Job_scheduling :

This code defines a function `earliest_due_date` that schedules jobs on machines based on their due dates to minimize the total lateness. Here's an explanation of the code:

1. `jobs.sort(key=lambda x: x[1])`: Sorts the list of jobs based on their due dates (the second element in each tuple), with the jobs having the earliest due dates appearing first in the list.

2. `n = len(jobs)`: Gets the number of jobs.

3. `schedule = []`: Initializes an empty list to store the schedule.

4. `completion_times = [0] * n`: Initializes a list `completion_times` with `n` elements, each representing the completion time of a machine (initialized to 0).

5. `for job in jobs:`: Iterates over each job in the sorted list of jobs.

6. `duration, due_date = job`: Unpacks the duration and due date of the current job.

7. `min_idx = completion_times.index(min(completion_times))`: Finds the index of the machine with the earliest completion time by finding the minimum value in `completion_times`.

8. `start_time = max(completion_times[min_idx], due_date - duration)`: Calculates the start time of the job on the selected machine, ensuring that it starts after the due date minus the duration if necessary to avoid lateness.

9. `completion_times[min_idx] = start_time + duration`: Updates the completion time of the selected machine to reflect the completion of the current job.

10. `schedule.append((min_idx, start_time, start_time + duration, due_date))`: Appends a tuple representing the scheduled job to the `schedule` list, including the machine index, start time, end time, and due date.

11. `return schedule`: Returns the schedule as a list of tuples.

12. `jobs = [(2, 5), (3, 6), (1, 8), (2, 9), (3, 10)]`: Defines an example list of jobs, where each job is represented by a tuple `(duration, due_date)`.

13. `schedule = earliest_due_date(jobs)`: Calls the `earliest_due_date` function with the example list of jobs to generate the schedule.

14. `for idx, start, end, due in schedule:`: Iterates over the scheduled jobs in the `schedule` list.

15. `print(f"Job {idx + 1}: Start Time={start}, End Time={end}, Due Date={due}")` : Prints the details of each scheduled job, including the job index (1-based), start time, end time, and due date.

Overall, this code efficiently schedules jobs on machines based on their due dates to minimize lateness, providing insight into job scheduling algorithms used in production planning and operations management.