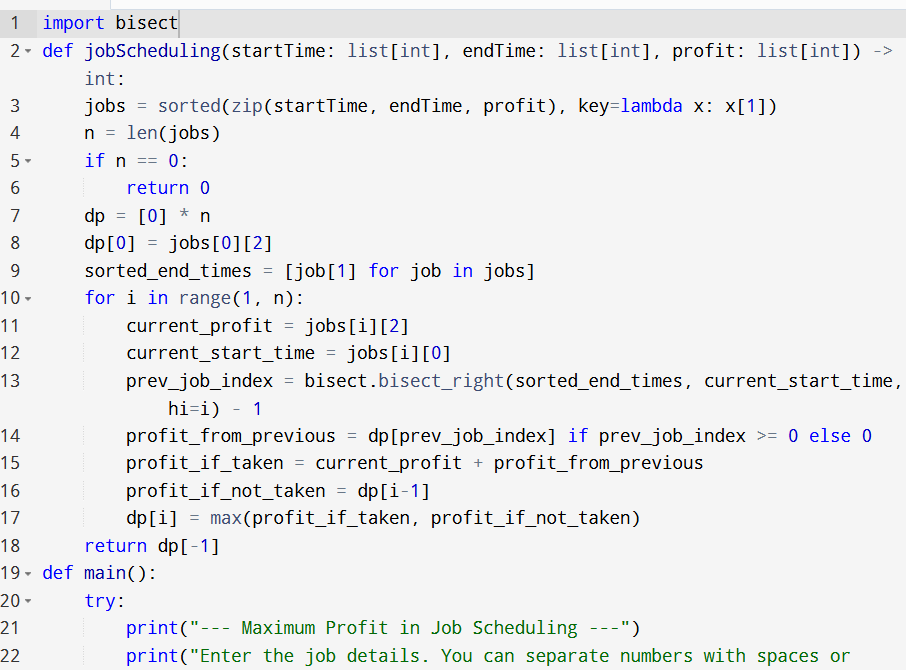
**5.4 Maximum Profit in Job Scheduling**

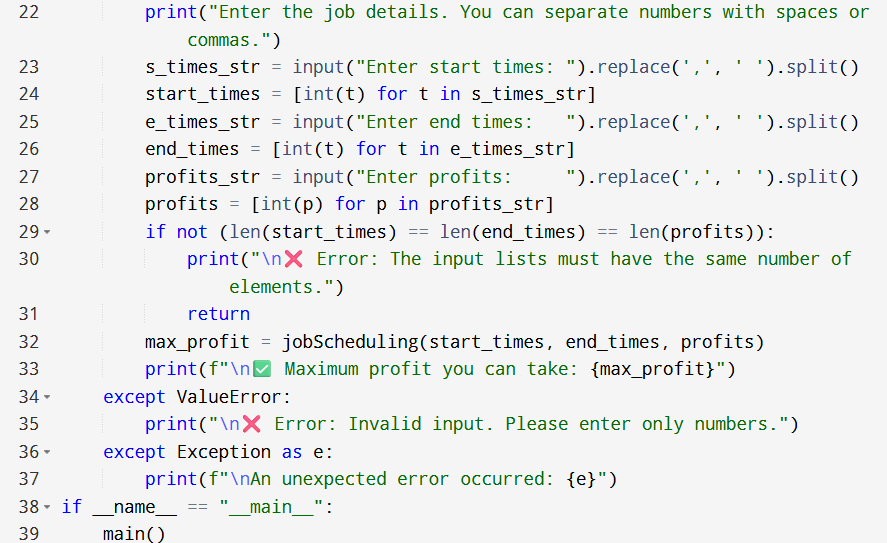
**Aim:** The aim is to find the maximum profit from a subset of non-overlapping jobs.

**Algorithm:**

1. Each job i has a start time sᵢ, finish time fᵢ, and profit pᵢ.
2. The aim is to select a subset of jobs such that no two jobs overlap,  
   and the total profit is maximized.
3. First, sort the jobs by their finish times in non-decreasing order:  
   f₁ ≤ f₂ ≤ … ≤ fₙ.
4. Let dp[i] represent the maximum profit achievable by considering the first i jobs.
5. For each job i, we have two choices:
6. Include job i → Add its profit pᵢ and also add the best profit from all jobs that finish before sᵢ.
7. Exclude job i → Just take dp[i-1].
8. To efficiently find the last job that finishes before sᵢ,  
   perform binary search on the sorted finish times.  
   Let L(i) = index of the last non-overlapping job before job i.
9. The recurrence relation becomes:  
   dp[i] = max( pᵢ + dp[L(i)], dp[i-1] ).
10. Base case:  
    dp[0] = 0 (no jobs, no profit).
11. Fill dp[1..n] iteratively using the recurrence.
12. The answer is dp[n], i.e., the maximum profit from all n jobs.

**Program:**

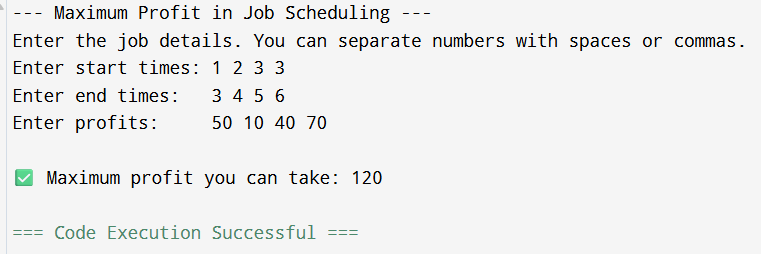
****



**Input:**

* Enter the job details. You can separate numbers with spaces or commas.
* Enter start times:

**Output:**

****

**Result:** Thus, the program is executed successfully and output is verified.

**Performance analysis:**

* Time Complexity: O(nlogn)
* Space Complexity: O(n).