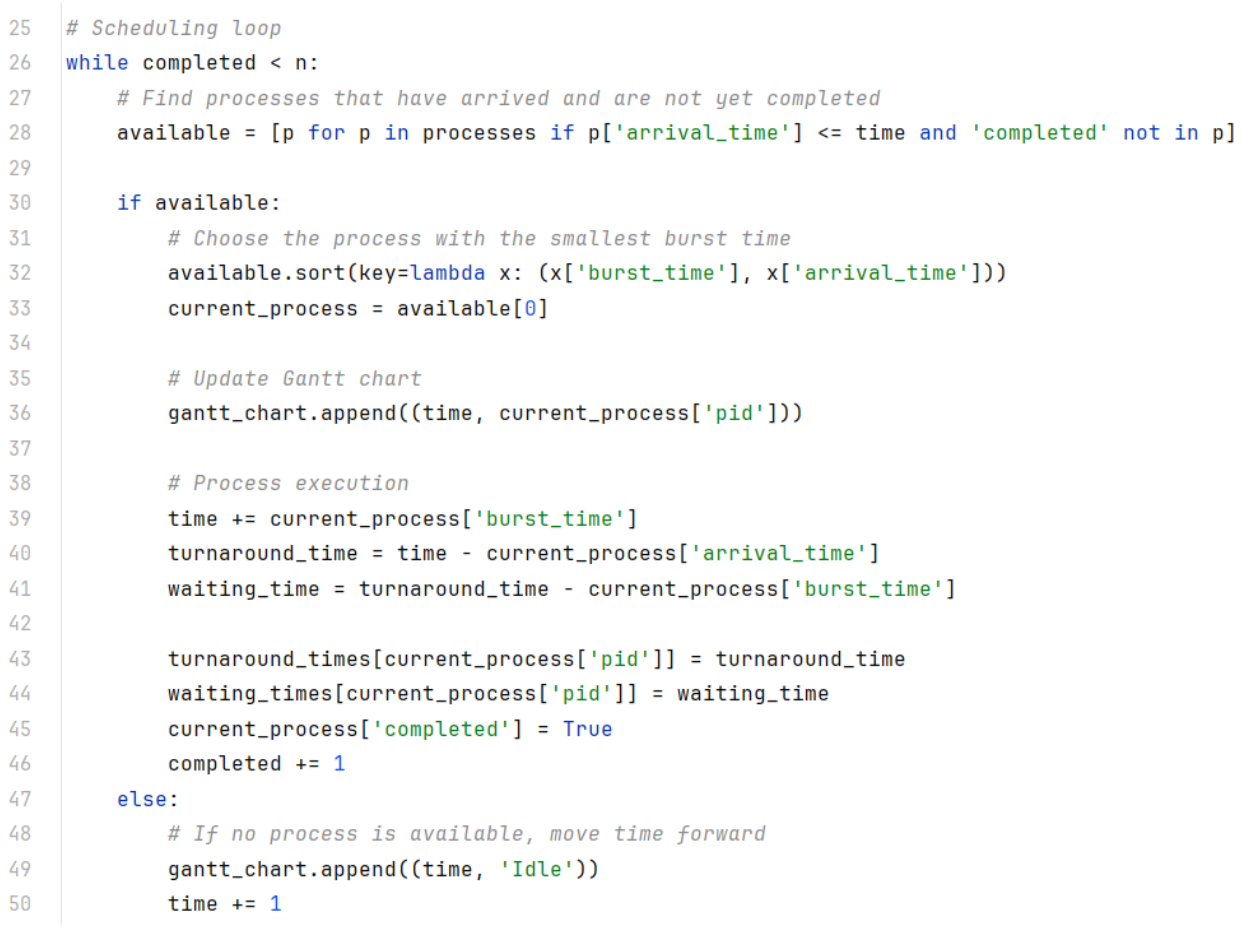
- **SJF Non-Preemptive**: Once a process starts, it cannot be interrupted until it finishes.
- **SJF Preemptive (SRTF)**: A process can be interrupted if a new process with a shorter remaining time arrives.

2. Code Implementation:

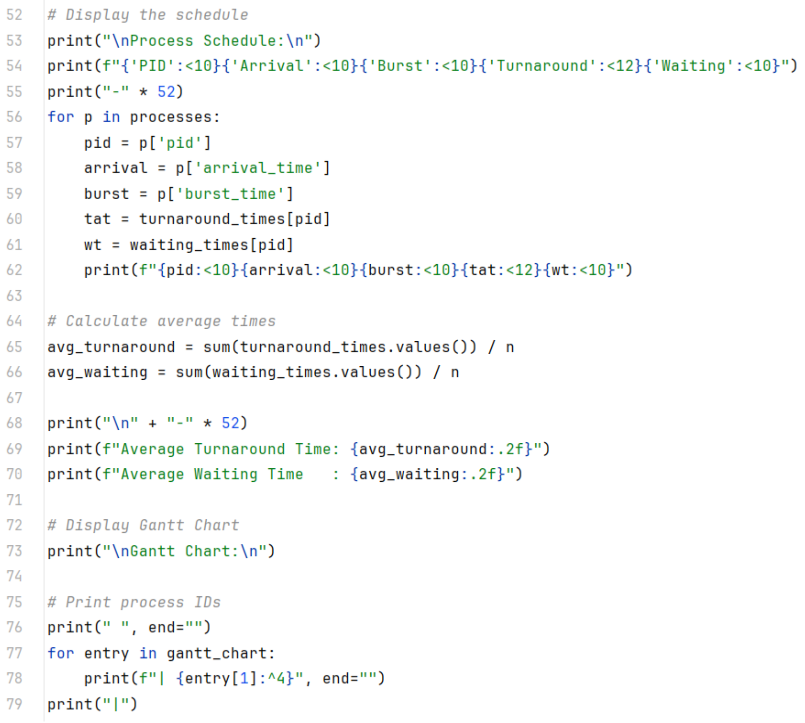
2.1 SJF Non-Preemptive Code:



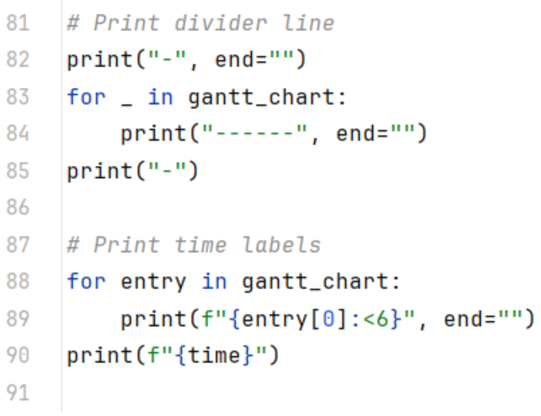
```python
# SJF Non-Preemptive Scheduling Algorithm in Python

# Input the number of processes
n = int(input("Enter the number of processes: "))

# Store the processes
processes = []

for i in range(n):
    pid = input(f"\nEnter Process ID for process {i + 1}: ")
    arrival_time = int(input(f"Enter Arrival Time for process {pid}: "))
    burst_time = int(input(f"Enter Burst Time for process {pid}: "))
    processes.append({'pid': pid, 'arrival_time': arrival_time, 'burst_time': burst_time})

# Sort the processes initially by arrival time
processes.sort(key=lambda x: (x['arrival_time'], x['burst_time']))

# Variables to track time and completed processes
time = 0
completed = 0
gantt_chart = []
waiting_times = {}
turnaround_times = {}
```



```
# Scheduling loop
while completed < n:
    # Find processes that have arrived and are not yet completed
    available = [p for p in processes if p['arrival_time'] <= time and 'completed' not in p]

    if available:
        # Choose the process with the smallest burst time
        available.sort(key=lambda x: (x['burst_time'], x['arrival_time']))
        current_process = available[0]

        # Update Gantt chart
        gantt_chart.append((time, current_process['pid']))

        # Process execution
        time += current_process['burst_time']
        turnaround_time = time - current_process['arrival_time']
        waiting_time = turnaround_time - current_process['burst_time']

        turnaround_times[current_process['pid']] = turnaround_time
        waiting_times[current_process['pid']] = waiting_time
        current_process['completed'] = True
        completed += 1
    else:
        # If no process is available, move time forward
        gantt_chart.append((time, 'Idle'))
        time += 1
```

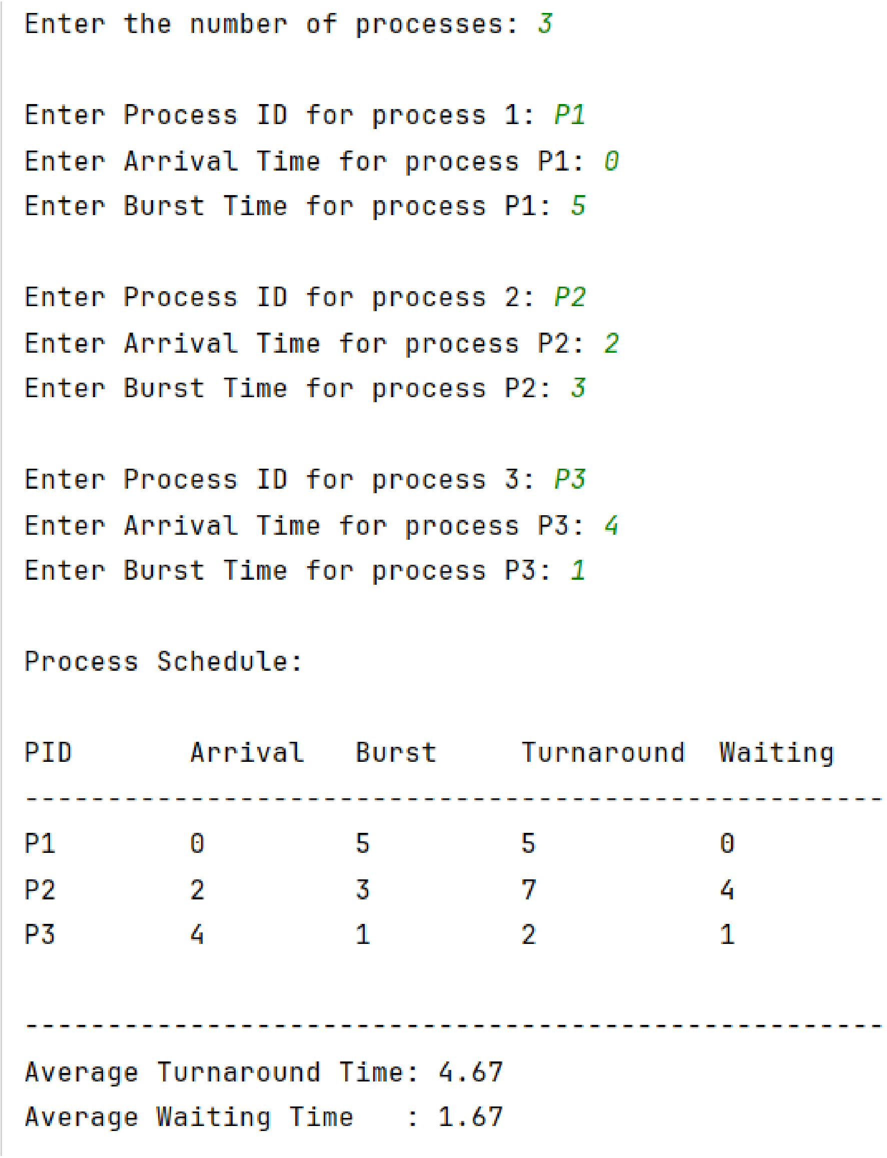


```
# Display the schedule
print("\nProcess Schedule:\n")
print(f"{'PID':<10}{'Arrival':<10}{'Burst':<10}{'Turnaround':<12}{'Waiting':<10}")
print("-" * 52)
for p in processes:
    pid = p['pid']
    arrival = p['arrival_time']
    burst = p['burst_time']
    tat = turnaround_times[pid]
    wt = waiting_times[pid]
    print(f"{pid:<10}{arrival:<10}{burst:<10}{tat:<12}{wt:<10}")

# Calculate average times
avg_turnaround = sum(turnaround_times.values()) / n
avg_waiting = sum(waiting_times.values()) / n

print("\n" + "-" * 52)
print(f"Average Turnaround Time: {avg_turnaround:.2f}")
print(f"Average Waiting Time   : {avg_waiting:.2f}")

# Display Gantt Chart
print("\nGantt Chart:\n")

# Print process IDs
print(" ", end="")
for entry in gantt_chart:
    print(f"| {entry[1]:^4}", end="")
print("|")
```

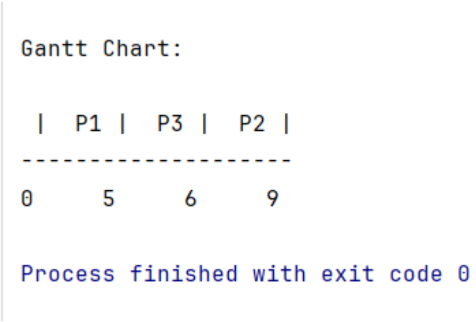


```
# Print divider line
print("-", end="")
for _ in gantt_chart:
    print("------", end="")
print("-")

# Print time labels
for entry in gantt_chart:
    print(f"{entry[0]:<6}", end="")
print(f"{time}")
```

2.2 Expected Output for SJF Non-Preemptive:

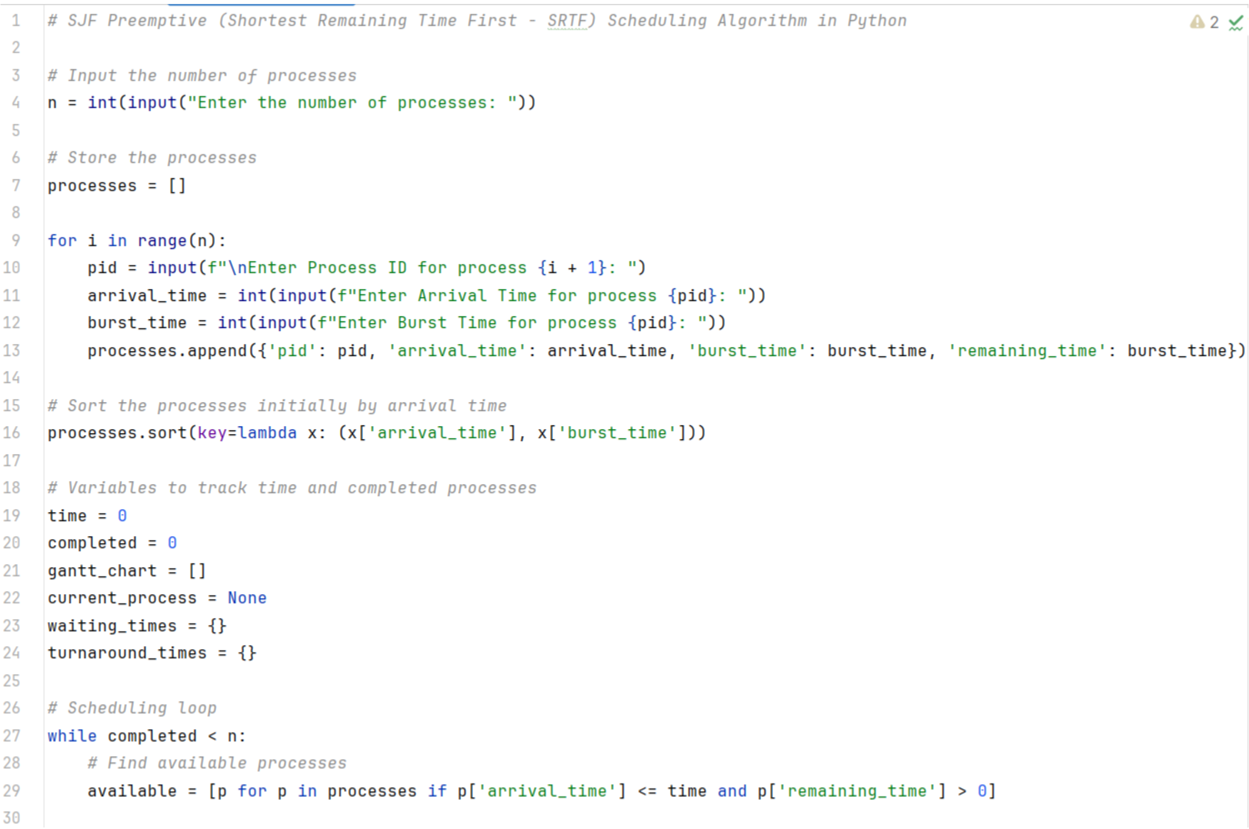


```
Enter the number of processes: 3

Enter Process ID for process 1: P1
Enter Arrival Time for process P1: 0
Enter Burst Time for process P1: 5

Enter Process ID for process 2: P2
Enter Arrival Time for process P2: 2
Enter Burst Time for process P2: 3

Enter Process ID for process 3: P3
Enter Arrival Time for process P3: 4
Enter Burst Time for process P3: 1

Process Schedule:

PID       Arrival   Burst     Turnaround  Waiting
----------------------------------------------------
P1        0         5         5           0
P2        2         3         7           4
P3        4         1         2           1

----------------------------------------------------
Average Turnaround Time: 4.67
Average Waiting Time   : 1.67
```

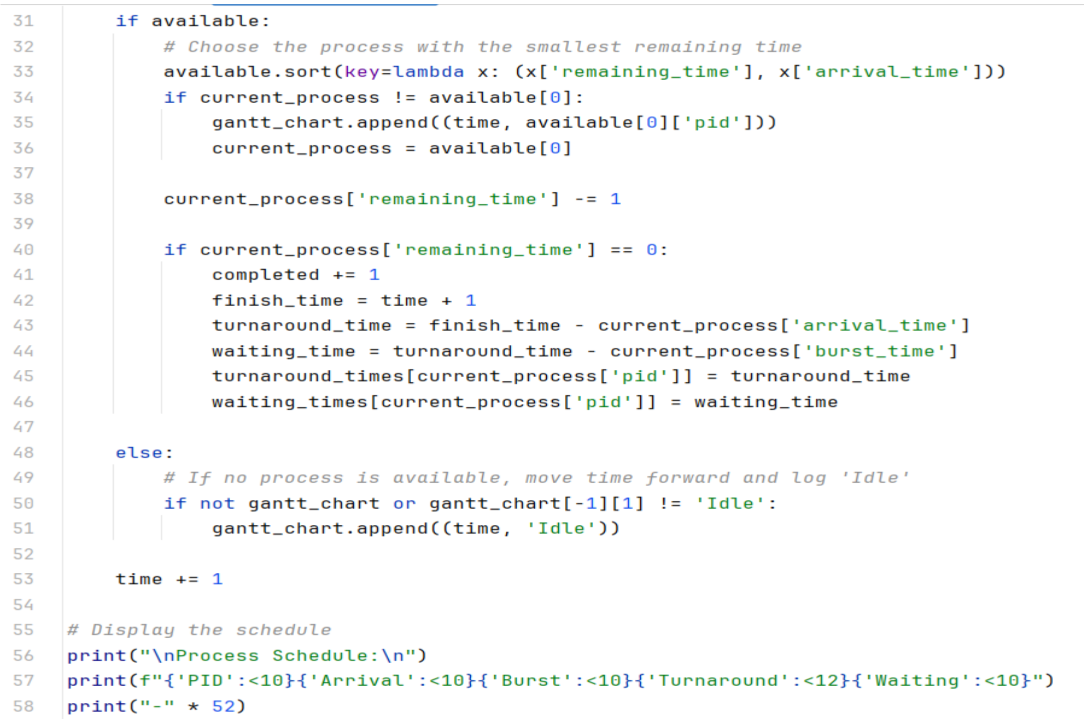


```
Gantt Chart:

 |  P1 |  P3 |  P2 |
--------------------
0     5     6     9

Process finished with exit code 0
```

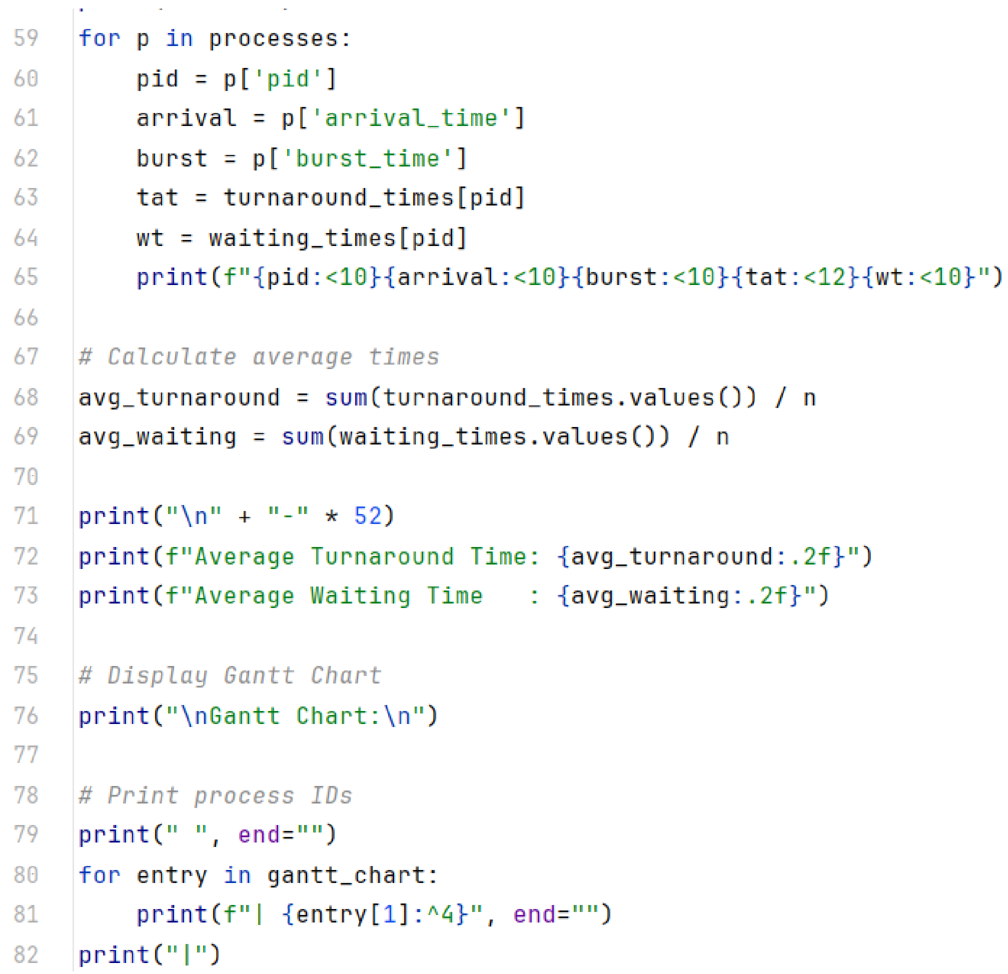
3.1 SJF Preemptive (SRTF) Code:



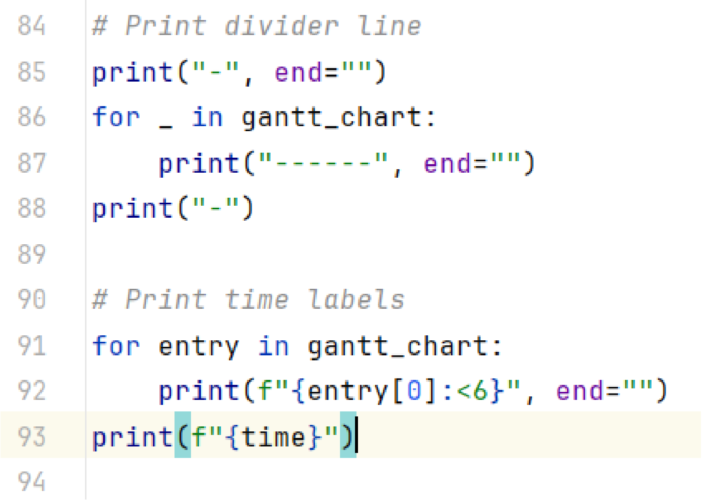
```python
# SJF Preemptive (Shortest Remaining Time First - SRTF) Scheduling Algorithm in Python

# Input the number of processes
n = int(input("Enter the number of processes: "))

# Store the processes
processes = []

for i in range(n):
    pid = input(f"\nEnter Process ID for process {i + 1}: ")
    arrival_time = int(input(f"Enter Arrival Time for process {pid}: "))
    burst_time = int(input(f"Enter Burst Time for process {pid}: "))
    processes.append({'pid': pid, 'arrival_time': arrival_time, 'burst_time': burst_time, 'remaining_time': burst_time})

# Sort the processes initially by arrival time
processes.sort(key=lambda x: (x['arrival_time'], x['burst_time']))

# Variables to track time and completed processes
time = 0
completed = 0
gantt_chart = []
current_process = None
waiting_times = {}
turnaround_times = {}

# Scheduling loop
while completed < n:
    # Find available processes
    available = [p for p in processes if p['arrival_time'] <= time and p['remaining_time'] > 0]

```



```python
    if available:
        # Choose the process with the smallest remaining time
        available.sort(key=lambda x: (x['remaining_time'], x['arrival_time']))
        if current_process != available[0]:
            gantt_chart.append((time, available[0]['pid']))
            current_process = available[0]

        current_process['remaining_time'] -= 1

        if current_process['remaining_time'] == 0:
            completed += 1
            finish_time = time + 1
            turnaround_time = finish_time - current_process['arrival_time']
            waiting_time = turnaround_time - current_process['burst_time']
            turnaround_times[current_process['pid']] = turnaround_time
            waiting_times[current_process['pid']] = waiting_time

    else:
        # If no process is available, move time forward and log 'Idle'
        if not gantt_chart or gantt_chart[-1][1] != 'Idle':
            gantt_chart.append((time, 'Idle'))

    time += 1

# Display the schedule
print("\nProcess Schedule:\n")
print(f"{'PID':<10}{'Arrival':<10}{'Burst':<10}{'Turnaround':<12}{'Waiting':<10}")
print("-" * 52)
```

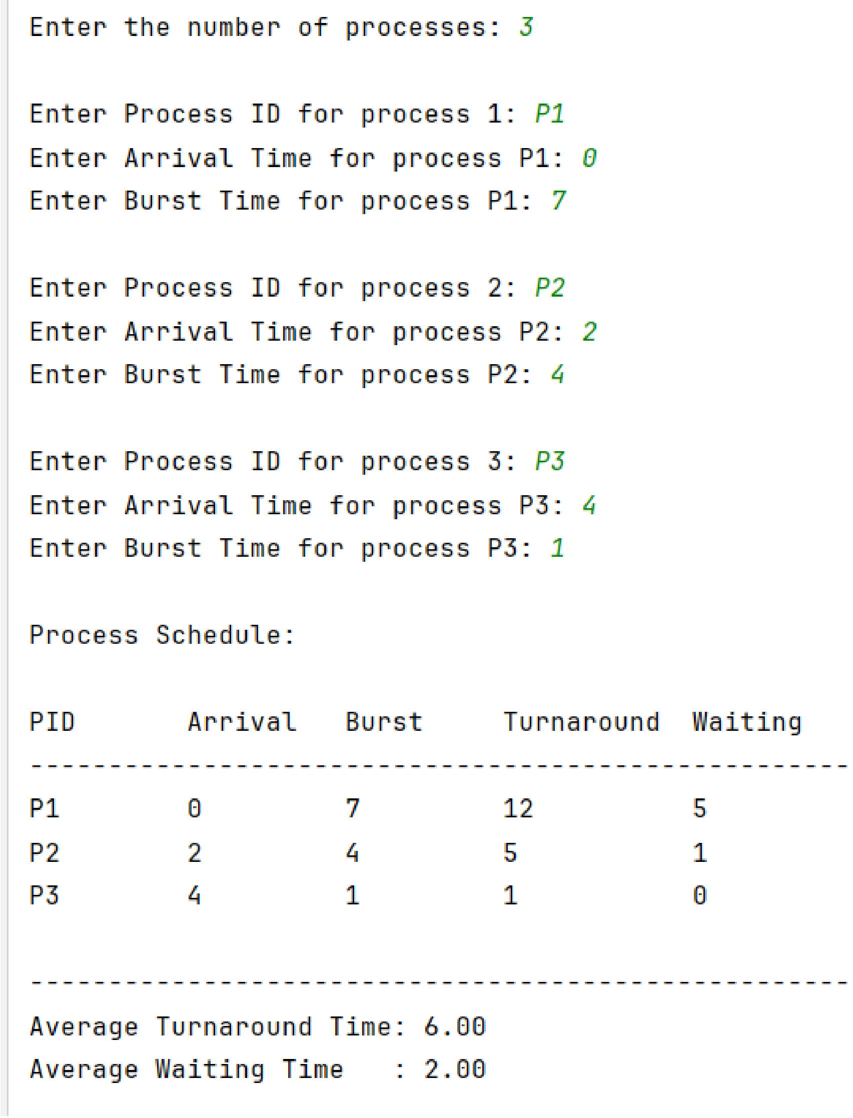


```python
for p in processes:
    pid = p['pid']
    arrival = p['arrival_time']
    burst = p['burst_time']
    tat = turnaround_times[pid]
    wt = waiting_times[pid]
    print(f"{pid:<10}{arrival:<10}{burst:<10}{tat:<12}{wt:<10}")

# Calculate average times
avg_turnaround = sum(turnaround_times.values()) / n
avg_waiting = sum(waiting_times.values()) / n

print("\n" + "-" * 52)
print(f"Average Turnaround Time: {avg_turnaround:.2f}")
print(f"Average Waiting Time    : {avg_waiting:.2f}")

# Display Gantt Chart
print("\nGantt Chart:\n")

# Print process IDs
print(" ", end="")
for entry in gantt_chart:
    print(f"| {entry[1]:^4}", end="")
print("|")
```

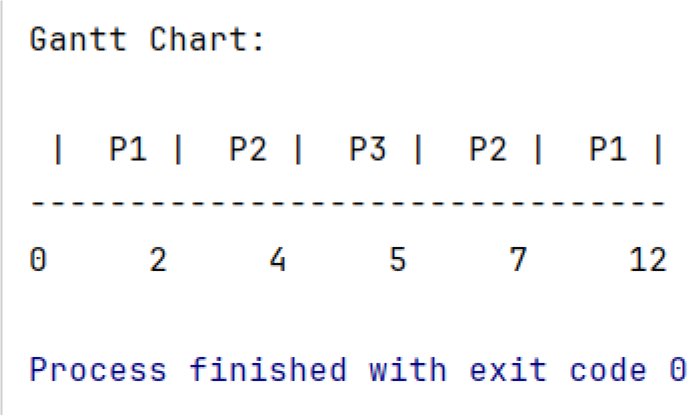


```python
# Print divider line
print("-", end="")
for _ in gantt_chart:
    print("------", end="")
print("-")

# Print time labels
for entry in gantt_chart:
    print(f"{entry[0]:<6}", end="")
print(f"{time}")
```

3.2 Expected Output for SJF Preemptive (SRTF):



```
Enter the number of processes: 3

Enter Process ID for process 1: P1
Enter Arrival Time for process P1: 0
Enter Burst Time for process P1: 7

Enter Process ID for process 2: P2
Enter Arrival Time for process P2: 2
Enter Burst Time for process P2: 4

Enter Process ID for process 3: P3
Enter Arrival Time for process P3: 4
Enter Burst Time for process P3: 1

Process Schedule:

PID       Arrival   Burst     Turnaround  Waiting
----------------------------------------------------
P1        0         7         12          5
P2        2         4         5           1
P3        4         1         1           0

----------------------------------------------------
Average Turnaround Time: 6.00
Average Waiting Time   : 2.00
```



```
Gantt Chart:

 |  P1 |  P2 |  P3 |  P2 |  P1 |
--------------------------------
0     2     4     5     7     12

Process finished with exit code 0
```

Comparison:

Criteria	SJF Non-Preemptive	SJF Preemptive (SRTF)
Complexity	Simple	More Complex
Context Switching	Low	High
Waiting Time (average)	Higher	Lower
Turnaround Time (average)	Higher	Lower
Fairness for New Process	Low	High
Implementation	Easier	Harder

Conclusion:

- SJF Preemptive (SRTF) gives better average times but at the cost of higher complexity and more context switching.
- SJF Non-Preemptive is easier to implement but may not be efficient for systems requiring dynamic scheduling.