# Name: Shashank K

# SE21UCSE198

# DAA class assignment

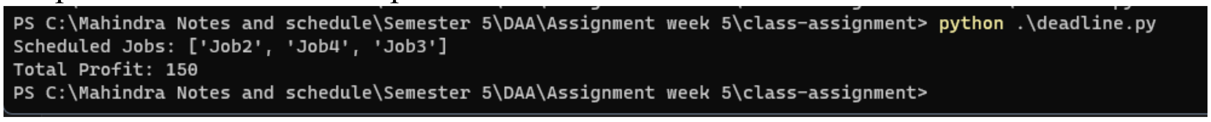
## Program

```python
def deadline_driven_greedy(jobs):
    sorted_jobs = sorted(jobs, key=lambda x: (x['profit'], -x['deadline']),
reverse=True)

    max_deadline = max(job['deadline'] for job in sorted_jobs)

    schedule = [-1] * max_deadline

    total_profit = 0
    for job in sorted_jobs:
        for i in range(job['deadline'] - 1, -1, -1):
            if schedule[i] == -1:
                schedule[i] = job['id']
                total_profit += job['profit']
                break

    return schedule, total_profit

jobs = [
    {'id': 'Job1', 'profit': 50, 'deadline': 2},
    {'id': 'Job2', 'profit': 60, 'deadline': 1},
    {'id': 'Job3', 'profit': 20, 'deadline': 3},
    {'id': 'Job4', 'profit': 70, 'deadline': 2},
    {'id': 'Job5', 'profit': 30, 'deadline': 1}
]

schedule, profit = deadline_driven_greedy(jobs)
print("Scheduled Jobs:", schedule)
print("Total Profit:", profit)
```

## Approach

Deadline based activity selection results in empty time periods being left out. Since we are not making use of this time it would be a waste of CPU power. My approach to this problem is to sort the jobs based on profit as first priority and deadline as second priority. This helps us solve the time gap issue. The simple deadline scheduling algorithm doesn't consider less profitable options but with this algorithm we can choose some unlikely to be chosen jobs to fill up empty time thus filling up our time array.

Output for the mentioned input



```
PS C:\Mahindra Notes and schedule\Semester 5\DAA\Assignment week 5\class-assignment> python .\deadline.py
Scheduled Jobs: ['Job2', 'Job4', 'Job3']
Total Profit: 150
PS C:\Mahindra Notes and schedule\Semester 5\DAA\Assignment week 5\class-assignment>
```