SCHEDULING ALGORITHMS (FIFO, HRRN, SJF, SRTF)

CODE:-

```
def FIFO():
    # Input section
    num_process = int(input("Enter the number of processes: "))  # Get the number of processes
    Arrival_Time = []
    Execution_Time = []

    # Collecting arrival time and execution time for each process
    for i in range(num_process):
        print(f"Enter the details of P{i}:")
        Arrival = int(input("Enter Arrival Time: "))
        Execution = int(input("Enter Execution Time: "))
        Arrival_Time.append(Arrival)
        Execution_Time.append(Execution)

    # Initialize lists for start time, completion time, turnaround time, and waiting time
    Start_Time = [0] * num_process
    Finish_Time = [0] * num_process
    Turnaround_Time = [0] * num_process
    Waiting_Time = [0] * num_process
    Utilization = [0]*num_process
    # Sort processes by Arrival Time (if not already sorted)
    process_order = sorted(range(num_process), key=lambda i: Arrival_Time[i])
    # Calculate start time, completion time, etc.
    current_time = 0
    for i in process_order:
        # Start time is either the current time or the arrival time, whichever is greater
        Start_Time[i] = max(current_time, Arrival_Time[i])
        # Completion time is the start time + execution time
        Finish_Time[i] = Start_Time[i] + Execution_Time[i]
        # Turnaround time is completion time - arrival time
        Turnaround_Time[i] = Finish_Time[i] - Arrival_Time[i]
        # Waiting time is turnaround time - execution time
        Waiting_Time[i] = Turnaround_Time[i] - Execution_Time[i]
        Utilization[i]=Execution_Time[i]/Turnaround_Time[i]
        # Update current time to the completion time of the current process
        current_time = Finish_Time[i]
    # Print the results
    print("\nP\tAT\tET\tST\tFT\tTAT\tWT\tUti")
    for i in range(num_process):
        print(f"P{i}\t{Arrival_Time[i]}\t{Execution_Time[i]}\t{Start_Time[i]}\t{Finish_Time[i]}\t{Turnaroun
d_Time[i]}\t{Waiting_Time[i]}\t{Utilization[i]}")
    print("\nGantt Chart:")
    current_time = 0
    print("|", end="")
    for i in process_order:
        start = max(current_time, Arrival_Time[i])
        # Print idle time (if there's any gap between current_time and process start time)
        if start > current_time:
            idle_time = start - current_time
            for _ in range(idle_time):
                print(" 0 |", end="")
        # Print process execution
        for _ in range(Execution_Time[i]):
            print(f" P{i} |", end="")
        # Update current time to the finish time of the process
        current_time = start + Execution_Time[i]
    print("\n")
def SRTF():
    # Input section
    num_process = int(input("Enter the number of processes: "))  # Get the number of processes
    Arrival_Time = []
    Execution_Time = []

    # Collecting arrival time and execution time for each process
    for i in range(num_process):
        print(f"Enter the details of P{i}:")
        Arrival = int(input("Enter Arrival Time: "))
        Execution = int(input("Enter Execution Time: "))
        Arrival_Time.append(Arrival)
        Execution_Time.append(Execution)
    # Remaining execution times (initially the same as execution times)
    Remaining_Time = Execution_Time[:]
```

```
    # Initialize variables
    current_time = 0
    completed = 0
    min_remaining_time = float('inf')
    shortest = 0
    finish_time = 0
    is_completed = [False] * num_process
    is_started = [False] * num_process  # To track if a process has started
    Start_Time = [-1] * num_process  # Initialize to -1, meaning not yet started
    Waiting_Time = [0] * num_process
    Turnaround_Time = [0] * num_process
    Finish_Time = [0] * num_process
    Utilization = [0] * num_process
    gantt_chart = []
    # Loop until all processes are completed
    while completed != num_process:
        # Find the process with the shortest remaining time at the current time
        for i in range(num_process):
            if Arrival_Time[i] <= current_time and not is_completed[i] and Remaining_Time[i] <
min_remaining_time and Remaining_Time[i] > 0:
                min_remaining_time = Remaining_Time[i]
                shortest = i
        if min_remaining_time == float('inf'):  # No process is ready to execute, so CPU is idle
            gantt_chart.append("0")  # Represent idle time with "0"
            current_time += 1
            continue
        # If the process is starting for the first time, record its start time
        if not is_started[shortest]:
            Start_Time[shortest] = current_time
            is_started[shortest] = True
        # Decrement the remaining time of the current shortest process
        Remaining_Time[shortest] -= 1
        gantt_chart.append(f"P{shortest}")  # Add process execution to Gantt chart
        if Remaining_Time[shortest] == 0:
            # If the process is completed, record the finish time
            completed += 1
            is_completed[shortest] = True
            finish_time = current_time + 1
            Finish_Time[shortest] = finish_time
            # Calculate turnaround and waiting time for the completed process
            Turnaround_Time[shortest] = Finish_Time[shortest] - Arrival_Time[shortest]
            Waiting_Time[shortest] = Turnaround_Time[shortest] - Execution_Time[shortest]
            Utilization[shortest] = Execution_Time[shortest] / Turnaround_Time[shortest]
            if Waiting_Time[shortest] < 0:
                Waiting_Time[shortest] = 0
        current_time += 1
        min_remaining_time = float('inf')
    # Print the results
    print("\nP\tAT\tET\tST\tFT\tTAT\tWT\tUti")
    for i in range(num_process):
        print(f"P{i}\t{Arrival_Time[i]}\t{Execution_Time[i]}\t{Start_Time[i]}\t{Finish_Time[i]}\t{Turnaroun
d_Time[i]}\t{Waiting_Time[i]}\t{Utilization[i]:.2f}")

    # Gantt Chart display
    print("\nGantt Chart:")
    print("|", end="")
    for i in gantt_chart:
        print(f" {i} |", end="")
    print("\n")
def SJF():
    # Input section
    num_process = int(input("Enter the number of processes: "))  # Get the number of processes
    Arrival_Time = []
    Execution_Time = []

    # Collecting arrival time and execution time for each process
    for i in range(num_process):
        print(f"Enter the details of P{i}:")
        Arrival = int(input("Enter Arrival Time: "))
        Execution = int(input("Enter Execution Time: "))
        Arrival_Time.append(Arrival)
        Execution_Time.append(Execution)
```

```
    # Initialize lists for start time, completion time, turnaround time, and waiting time
    Start_Time = [0] * num_process
    Finish_Time = [0] * num_process
    Turnaround_Time = [0] * num_process
    Waiting_Time = [0] * num_process
    Utilization = [0]*num_process
    # Sort processes by execution time (if multiple processes arrive simultaneously, shortest job runs
first)
    process_order = sorted(range(num_process), key=lambda i: (Arrival_Time[i], Execution_Time[i]))
    # Calculate start time, completion time, etc.
    current_time = 0
    for i in process_order:
        # Start time is either the current time or the arrival time, whichever is greater
        Start_Time[i] = max(current_time, Arrival_Time[i])
        # Completion time is the start time + execution time
        Finish_Time[i] = Start_Time[i] + Execution_Time[i]
        # Turnaround time is completion time - arrival time
        Turnaround_Time[i] = Finish_Time[i] - Arrival_Time[i]
        # Waiting time is turnaround time - execution time
        Waiting_Time[i] = Turnaround_Time[i] - Execution_Time[i]
        Utilization[i]=Execution_Time[i]/Turnaround_Time[i]
        # Update current time to the completion time of the current process
        current_time = Finish_Time[i]
    # Print the results
    print("\nP\tAT\tET\tST\tFT\tTAT\tWT\tUti")
    for i in range(num_process):
        print(f"P{i}\t{Arrival_Time[i]}\t{Execution_Time[i]}\t{Start_Time[i]}\t{Finish_Time[i]}\t{Turnaroun
d_Time[i]}\t{Waiting_Time[i]}\t{Utilization[i]}")

    print("\nGantt Chart:")
    current_time = 0
    print("|", end="")
    for i in process_order:
        start = max(current_time, Arrival_Time[i])
        # Print idle time (if there's any gap between current_time and process start time)
        if start > current_time:
            idle_time = start - current_time
            for _ in range(idle_time):
                print(" 0 |", end="")
        # Print process execution
        for _ in range(Execution_Time[i]):
            print(f" P{i} |", end="")
        # Update current time to the finish time of the process
        current_time = start + Execution_Time[i]
    print("\n")
def HRRN():
    num_process = int(input("Enter number of processes: "))
    Arrival_Time = []
    Execution_Time = []
    for i in range(num_process):
        print(f"Enter the details of p{i}")
        Arrival = int(input("Enter Arrival Time: "))
        Execution = int(input("Enter Execution Time: "))
        Arrival_Time.append(Arrival)
        Execution_Time.append(Execution)
    # Step 3: Keep track of when each friend starts, finishes, and waits
    Start_Time = [0] * num_process
    Finish_Time = [0] * num_process
    Wait_Time = [0] * num_process
    TurnAround_Time = [0] * num_process
    Utilization = [0] * num_process
    Wait_Time = [0] * num_process
    # Step 4: A list to keep track of which friends have Executioned
    Finished = [False] * num_process
    current_time = 0
    finished_process = 0
    gantt_chart=[]
    # Step 5: Let friends Execution until everyone has Executioned
    while finished_process < num_process:
        max_response_ratio = -1
        next_process = -1
```

```
        # Step 6: Find the friend with the highest response ratio
        for i in range(num_process):
            if Arrival_Time[i] <= current_time and not Finished[i]:
                wait_time = current_time - Arrival_Time[i]
                response_ratio = (wait_time + Execution_Time[i]) / Execution_Time[i]
                if response_ratio > max_response_ratio:
                    max_response_ratio = response_ratio
                    next_process = i
         # Step 7: If no process is ready, move the time forward (CPU is idle)
        if next_process == -1:
            gantt_chart.append("0")  # Idle time represented by "0"
            current_time += 1
            continue
        # Step 8: Let the selected friend Execution
        Start_Time[next_process] = current_time
        Finish_Time[next_process] = current_time + Execution_Time[next_process]
        Wait_Time[next_process] = Start_Time[next_process] - Arrival_Time[next_process]
        TurnAround_Time[next_process]=Finish_Time[next_process]-Arrival_Time[next_process]
        Utilization[next_process]=TurnAround_Time[next_process]/Execution_Time[next_process]
        # Add process execution to the Gantt chart
        gantt_chart.extend([f"P{next_process}"] * Execution_Time[next_process])
        current_time = Finish_Time[next_process]
        # Mark the friend as finished
        Finished[next_process] = True
        finished_process += 1
    # Step 9: Show when each friend Executioned and how long they waited
    print("\nP\tAT\tET\tST\tFT\tTAT\tWT\tUti")
    for i in range(num_process):
        print(f"P{i}\t{Arrival_Time[i]}\t{Execution_Time[i]}\t{Start_Time[i]}\t{Finish_Time[i]}\t{TurnAroun
d_Time[i]}\t{Wait_Time[i]}\t{Utilization[i]:.2f}")

    # Gantt Chart display
    print("\nGantt Chart:")
    print("|", end="")
    for i in gantt_chart:
        print(f" {i} |", end="")
    print("\n")
option = int (input ("Enter 1 for FIFO, 2 for SRTF, 3 for SJF, 4 for HRRN: "))
if(option == 1):
    FIFO()
elif(option == 2):
    SRTF()
elif(option == 3):
    SJF()
elif(option == 4):
    HRRN()
```

OUTPUT:-

FIFO:

```
Enter 1 for FIFO, 2 for SRTF, 3 for SJF, 4 for HRRN: 1
Enter the number of processes: 4
Enter the details of P0:
Enter Arrival Time: 0
Enter Execution Time: 2
Enter the details of P1:
Enter Arrival Time: 1
Enter Execution Time: 2
Enter the details of P2:
Enter Arrival Time: 5
Enter Execution Time: 3
Enter the details of P3:
Enter Arrival Time: 6
Enter Execution Time: 4

P       AT      ET      ST      FT      TAT     WT      Uti
P0      0       2       0       2       2       0       1.0
P1      1       2       2       4       3       1       0.6666666666666666
P2      5       3       5       8       3       0       1.0
P3      6       4       8       12      6       2       0.6666666666666666

Gantt Chart:
| P0 | P0 | P1 | P1 | 0 | P2 | P2 | P2 | P3 | P3 | P3 | P3 |
```

SRTF:

```
Enter 1 for FIFO, 2 for SRTF, 3 for SJF, 4 for HRRN: 2
Enter the number of processes: 5
Enter the details of P0:
Enter Arrival Time: 0
Enter Execution Time: 10
Enter the details of P1:
Enter Arrival Time: 1
Enter Execution Time: 1
Enter the details of P2:
Enter Arrival Time: 2
Enter Execution Time: 2
Enter the details of P3:
Enter Arrival Time: 3
Enter Execution Time: 1
Enter the details of P4:
Enter Arrival Time: 4
Enter Execution Time: 5

P       AT      ET      ST      FT      TAT     WT      Uti
P0      0       10      0       19      19      9       0.53
P1      1       1       1       2       1       0       1.00
P2      2       2       2       4       2       0       1.00
P3      3       1       4       5       2       1       0.50
P4      4       5       5       10      6       1       0.83

Gantt Chart:
| P0 | P1 | P2 | P2 | P3 | P4 | P4 | P4 | P4 | P4 | P0 | P0 | P0 | P0 | P0 | P0 | P0 | P0 | P0 |
```

SJF:

```
Enter 1 for FIFO, 2 for SRTF, 3 for SJF, 4 for HRRN: 3
Enter the number of processes: 5
Enter the details of P0:
Enter Arrival Time: 0
Enter Execution Time: 10
Enter the details of P1:
Enter Arrival Time: 1
Enter Execution Time: 1
Enter the details of P2:
Enter Arrival Time: 2
Enter Execution Time: 2
Enter the details of P3:
Enter Arrival Time: 3
Enter Execution Time: 1
Enter the details of P4:
Enter Arrival Time: 4
Enter Execution Time: 5

P       AT      ET      ST      FT      TAT     WT      Uti
P0      0       10      0       10      10      0       1.0
P1      1       1       10      11      10      9       0.1
P2      2       2       11      13      11      9       0.18181818181818182
P3      3       1       13      14      11      10      0.09090909090909091
P4      4       5       14      19      15      10      0.3333333333333333

Gantt Chart:
| P0 | P0 | P0 | P0 | P0 | P0 | P0 | P0 | P0 | P0 | P1 | P2 | P2 | P3 | P4 | P4 | P4 | P4 | P4 |
```

HRRN:

```
Enter 1 for FIFO, 2 for SRTF, 3 for SJF, 4 for HRRN: 4
Enter number of processes: 5
Enter the details of p0
Enter Arrival Time: 0
Enter Execution Time: 3
Enter the details of p1
Enter Arrival Time: 2
Enter Execution Time: 6
Enter the details of p2
Enter Arrival Time: 4
Enter Execution Time: 4
Enter the details of p3
Enter Arrival Time: 6
Enter Execution Time: 5
Enter the details of p4
Enter Arrival Time: 8
Enter Execution Time: 2

P       AT      ET      ST      FT      TAT     WT      Uti
P0      0       3       0       3       3       0       1.00
P1      2       6       3       9       7       1       1.17
P2      4       4       9       13      9       5       2.25
P3      6       5       15      20      14      9       2.80
P4      8       2       13      15      7       5       3.50
```

```
Gantt Chart:
| P0 | P0 | P0 | P1 | P1 | P1 | P1 | P1 | P1 | P2 | P2 | P2 | P2 | P4 | P4 | P3 | P3 | P3 | P3 | P3 |
```