Challenge: Weighted Scheduling Problem

In this lesson, you will solve the classic problem of weighted scheduling using dynamic programming.

We'll cover the following

Problem statement

You have one auditorium, and only one class can take place at the time; however, you have different options for these classes. Each class, *c*, is characterized by its start time, end time, and total utility we can get out of this class. The utility here could be the number of attendants of the class; the more attendants, the more money you can make as an auditorium owner. Here is your problem statement:

Given *n* number of classes, you have to find a schedule that maximizes the utility of an auditorium.

Input

A list of classes, where each class is a tuple of three numbers. The first number denotes the start time, the second number denotes the end time, and the third number denotes utility value.

```
schedule = [(0,2,25), (1,5,40), (6,8,170), (3,7,220)]
```

Note that start and end time are strictly increasing integers.

Output

Your algorithm will return an integer, denoting the maximum possible utility

achievable from the given schedule.

```
WeightedSchedule(schedule) = 245
```

Coding challenge

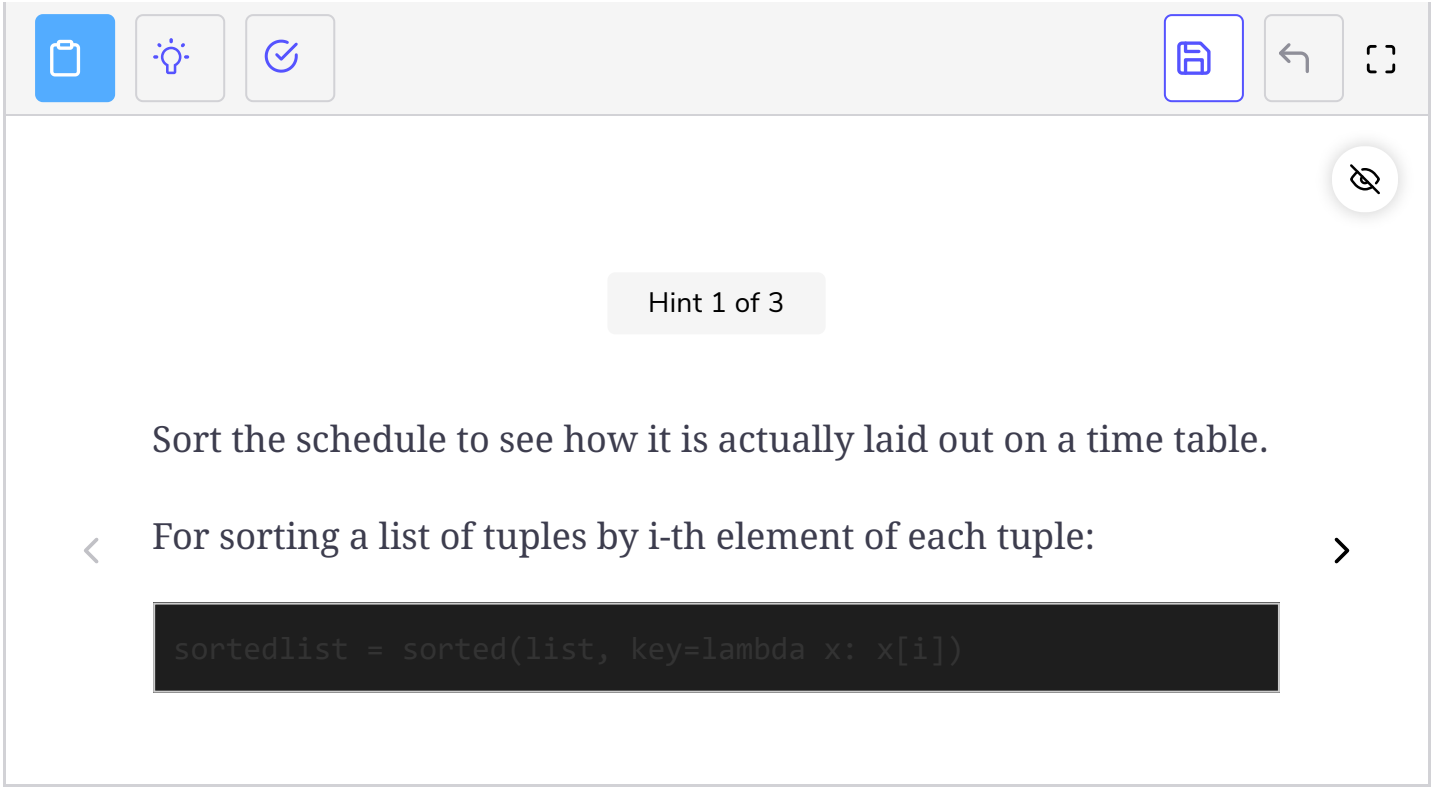
You might find the function `lastConflict()` useful. Given the index of a job and a list of jobs, it finds the last job that does not conflict with the current job given by the index.

Jot down a few examples and try to solve them manually to get a hang of the problem, then build the solution. This is a slightly more difficult problem, so take your time. Best of luck!

```
# Given the index of the class and the list of schedule, this function returns the last class that
def lastConflict(index, schedule, isSorted = False):
  if not isSorted:
    schedule = sorted(schedule, key=lambda tup: tup[1])
  for i in range(index, -1, -1):
    if schedule[index][0] >= schedule[i][1]:
      return i
  return None

def WeightedSchedule(schedule):
  # write your code here
  return 0

stressTesting = True
```



In the next lesson, we will review some solutions to this challenge.