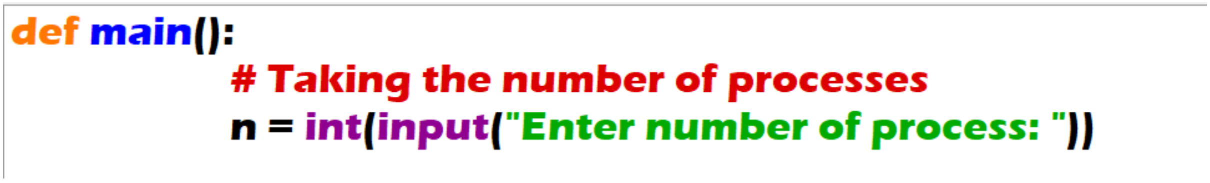
D. Longest Job First

3. The process can be classified into many groups in
   A. shortest job scheduling algorithm
   B. multilevel queue scheduling algorithm
   C. round-robin scheduling algorithm
   D. priority scheduling algorithm
4. The turnaround time for short jobs during multiprogramming is usually Shortened and that for long jobs is slightly ___________
   A. Shortened
   B. Unchanged
   C. Lengthened
   D. Shortened
5. Time quantum can be said
   A. multilevel queue scheduling algorithm
   B. round-robin scheduling algorithm
   C. shortest job scheduling algorithm
   D. priority scheduling algorithm

Student Work Area

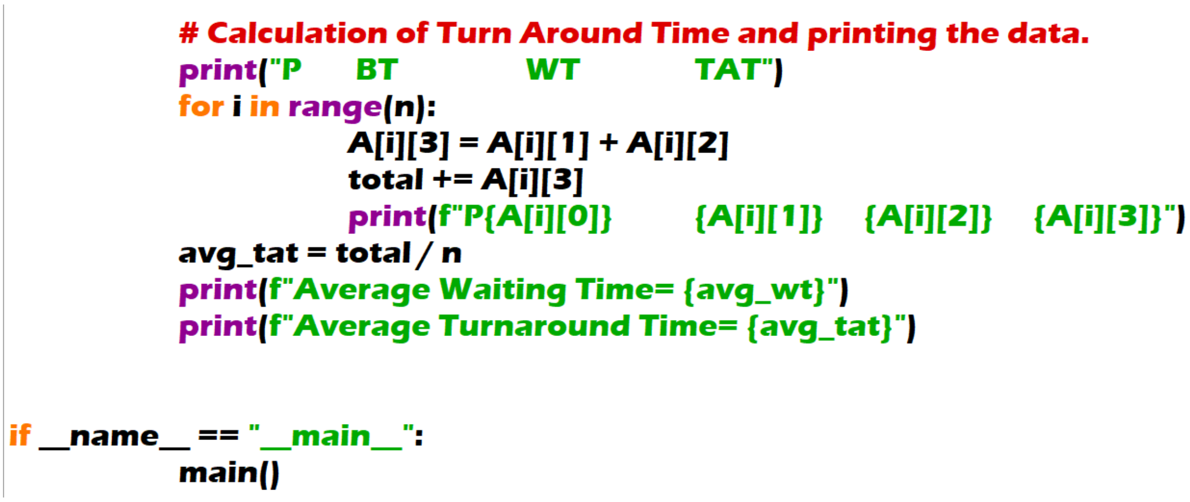
Algorithm/Flowchart/Code/Sample Outputs

<u>SHORTEST-JOB FIRST :-</u>

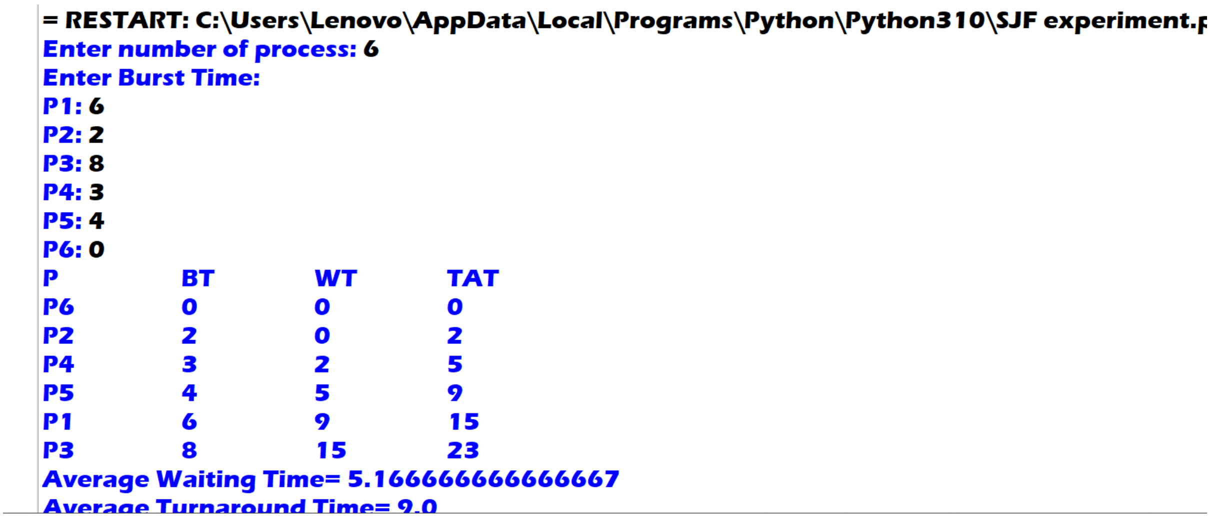


```
def main():
        # Taking the number of processes
        n = int(input("Enter number of process: "))
```

```
# Matrix for storing Process Id, Burst Time, Average Waiting Time & Average Turn Around Time.
A = [[0 for j in range(4)] for i in range(100)]
total, avg_wt, avg_tat = 0, 0, 0
print("Enter Burst Time:")
for i in range(n): # User Input Burst Time and alloting Process Id.
        A[i][1] = int(input(f"P{i+1}: "))
        A[i][0] = i + 1
for i in range(n): # Sorting process according to their Burst Time.
        index = i
        for j in range(i + 1, n):
                if A[j][1] < A[index][1]:
                        index = j
        temp = A[i][1]
        A[i][1] = A[index][1]
        A[index][1] = temp
        temp = A[i][0]
        A[i][0] = A[index][0]
        A[index][0] = temp
A[0][2] = 0 # Calculation of Waiting Times
for i in range(1, n):
        A[i][2] = 0
        for j in range(i):
                A[i][2] += A[j][1]
        total += A[i][2]
avg_wt = total / n
total = 0
```

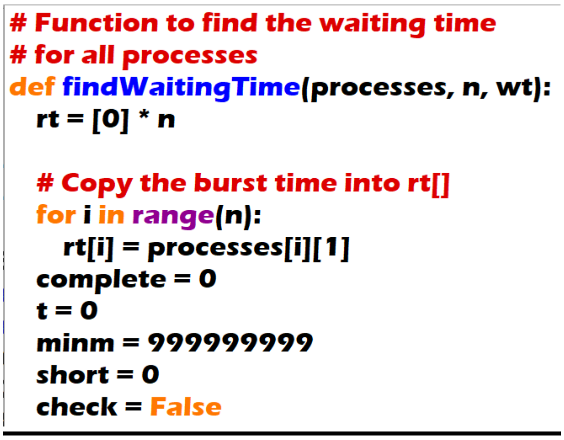


```
        # Calculation of Turn Around Time and printing the data.
        print("P      BT          WT          TAT")
        for i in range(n):
                A[i][3] = A[i][1] + A[i][2]
                total += A[i][3]
                print(f"P{A[i][0]}      {A[i][1]}    {A[i][2]}    {A[i][3]}")
        avg_tat = total / n
        print(f"Average Waiting Time= {avg_wt}")
        print(f"Average Turnaround Time= {avg_tat}")


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

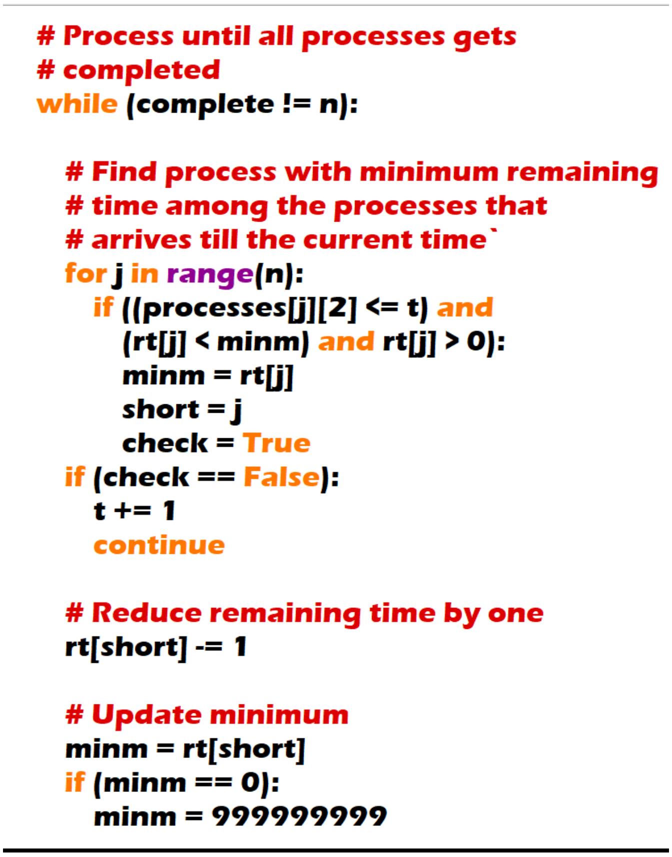


```
= RESTART: C:\Users\Lenovo\AppData\Local\Programs\Python\Python310\SJF experiment.p
Enter number of process: 6
Enter Burst Time:
P1: 6
P2: 2
P3: 8
P4: 3
P5: 4
P6: 0
P          BT         WT         TAT
P6         0          0          0
P2         2          0          2
P4         3          2          5
P5         4          5          9
P1         6          9          15
P3         8          15         23
Average Waiting Time= 5.166666666666667
Average Turnaround Time= 9.0
```

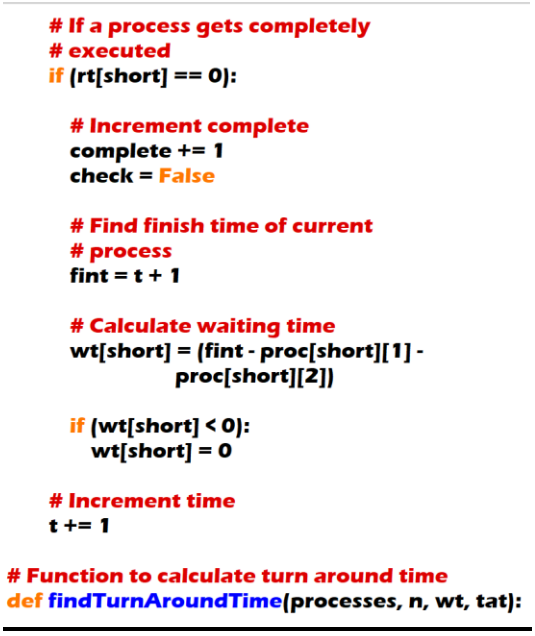
SHORTEST-REMAINING TIME FIRST :-



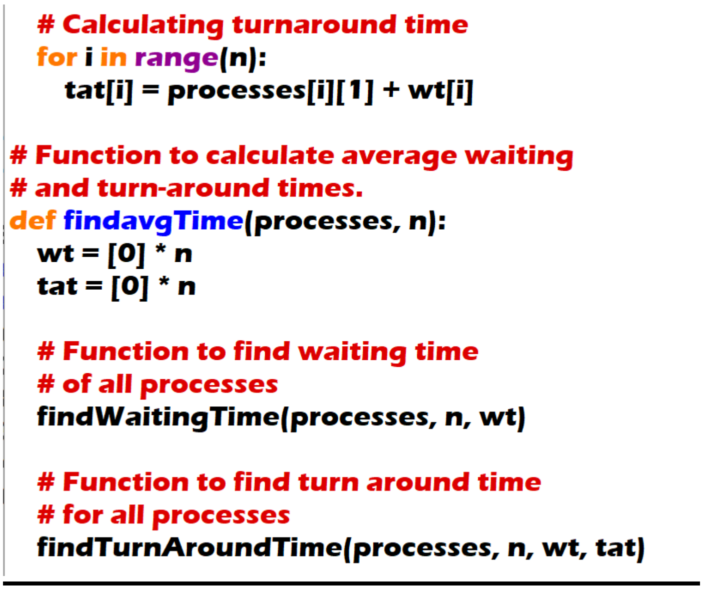
```
# Function to find the waiting time
# for all processes
def findWaitingTime(processes, n, wt):
  rt = [0] * n

  # Copy the burst time into rt[]
  for i in range(n):
    rt[i] = processes[i][1]
  complete = 0
  t = 0
  minm = 999999999
  short = 0
  check = False
```



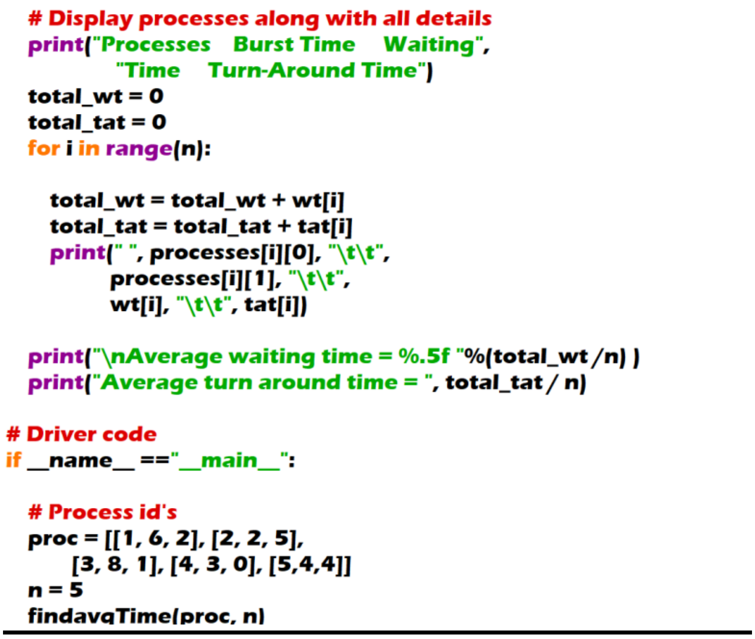
```
# Process until all processes gets
# completed
while (complete != n):

    # Find process with minimum remaining
    # time among the processes that
    # arrives till the current time`
    for j in range(n):
        if ((processes[j][2] <= t) and
            (rt[j] < minm) and rt[j] > 0):
            minm = rt[j]
            short = j
            check = True
    if (check == False):
        t += 1
        continue

    # Reduce remaining time by one
    rt[short] -= 1

    # Update minimum
    minm = rt[short]
    if (minm == 0):
        minm = 999999999
```



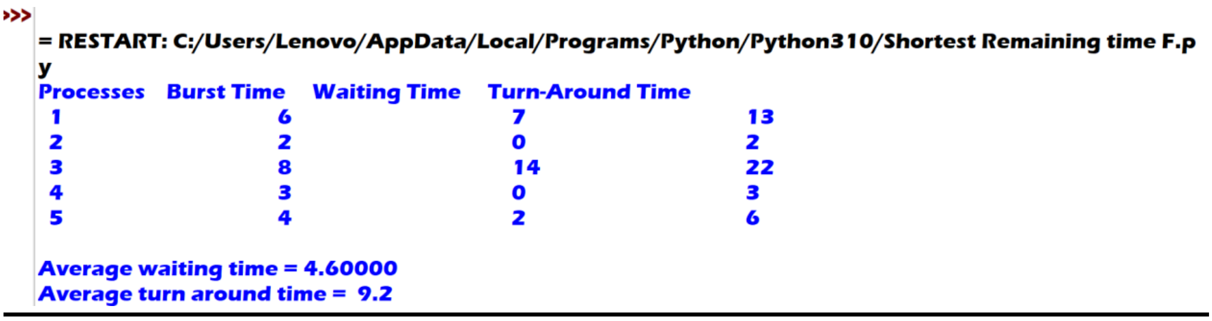
```
    # If a process gets completely
    # executed
    if (rt[short] == 0):

        # Increment complete
        complete += 1
        check = False

        # Find finish time of current
        # process
        fint = t + 1

        # Calculate waiting time
        wt[short] = (fint - proc[short][1] -
                         proc[short][2])

        if (wt[short] < 0):
            wt[short] = 0

    # Increment time
    t += 1

# Function to calculate turn around time
def findTurnAroundTime(processes, n, wt, tat):
```



```
    # Calculating turnaround time
    for i in range(n):
        tat[i] = processes[i][1] + wt[i]

# Function to calculate average waiting
# and turn-around times.
def findavgTime(processes, n):
    wt = [0] * n
    tat = [0] * n

    # Function to find waiting time
    # of all processes
    findWaitingTime(processes, n, wt)

    # Function to find turn around time
    # for all processes
    findTurnAroundTime(processes, n, wt, tat)
```



```
    # Display processes along with all details
    print("Processes   Burst Time    Waiting",
                "Time    Turn-Around Time")
    total_wt = 0
    total_tat = 0
    for i in range(n):

        total_wt = total_wt + wt[i]
        total_tat = total_tat + tat[i]
        print(" ", processes[i][0], "\t\t",
                  processes[i][1], "\t\t",
                  wt[i], "\t\t", tat[i])

    print("\nAverage waiting time = %.5f "%(total_wt /n) )
    print("Average turn around time = ", total_tat / n)

# Driver code
if __name__ =="__main__":

    # Process id's
    proc = [[1, 6, 2], [2, 2, 5],
            [3, 8, 1], [4, 3, 0], [5,4,4]]
    n = 5
    findavgTime(proc, n)
```



```
>>>
= RESTART: C:/Users/Lenovo/AppData/Local/Programs/Python/Python310/Shortest Remaining time F.p
y
Processes   Burst Time   Waiting Time   Turn-Around Time
 1                6               7               13
 2                2               0               2
 3                8               14              22
 4                3               0               3
 5                4               2               6

Average waiting time = 4.60000
Average turn around time =  9.2
```