# UIT2512---Operating Systems Practices Lab

## 1) Implementation of Priority CPU Scheduling Algorithm (Non Preemptive & Preemptive) in Python

**Name:** Vasundhara.B

**Roll no:** 3122 21 5002 119

## AIM:

To implement the Priority CPU Scheduling Algorithm in Python, demonstrating the allocation of CPU time to processes based on their priority in both non-preemptive and preemptive modes.

## Description:

The Priority CPU Scheduling Algorithm assigns CPU time to processes based on their priority values. In the non-preemptive version, once a process starts executing, it will continue until it completes, regardless of whether a higher-priority process arrives later. In contrast, the preemptive version may preempt the currently executing process if a higher-priority process becomes available.

## CODE (non_preemptive):

```
class PrioritySchedulingProcess:
    def __init__(self, process_name, burst_time, waiting_time, turnaround_time,
completion_time, priority):
        self.process_name = process_name
        self.burst_time = burst_time
        self.waiting_time = waiting_time
        self.turnaround_time = turnaround_time
        self.completion_time = completion_time
        self.priority = priority

def main():
    number_of_process = int(input("Enter the total number of Processes: "))

    processes = []
    ASCII_number = 65  # ASCII 'A'

    for i in range(number_of_process):
        process_name = chr(ASCII_number)
        print(f"\nEnter the details of process {process_name}:")
        burst_time = int(input("Enter the burst time: "))
        priority = int(input("Enter the priority: "))
```

```
        process = PrioritySchedulingProcess(process_name, burst_time, 0, 0, 0,
priority)
        processes.append(process)

        ASCII_number += 1

    # Sort processes based on priority (higher priority first)
    processes.sort(key=lambda x: x.priority, reverse=True)

    total_waiting_time = 0
    total_turnaround_time = 0

    processes[0].waiting_time = 0
    processes[0].completion_time = processes[0].burst_time

    for i in range(1, number_of_process):
        processes[i].waiting_time = 0
        for j in range(i):
            processes[i].waiting_time += processes[j].burst_time

        processes[i].completion_time = processes[i - 1].completion_time +
processes[i].waiting_time
        total_waiting_time += processes[i].waiting_time
        total_turnaround_time += processes[i].completion_time

    average_waiting_time = total_waiting_time / number_of_process
    average_turnaround_time = total_turnaround_time / number_of_process
    print("GANTT CHART\n")
    gc="|"
    for process in processes:
        gc+=(process.process_name+("_"*process.completion_time)+"|")
    print(gc)
    print()
    print("\n\nProcess Name\tBurst Time\tCompletion Time\tWaiting Time\tTurnaround
Time")
    print("----------------------------------------------------------------------
--")


    for process in processes:
        process.turnaround_time = process.completion_time
        print(f"{process.process_name}\t\t{process.burst_time}\t\t{process.complet
ion_time}\t\t{process.waiting_time}\t\t{process.turnaround_time}")
        print("----------------------------------------------------------------------
------")

    print(f"\nAverage Waiting Time: {average_waiting_time:.2f}")
    print(f"Average Turnaround Time: {average_turnaround_time:.2f}")

if __name__ == "__main__":
    main()
```

## OUTPUT:

```
Enter the total number of Processes: 3

Enter the details of process A:
Enter the burst time: 5
Enter the priority: 2

Enter the details of process B:
Enter the burst time: 6
Enter the priority: 1

Enter the details of process C:
Enter the burst time: 7
Enter the priority: 3
GANTT CHART

|C_______|A_____________|B_____________________|


Process Name    Burst Time      Completion Time Waiting Time    Turnaround Time
-------------------------------------------------------------------------
C               7               7               0               7
-------------------------------------------------------------------------
A               5               14              7               14
-------------------------------------------------------------------------
B               6               26              12              26
-------------------------------------------------------------------------

Average Waiting Time: 6.33
Average Turnaround Time: 13.33
PS C:\Users\B Vasundhara\Documents\OS>
```

## CODE (preemptive):

```
class Process:
    def __init__(self, no, at, bt, rt, ct, wt, tat, pri, temp):
        self.no = no
        self.at = at
        self.bt = bt
        self.rt = rt
        self.ct = ct
        self.wt = wt
        self.tat = tat
        self.pri = pri
        self.temp = temp


def read(i):
    print("\nProcess No:", i)
    no = i
    at = int(input("Enter Arrival Time: "))
    bt = int(input("Enter Burst Time: "))
```

```
    rt = bt
    pri = int(input("Enter Priority: "))
    temp = pri
    return Process(no, at, bt, rt, 0, 0, 0, pri, temp)


def main():
    print("<--Highest Priority First Scheduling Algorithm (Preemptive)-->\n")
    n = int(input("Enter Number of Processes: "))
    processes = [read(i + 1) for i in range(n)]
    remaining = n
    execution_order = []  # Track the order of execution
    gantt_chart = "|"  # Initialize the Gantt chart

    for i in range(n - 1):
        for j in range(n - i - 1):
            if processes[j].at > processes[j + 1].at:
                processes[j], processes[j + 1] = processes[j + 1], processes[j]

    max_val = processes[0].temp
    max_index = 0
    for j in range(n):
        if processes[j].at <= processes[0].at:
            if processes[j].temp > max_val:
                max_val = processes[j].temp
                max_index = j
    i = max_index
    c = processes[i].ct = processes[i].at + 1
    processes[i].rt -= 1
    execution_order.append(processes[i].no)
    gantt_chart += str(processes[i].no) + "_" * (processes[i].ct - 1) + "|"

    if processes[i].rt == 0:
        processes[i].temp = float("-inf")
        remaining -= 1

    while remaining > 0:
        max_val = processes[0].temp
        max_index = 0
        for j in range(n):
            if processes[j].at <= c:
                if processes[j].temp > max_val:
                    max_val = processes[j].temp
                    max_index = j
        i = max_index
        processes[i].ct = c = c + 1
        processes[i].rt -= 1
        execution_order.append(processes[i].no)
        gantt_chart += str(processes[i].no) + "_"  + "|"

        if processes[i].rt == 0:
```

```
            processes[i].temp = float("-inf")
            remaining -= 1

    print("\nProcessNo\tAT\tBT\tPri\tCT\tTAT\tWT")
    avgtat = 0
    avgwt = 0
    for i in range(n):
        processes[i].tat = processes[i].ct - processes[i].at
        avgtat += processes[i].tat
        processes[i].wt = processes[i].tat - processes[i].bt
        avgwt += processes[i].wt
        print(f"P{processes[i].no}\t\t{processes[i].at}\t{processes[i].bt}\t{proce
sses[i].pri}\t"
              f"{processes[i].ct}\t{processes[i].tat}\t{processes[i].wt}")

    avgtat /= n
    avgwt /= n
    print(f"\nAverage TurnAroundTime={avgtat}\nAverage WaitingTime={avgwt}")

    # Print the order of execution in the Gantt chart format
    print("\nGANTT CHART\n")
    print(gantt_chart)


if __name__ == "__main__":
    main()
```

**OUTPUT:**

```
Enter Number of Processes: 3

Process No: 1
Enter Arrival Time: 1
Enter Burst Time: 5
Enter Priority: 2

Process No: 2
Enter Arrival Time: 3
Enter Burst Time: 6
Enter Priority: 1

Process No: 3
Enter Arrival Time: 2
Enter Burst Time: 7
Enter Priority: 3

ProcessNo       AT      BT      Pri     CT      TAT     WT
P1              1       5       2       13      12      7
P3              2       7       3       9       7       0
P2              3       6       1       19      16      10

Average TurnAroundTime=11.666666666666666
Average WaitingTime=5.666666666666667

GANTT CHART

|1_|3_|3_|3_|3_|3_|3_|3_|1_|1_|1_|1_|2_|2_|2_|2_|2_|2_|
PS C:\Users\B Vasundhara\Documents\OS>
```