Experiment No : 03
Date of Experiment : 20 / 04 / 2025
Experiment Name : Shortest Job First (SJF) Scheduling

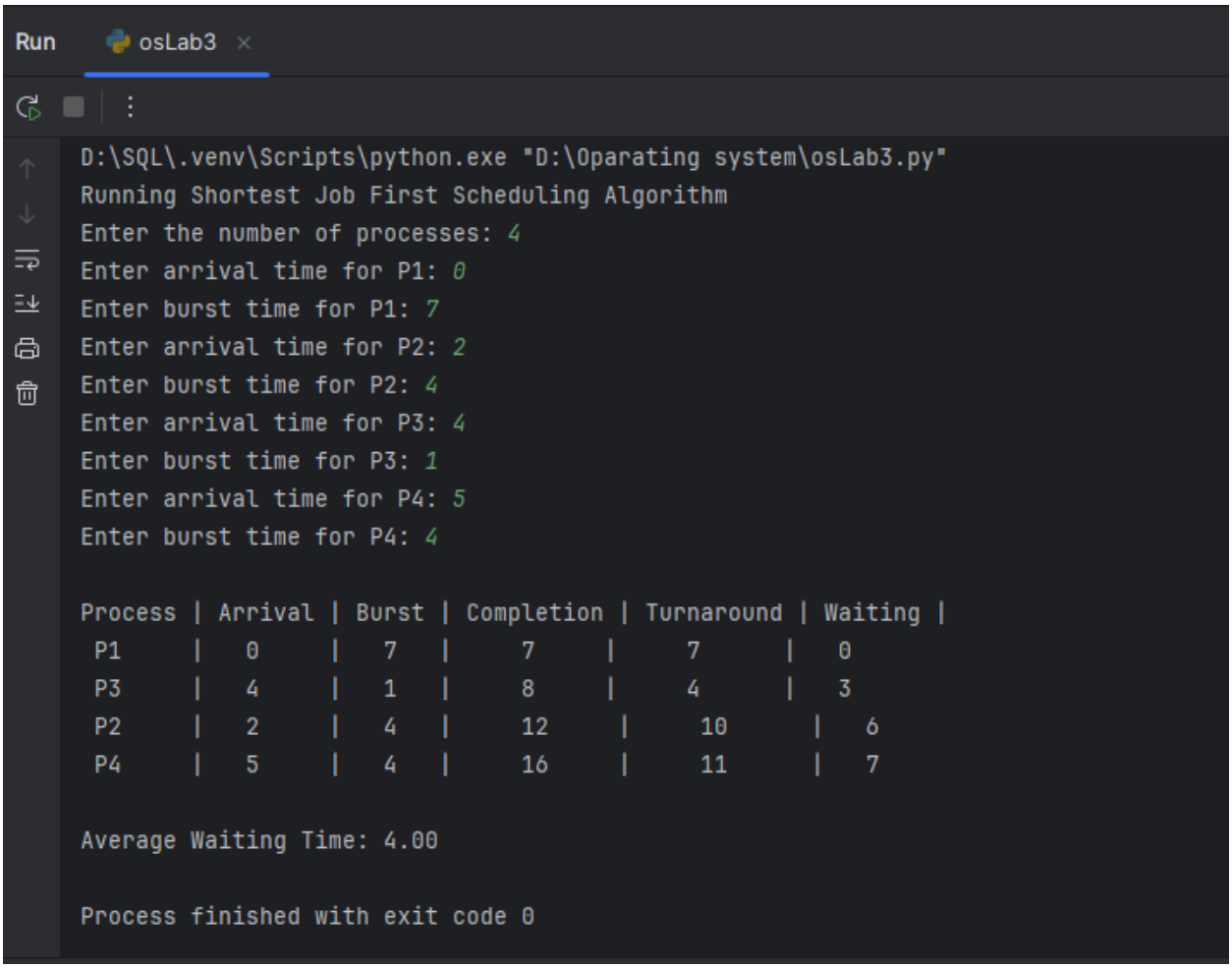
Theory:

Shortest Job First (SJF) is a non-preemptive CPU scheduling algorithm in which the process with the **shortest burst time** is selected for execution next. If two processes have the same burst time, the one that arrived earlier is selected. SJF is optimal in terms of minimizing the **average waiting time** among all scheduling algorithms.

Code:

```
print("Running Shortest Job First Scheduling Algorithm")
n = int(input("Enter the number of processes: "))
d = {}
# Input arrival and burst times for each process
for i in range(n):
    key = "P" + str(i + 1)
    arrival = int(input(f"Enter arrival time for {key}: "))
    burst = int(input(f"Enter burst time for {key}: "))
    d[key] = [arrival, burst]
d = sorted(d.items(), key=lambda item: item[1][0])
# Create a ready queue to schedule by shortest job
time = 0
completed = []
remaining = d.copy()
while remaining:
    # Get processes that have arrived till current time
    available = [proc for proc in remaining if proc[1][0] <= time]
    if available:
        # Select process with shortest burst time
        available.sort(key=lambda x: x[1][1])
        current = available[0]
    else:
        # If no process has arrived yet, jump to the next arrival time
        current = min(remaining, key=lambda x: x[1][0])
        time = current[1][0]
    time += current[1][1]
    completed.append((current[0], current[1][0], current[1][1], time))
    remaining.remove(current)
# Calculate Turnaround Time (TAT) and Waiting Time (WT)
TAT = [comp[3] - comp[1] for comp in completed]
WT = [tat - comp[2] for tat, comp in zip(TAT, completed)]
avg_WT = sum(WT) / n
print("\nProcess | Arrival | Burst | Completion | Turnaround | Waiting |")
for i in range(n):
    p, a, b, c = completed[i]
    print(f" {p}    |   {a}    |  {b}  |    {c}    |    {TAT[i]}     |   {WT[i]}")
print(f"\nAverage Waiting Time: {avg_WT:.2f}")
```

Input and Output:



Explanation:

- **Input Handling:** User provides arrival and burst times for each process.
- **Scheduling Logic:** The algorithm looks for all available processes (those that have arrived) and selects the one with the **shortest burst time**.
- **Completion Time:** Updated dynamically based on the current clock time.
- **Turnaround Time:** Calculated as Completion Time - Arrival Time.
- **Waiting Time:** Calculated as Turnaround Time - Burst Time.
- **Output:** Displays a table with all relevant data, including the average waiting time.

Discussion:

SJF scheduling is efficient in reducing the average waiting time of processes. However:

1. **Starvation:** Long processes may suffer if short processes keep arriving.
2. **Burst Time Estimation:** SJF requires knowledge of burst time in advance, which might not always be practical.
3. **Non-preemptive Nature:** Once a process starts, it runs to completion, which might not be ideal in real-time systems.

Conclusion:

SJF scheduling provides optimal average waiting time and is useful in batch systems where process durations are known. It improves throughput and minimizes overall turnaround time. However, due to its assumptions and starvation risk, it's often improved using **preemptive** variants like **Shortest Remaining Time First (SRTF)**. Despite its limitations, SJF is a key concept in operating system scheduling algorithms.