Schedule Optimization Model

This Python program constructs a **university course timetable** using **Google OR-Tools' CP-SAT solver**. It assigns course sections to time slots and rooms while satisfying hard constraints and minimizing violations of soft preferences.

Data Inputs

Loaded from Excel (`data.xlsx` and `sample_schedule.xlsx`):

- `classrooms`: Room metadata (Room name, Size, Type, Building)
- `slots`: Valid time slot ranges by day and type (`Theory`, `Lab-2hr`, `Lab-3hr`)
- `faculties`: Instructor initials, designations, and building preferences
- `sample_schedule`: Raw course scheduling preferences

Decision Variables

- `room_vars[cid]`: Assigned room index for course `cid`
- `timeslot_vars[cid][i]`: Assigned timeslot index for `i`-th day of course `cid`

Hard Constraints (Must be satisfied)

ID	Constraint	Description
H1	**Valid Room Type**	Labs must be in lab rooms; theory in any
H2	**Room Capacity**	Assigned room must accommodate enrolled students
H3	**Time Slot Type Match**	Based on course duration and lab/theory tag
H4	**Room-Time Clash**	No two courses in the same room at the same time
H5	**Instructor Conflict**	A faculty cannot teach two classes simultaneously

Soft Constraints (Minimized using penalties)

Each violation adds a weighted Boolean penalty:

ID	Constraint	Condition	Penalty
S1	**Professors/Deans/Heads get fixed room & timeslot**	Room or timeslot not matched	`100`
S2	**Adjunct Faculty get fixed timeslot**	Timeslot not matched	`100`
S3	**Weekly Idle Time > 11 hrs**	Faculty idle time total > 660 minutes	`90`
S4	**Main Building for Professors, Associate Professors, and Adjuncts**	Room not in Main	`80`
S5	**Back-to-Back Classes in Different Buildings**	≤10 min gap but rooms differ	`70`
S6	**Office-Building Preference (FUB, AB1, AB3)**	Room not in faculty's office building	`60`
S7	**Daily Idle Time > 4 hrs**	Faculty idle time span exceeds limit	`50`
S8	**Room Preference** (if given)	Preferred room not assigned	`50`
S9	**Others' Time Preference**	Preferred time not matched	`20`

Objective Function

```
model.Minimize(sum(weight * bool_var for (bool_var, _), weight in zip(penalty_vars, penalty_weights)))
```

The solver minimizes the weighted sum of all soft constraint violations.

Output

- Prints constraint violations with reasons and weights
- Writes final feasible schedule to `final_schedule_output.xlsx`
- Reports room usage and matched/unmatched preferences for each course