Assignment - 2

Aim : CPU scheduling algorithms

Theory :

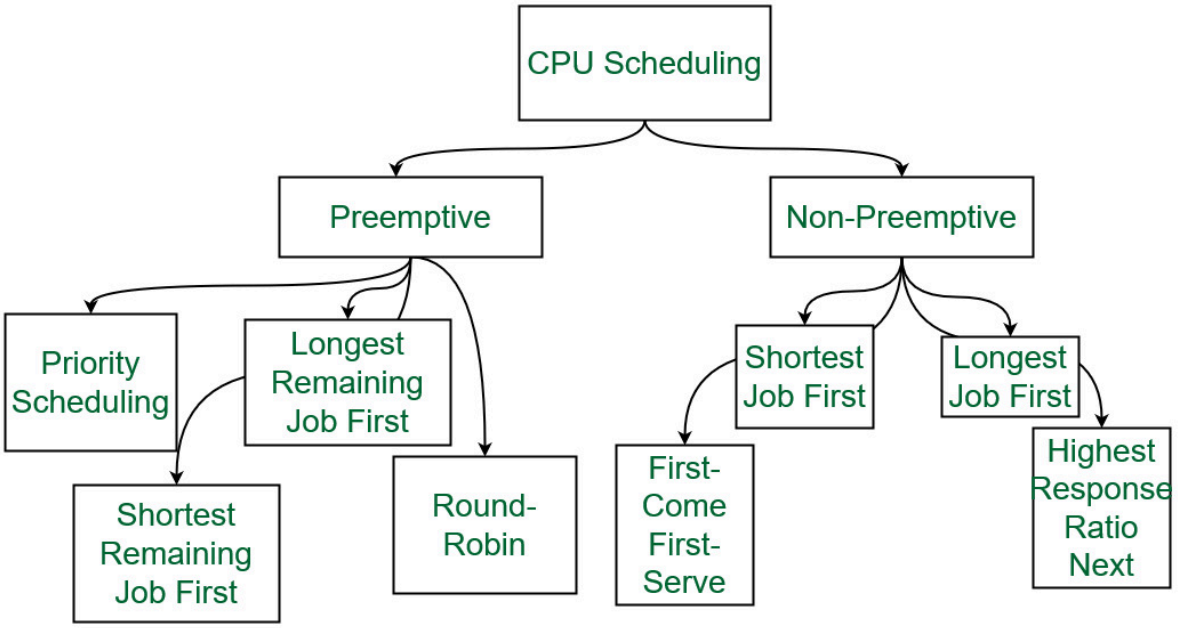
Scheduling of processes/work is done to finish the work on time. **CPU Scheduling** is a process that allows one process to use the CPU while another process is delayed (in standby) due to unavailability of any resources such as I / O etc, thus making full use of the CPU. The purpose of CPU Scheduling is to make the system more efficient, faster, and fairer.

What is a process?

In computing, a process is the instance of a computer program that is being executed by one or many threads. It contains the program code and its activity. Depending on the operating system (OS), a process may be made up of multiple threads of execution that execute instructions concurrently.

What is Process Scheduling?

Process Scheduling is the process of the process manager handling the removal of an active process from the CPU and selecting another process based on a specific strategy.



1. First Come First Serve:

FCFS considered to be the simplest of all operating system scheduling algorithms. First come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first and is implemented by using FIFO queue.

2. Shortest Job First(SJF):

Shortest job first (SJF) is a scheduling process that selects the waiting process with the smallest execution time to execute next. This scheduling method may or may not be preemptive. Significantly reduces the average waiting time for other processes waiting to be executed. The full form of SJF is Shortest Job First.

3. Priority Scheduling:

Preemptive Priority CPU Scheduling Algorithm is a pre-emptive method of CPU scheduling algorithm that works **based on the priority** of a process. In this algorithm, the editor sets the functions to be as important, meaning that the most important process must be done first. In the case of any conflict, that is, where there is more than one process with equal value, then the most important CPU planning algorithm works on the basis of the FCFS (First Come First Serve) algorithm.

4. Round robin:

Round Robin is a CPU scheduling algorithm where each process is cyclically assigned a fixed time slot. It is the preemptive version of First come First Serve CPU Scheduling algorithm. Round Robin CPU Algorithm generally focuses on Time Sharing technique.

Program & Output

```
import matplotlib.pyplot as plt
import numpy as np

arrival_time = []
burst_time = []



def first_come_first_serve():
   print("First serve algorithm")
   completion_time = [0]*len(arrival_time)
   waiting_time = [0]*len(arrival_time)
   turnaround_time = [0]*len(arrival_time)

   completion_time[0] = burst_time[0];
   for i in range(1, len(arrival_time)):
       completion_time[i] = max(completion_time[i-1] + burst_time[i],
0)

   total_turnaround_time = 0
   total_waiting_time = 0

   for i in range(len(arrival_time)):
       turnaround_time[i] = max(completion_time[i] - arrival_time[i],
0)  #completion_time[i] - arrival_time[i] is turnaround time
       waiting_time[i] = max(turnaround_time[i] - burst_time[i],0)
#turnaround time - burst_time[i] is waiting time
       total_turnaround_time += turnaround_time[i]
       total_waiting_time += waiting_time[i]

   average_waiting_time = total_waiting_time / len(arrival_time)
   average_turnaround_time = total_turnaround_time / len(arrival_time)



   print("Process \t Arrival time \t Burst time \t Completion time \t
Turnaround time \t waiting time")
   for i in range(len(arrival_time)):
       print(f"{i+1}\t\t {arrival_time[i]}\t\t {burst_time[i]}\t\t
{completion_time[i]}\t\t\t {turnaround_time[i]}\t\t\t
{waiting_time[i]}")

   print(f"\nAverage Waiting Time: {average_waiting_time}")
```

```
    print(f"\nAverage Turn Around Time: {average_turnaround_time}")

    plt.figure(figsize=(10, 2))
    colors = np.random.rand(len(arrival_time), 3)
    for i in range(len(arrival_time)):
        plt.barh(y=0, width=burst_time[i], left=max(completion_time[i] -
burst_time[i], arrival_time[i]), color=colors[i], edgecolor='black')
        plt.text(max(completion_time[i] - burst_time[i],
arrival_time[i]) + burst_time[i] / 2, 0.5, f"P{i+1}", ha='center',
va='center', color='white')

    plt.title('Gantt Chart - FCFS')
    plt.xlabel('Time')
    plt.yticks([])
    plt.show()




def shortest_job_first():
    print("Shortest job first")

    while True:
        print("\nChoose one algorithm : ")
        print("1. Preemptive ")
        print("2. Non Preemptive")
        print("3. Exit")

        ch = int(input("Enter algorithm No : "))

        match ch:
            case 1 :
                sjf_preemptive()
            case 2 :
                sjf_non_preemptive()
            case 3 :
                break
            case _:
                print("Algorithm not found")
                break
```

```
def sjf_preemptive():
   print("Preemptive")

   switch_time = int(input("Enter switch time : "))

   n = len(arrival_time)
   remaining_time = burst_time.copy()
   completion_time = [0] * n
   waiting_time = [0]* n
   turnaround_time = [0]* n
   total_waiting_time = 0
   total_turnaround_time = 0
   current_time = 0
   process_sequence = []

   while True:
       shortest_job = None
       shortest_time = float('inf')

       for i in range(n):
           if(arrival_time[i] <= current_time and remaining_time[i] <
shortest_time and remaining_time[i] > 0):
               shortest_job = i
               shortest_time = remaining_time[i]

       if shortest_job is None:
           break

       process_sequence.append(shortest_job + 1)
       remaining_time[shortest_job] -= switch_time

       if remaining_time[shortest_job] == 0 :
           completion_time[shortest_job] = current_time
           turnaround_time[shortest_job] =
completion_time[shortest_job] - arrival_time[shortest_job]
           waiting_time[shortest_job] = turnaround_time[shortest_job] -
burst_time[shortest_job]
           total_waiting_time += waiting_time[shortest_job]
           total_turnaround_time += turnaround_time[shortest_job]

       current_time += switch_time
```

```
    average_turnaround_time = total_turnaround_time/n
    average_waiting_time = total_waiting_time/n

    print("Process \t Arrival time \t Burst time \t Completion time \t
Turnaround time \t waiting time")
    for i in range(len(arrival_time)):
        print(f"{i+1}\t\t {arrival_time[i]}\t\t {burst_time[i]}\t\t
{completion_time[i]}\t\t\t {turnaround_time[i]}\t\t\t
{waiting_time[i]}")

    print(f"\nAverage Waiting Time: {average_waiting_time}")
    print(f"\nAverage Turn Around Time: {average_turnaround_time}")

    colors = plt.cm.viridis(np.linspace(0, 1, n))

    plt.figure(figsize=(10, 1.5))
    for i,process in enumerate(process_sequence):
        plt.barh(0, switch_time, left=i * switch_time,
color=colors[process-1], align='center', edgecolor='black')
        plt.text(i * switch_time + switch_time / 2, 0,
f'P{process_sequence[i]}', ha='center', va='center', color='white')
    plt.xlim(0, len(process_sequence) * switch_time)
    plt.ylim(-0.5, 0.5)
    plt.yticks([])
    plt.title('Gantt Chart - Round Robin Scheduling')
    plt.xlabel('Time')
    plt.show()




def sjf_non_preemptive():
    print("Non Preemptive")
    n = len(arrival_time)
    remaining_time = burst_time.copy()
    completion_time = [0] * n
    waiting_time = [0]* n
    turnaround_time = [0]* n
    total_waiting_time = 0
    total_turnaround_time = 0
    current_time = 0
```

```
    while True:
        shortest_job = None
        shortest_time = float('inf')

        for i in range(n):
            if(arrival_time[i] <= current_time and remaining_time[i] <
shortest_time and remaining_time[i] > 0):
                shortest_job = i
                shortest_time = remaining_time[i]

        if shortest_job is None:
            break

        current_time += remaining_time[shortest_job]
        completion_time[shortest_job] = current_time
        remaining_time[shortest_job] = 0
        turnaround_time[shortest_job] = completion_time[shortest_job] -
arrival_time[shortest_job]
        waiting_time[shortest_job] = turnaround_time[shortest_job] -
burst_time[shortest_job]
        total_waiting_time += waiting_time[shortest_job]
        total_turnaround_time += turnaround_time[shortest_job]

    average_turnaround_time = total_turnaround_time/n
    average_waiting_time = total_waiting_time/n

    print("Process \t Arrival time \t Burst time \t Completion time \t
Turnaround time \t waiting time")
    for i in range(len(arrival_time)):
        print(f"{i+1}\t\t {arrival_time[i]}\t\t {burst_time[i]}\t\t
{completion_time[i]}\t\t\t {turnaround_time[i]}\t\t\t
{waiting_time[i]}")

    print(f"\nAverage Waiting Time: {average_waiting_time}")
    print(f"\nAverage Turn Around Time: {average_turnaround_time}")

    plt.figure(figsize=(10, 2))
    colors = np.random.rand(len(arrival_time), 3)
    for i in range(len(arrival_time)):
        plt.barh(y=0, width=burst_time[i], left=max(completion_time[i] -
burst_time[i], arrival_time[i]), color=colors[i], edgecolor='black')
```

```
        plt.text(max(completion_time[i] - burst_time[i],
arrival_time[i]) + burst_time[i] / 2, 0.5, f"P{i+1}", ha='center',
va='center', color='white')

    plt.title('Gantt Chart - Priority')
    plt.xlabel('Time')
    plt.yticks([])
    plt.show()




def round_robin():
    print("Round Robin Algorithm")
    n = len(arrival_time)
    remaining_time = burst_time.copy()
    completion_time = [0] * n
    turnaround_time = [0] * n
    waiting_time = [0] * n
    total_waiting_time = 0
    total_turnaround_time = 0
    current_time = 0
    process_sequence = []  # to store the sequence of processes executed

    time_slice = int(input("Enter time slice: "))

    while True:
        all_processes_completed = True
        for i in range(n):
            if remaining_time[i] > 0:
                all_processes_completed = False
                process_sequence.append(i + 1)  # Append process number
to the sequence
                if remaining_time[i] > time_slice:
                    current_time += time_slice
                    remaining_time[i] -= time_slice
                else:
                    current_time += remaining_time[i]
                    remaining_time[i] = 0
                    completion_time[i] = current_time
                    turnaround_time[i] = completion_time[i] -
arrival_time[i]
                    waiting_time[i] = turnaround_time[i] - burst_time[i]
```

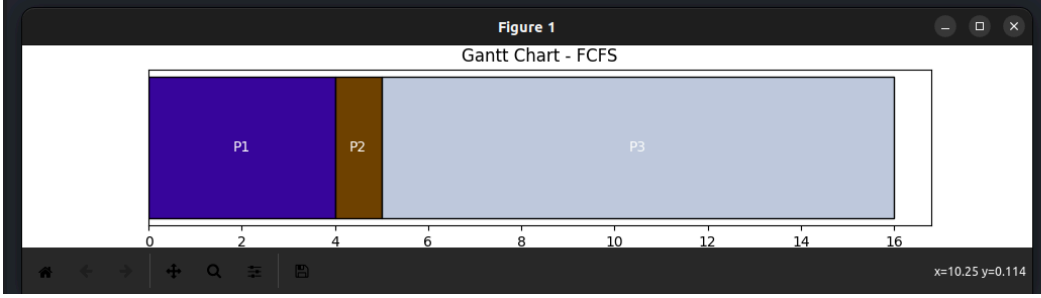
```
                    total_waiting_time += waiting_time[i]
                    total_turnaround_time += turnaround_time[i]

        if all_processes_completed:
            break

    average_waiting_time = total_waiting_time / n
    average_turnaround_time = total_turnaround_time / n

    print("Process \t Arrival time \t Burst time \t Completion time \t
Turnaround time \t Waiting time")
    for i in range(len(arrival_time)):
        print(f"{i+1}\t\t {arrival_time[i]}\t\t {burst_time[i]}\t\t
{completion_time[i]}\t\t\t {turnaround_time[i]}\t\t\t
{waiting_time[i]}")

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

    colors = plt.cm.viridis(np.linspace(0, 1, n))

    # Plotting Gantt Chart
    plt.figure(figsize=(10, 1.5))
    for i, process in enumerate(process_sequence):
        plt.barh(0, time_slice, left=i * time_slice,
color=colors[process-1], align='center', edgecolor='black')
        plt.text(i * time_slice + time_slice / 2, 0, f'P{process}',
ha='center', va='center', color='white')
    plt.xlim(0, len(process_sequence) * time_slice)
    plt.ylim(-0.5, 0.5)
    plt.yticks([])
    plt.title('Gantt Chart - Round Robin Scheduling')
    plt.xlabel('Time')
    plt.show()


def priority_algo():
    print("Priority Based Algorithm")
    n = len(arrival_time)
    completion_time = [0] * n
    waiting_time = [0] * n
    turnaround_time = [0] * n
    total_waiting_time = 0
```

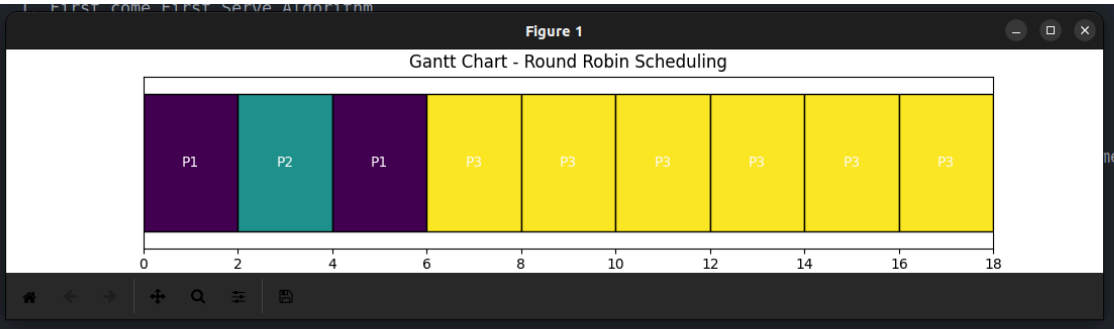
```
    total_turnaround_time = 0
    current_time = 0
    priority = []

    for i in range(n):
        priority.append(int(input(f"Enter Priority of P{i+1} : ")))

    processes = list(range(n))
    processes.sort(key=lambda x: priority[x], reverse=True)

    for i in processes:
        current_time = max(current_time, arrival_time[i])
        completion_time[i] = current_time + burst_time[i]
        turnaround_time[i] = completion_time[i] - arrival_time[i]
        waiting_time[i] = turnaround_time[i] - burst_time[i]
        total_waiting_time += waiting_time[i]
        total_turnaround_time += turnaround_time[i]
        current_time = completion_time[i]

    average_turnaround_time = total_turnaround_time / n
    average_waiting_time = total_waiting_time / n

    print("Process \t Arrival time \t Burst time \t Priority \t
Completion time \t Turnaround time \t Waiting time")
    for i in range(n):
        print(f"{i+1}\t\t {arrival_time[i]}\t\t {burst_time[i]}\t\t
{priority[i]}\t\t {completion_time[i]}\t\t\t {turnaround_time[i]}\t\t\t
{waiting_time[i]}")

    print(f"\nAverage Waiting Time: {average_waiting_time}")
    print(f"Average Turn Around Time: {average_turnaround_time}")

    plt.figure(figsize=(10, 2))
    colors = np.random.rand(len(arrival_time), 3)
    for i in range(len(arrival_time)):
        plt.barh(y=0, width=burst_time[i], left=max(completion_time[i] -
burst_time[i], arrival_time[i]), color=colors[i], edgecolor='black')
        plt.text(max(completion_time[i] - burst_time[i],
arrival_time[i]) + burst_time[i] / 2, 0.5, f"P{i+1}", ha='center',
va='center', color='white')

    plt.title('Gantt Chart - Priority')
    plt.xlabel('Time')
```

```
    plt.yticks([])
    plt.show()




if __name__ == "__main__":

    noOfProcess = int(input("Enter number of processes : "))

    for index in range(noOfProcess):
        print("\nProcess no ",index+1)
        arrival_time.append(int(input("Enter Arrival time : ")))
        burst_time.append(int(input("Enter Burst time : ")))

    while True:
        print("\nChoose one algorithm : ")
        print("1. First come First Serve Algorithm")
        print("2. Short job first Algorithm")
        print("3. Round Robin Algorithm")
        print("4. Priority based Algorithm")
        print("5. Exit")

        option = int(input("Enter algorithm No : "))

        match option:
            case 1 :
                first_come_first_serve()
            case 2 :
                shortest_job_first()
            case 3 :
                round_robin()
            case 4 :
                priority_algo()
            case 5 :
                print("Exiting")
                break
            case _:
                print("Algorithm not found")
                break
```

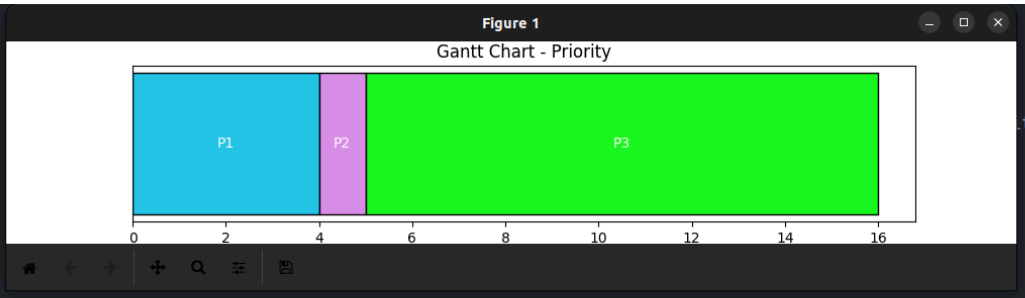
```
➜  OS python3 scheduling_algorithm.py
Enter number of processes : 3

Process no  1
Enter Arrival time : 0
Enter Burst time : 4

Process no  2
Enter Arrival time : 2
Enter Burst time : 1

Process no  3
Enter Arrival time : 4
Enter Burst time : 10

Choose one algorithm :
1. First come First Serve Algorithm
2. Short job first Algorithm
3. Round Robin Algorithm
4. Priority based Algorithm
5. Exit
Enter algorithm No :
```

```
Choose one algorithm :
1. First come First Serve Algorithm
2. Short job first Algorithm
3. Round Robin Algorithm
4. Priority based Algorithm
5. Exit
Enter algorithm No : 1
First serve algorithm
Process         Arrival time    Burst time      Completion time         Turnaround time         waiting time
1               0               4               4                       4                       0
2               2               1               5                       3                       2
3               4               11              16                      12                      1

Average Waiting Time: 1.0

Average Turn Around Time: 6.333333333333333
```

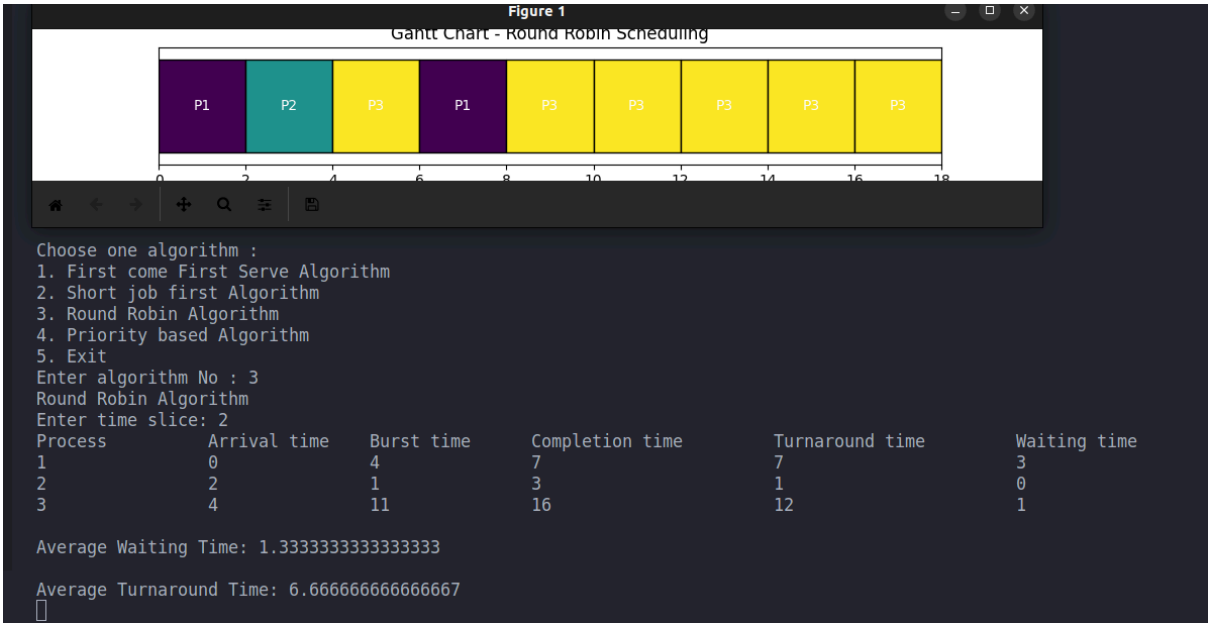





```
Choose one algorithm :
1. First come First Serve Algorithm
2. Short job first Algorithm
3. Round Robin Algorithm
4. Priority based Algorithm
5. Exit
Enter algorithm No : 2
Shortest job first

Choose one algorithm :
1. Preemptive
2. Non Preemptive
3. Exit
Enter algorithm No : 1
Preemptive
Enter switch time : 2
Process         Arrival time    Burst time      Completion time         Turnaround time         waiting time
1               0               4               4                       4                       0
2               2               1               0                       0                       0
3               4               11              0                       0                       0

Average Waiting Time: 0.0

Average Turn Around Time: 1.3333333333333333
```




```
Choose one algorithm :
1. Preemptive
2. Non Preemptive
3. Exit
Enter algorithm No : 2
Non Preemptive
Process         Arrival time    Burst time      Completion time         Turnaround time         waiting time
1               0               4               4                       4                       0
2               2               1               5                       3                       2
3               4               11              16                      12                      1

Average Waiting Time: 1.0

Average Turn Around Time: 6.333333333333333
```




```
Choose one algorithm :
1. First come First Serve Algorithm
2. Short job first Algorithm
3. Round Robin Algorithm
4. Priority based Algorithm
5. Exit
Enter algorithm No : 3
Round Robin Algorithm
Enter time slice: 2
Process          Arrival time    Burst time      Completion time         Turnaround time         Waiting time
1                0               4               7                       7                       3
2                2               1               3                       1                       0
3                4               11              16                      12                      1

Average Waiting Time: 1.3333333333333333

Average Turnaround Time: 6.666666666666667
```


```
ting time
Choose one algorithm :
1. First come First Serve Algorithm
2. Short job first Algorithm
3. Round Robin Algorithm
4. Priority based Algorithm
5. Exit
Enter algorithm No : 4
Priority Based Algorithm
Enter Priority of P1 : 2
Enter Priority of P2 : 1
Enter Priority of P3 : 3
Process          Arrival time    Burst time      Priority        Completion time         Turnaround time         Wai
1                0               4               2               19                      19                      15
2                2               1               1               20                      18                      17
3                4               11              3               15                      11                      0

Average Waiting Time: 10.666666666666666
Average Turn Around Time: 16.0
```

Conclusion :

Hence, here we learnt about what is process, what is CPU scheduling, what are the types of CPU scheduling algorithms and implemented those in python and plotted gantt charts using matplotlib library.