Name: Abhinav Rajesh	**Roll No:** 03
Exp. No: 6	**Date:** 23/02/2022

SJF

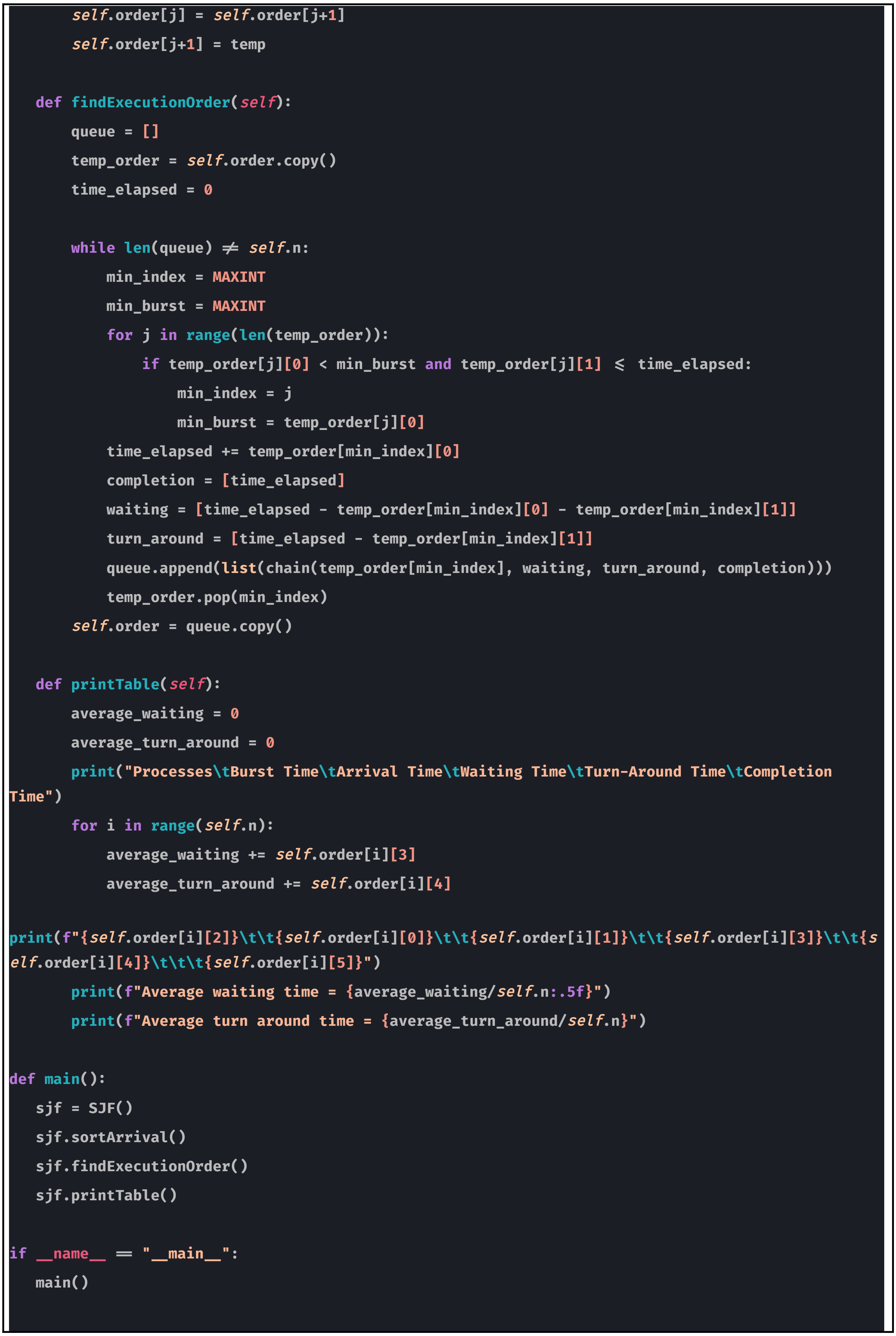
Aim: Write a program to implement SJF scheduling with arrival time(Non-pre-emptive)

Algorithm:

1. START
2. Define Class SJF
    a. Initialize constructor and get input from the user about number of processes and the burst and arrival time for these processes
    b. Create function sortArrival to sort the processes according to their arrival time
    c. Create a utility function swap to swap the orders
    d. Create function findExecutionOrder to compute the queue for the SJF
    e. Create function printTable to print the computed data in the required table format
3. Create main function
    a. Create an object "sjf" of the class "SJF"
    b. Call the methods sortArrival, findExecutionOrder and printTable in this order
4. STOP

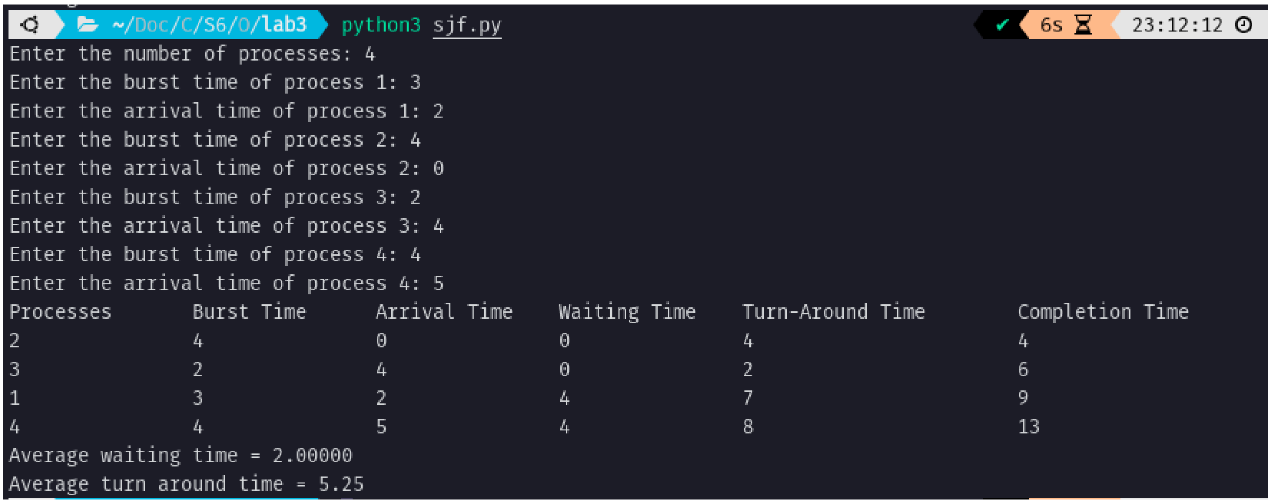
Program

```
from xmlrpc.client import MAXINT
from itertools import chain

class SJF:
    def __init__(self):
        self.order = []
        self.n = int(input("Enter the number of processes: "))

        for i in range(self.n):
            burst = int(input(f"Enter the burst time of process {i+1}: "))
            arrival = int(input(f"Enter the arrival time of process {i+1}: "))
            self.order.append([burst, arrival, i+1])

    def sortArrival(self):
        for i in range(self.n):
            for j in range(i, self.n - i -1):
                if self.order[j][1] == self.order[j+1][1]:
                    if self.order[j][0] > self.order[j+1][0]:
                        self.swap(j)
                elif self.order[j][1] > self.order[j+1][1]:
                    self.swap(j)

    def swap(self, j):
        temp = self.order[j]
```



```
            self.order[j] = self.order[j+1]
            self.order[j+1] = temp

    def findExecutionOrder(self):
        queue = []
        temp_order = self.order.copy()
        time_elapsed = 0

        while len(queue) != self.n:
            min_index = MAXINT
            min_burst = MAXINT
            for j in range(len(temp_order)):
                if temp_order[j][0] < min_burst and temp_order[j][1] <= time_elapsed:
                    min_index = j
                    min_burst = temp_order[j][0]
            time_elapsed += temp_order[min_index][0]
            completion = [time_elapsed]
            waiting = [time_elapsed - temp_order[min_index][0] - temp_order[min_index][1]]
            turn_around = [time_elapsed - temp_order[min_index][1]]
            queue.append(list(chain(temp_order[min_index], waiting, turn_around, completion)))
            temp_order.pop(min_index)
        self.order = queue.copy()

    def printTable(self):
        average_waiting = 0
        average_turn_around = 0
        print("Processes\tBurst Time\tArrival Time\tWaiting Time\tTurn-Around Time\tCompletion
Time")
        for i in range(self.n):
            average_waiting += self.order[i][3]
            average_turn_around += self.order[i][4]

print(f"{self.order[i][2]}\t\t{self.order[i][0]}\t\t{self.order[i][1]}\t\t{self.order[i][3]}\t\t{s
elf.order[i][4]}\t\t\t{self.order[i][5]}")
        print(f"Average waiting time = {average_waiting/self.n:.5f}")
        print(f"Average turn around time = {average_turn_around/self.n}")

def main():
    sjf = SJF()
    sjf.sortArrival()
    sjf.findExecutionOrder()
    sjf.printTable()

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

Output



```
~/Doc/C/S6/O/lab3  python3 sjf.py                                                    ✔  6s ⧗  23:12:12
Enter the number of processes: 4
Enter the burst time of process 1: 3
Enter the arrival time of process 1: 2
Enter the burst time of process 2: 4
Enter the arrival time of process 2: 0
Enter the burst time of process 3: 2
Enter the arrival time of process 3: 4
Enter the burst time of process 4: 4
Enter the arrival time of process 4: 5
Processes       Burst Time      Arrival Time    Waiting Time    Turn-Around Time        Completion Time
2               4               0               0               4                       4
3               2               4               0               2                       6
1               3               2               4               7                       9
4               4               5               4               8                       13
Average waiting time = 2.00000
Average turn around time = 5.25
```

Result:

Python Program to implement SJF scheduling Non-preemptive is compiled and executed successfully

Remarks: