Q. Programming Pre-Screen

Conference room scheduling.
Find the nearest open conference room for a team in which a team can hold its meeting. Given n team members with the floor on which they work and the time they want to meet, and a list of conference rooms identified by their floor and room number as a decimal number, maximum number of people it fits and pairs of times they are open - find the best place for the team to have their meeting. If there is more than one room available that fits the team at the
chosen time then the best place is on the floor the closest to where the team works.

E.g.
rooms.txt
7.11,8,9:00,9:15,14:30,15:00
8.23,6,10:00,11:00,14:00,15:00
8.43,7,11:30,12:30,17:00,17:30
9.511,9,9:30,10:30,12:00,12:15,15:15,16:15
9.527,4,9:00,11:00,14:00,16:00
9.547,8,10;30,11:30,13:30,15:30,16:30,17:30

Input: 5,8,10:30,11:30 # 5 team members, located on the 8th floor, meeting time 10:30 - 11:30

Output:
9.547

Please explain: how you solved the problem and how it would behave based on the different parameters (number of team members, longer meeting times, many rooms with random booking times). How would you test the program to ensure it always produced the correct results?

For extra credit, can you improve the solution to split the meeting across more than one room if say only one room is available for a fraction of the meeting and another room is free later to hold the remainder of the meeting during the set time. If you want to make this more powerful - assume that the number of room splits can happen in proportion to the length of the meeting so that say if a meeting is 8 hrs long then the algorithm could schedule it across say up to 4 rooms if a single room was not available for the whole time.

Ans:
```
def roomSearch(n,f,s,e):
    row = data.loc[(data['Start Time']==s) & (data['End Time']==e)]
    if (n < row['Max People'].array):
        return row['Floor'],row['Room']
    else:
        print('No room available')
```

-Floor,Room,Max People,Start Time,End Time

- 7,11,8,0900,0915
- 7,11,8,1430,1500
- 8,23,6,1000,1100
- 8,23,6,1400,1500
- 8,43,7,1130,1230
- 8,43,7,1700,1730
- 9,511,9,0930,1030
- 10,501,9,1030,1130
- 9,511,9,1200,1215
- 9,511,9,1515,1615
- 9,527,4,0900,1100
- 9,527,4,1400,1600
- 9,547,8,1030,1130
- 9,547,8,1330,1530
- 9,547,8,1630,1730

Function takes number of people n, floor f, start time s and end time e as inputs.
First locates based on the start time and end time.

Then checks if it has the capacity for people and returns the floor and room. Super basic.

For next part to pick nearest room, explain we can do a binary search to find nearest floor in case there is more than 1 room on different floor.

There will always be room to improve. I believe in refining and tuning the product to best meet the goals. For scheduling a conference, we can refine the code further.

1. Use Greedy algorithm to further address splitting the meeting in different rooms for the entire day. Nearest floor will again take priority as it is good time management

2. Instead of printing no rooms available, we can suggest conference rooms that have similar time slot availability.

3. If there are two conference rooms available on same floor, allocate the room that fills up first. Like two rooms available with max people of 6 & 7 and you have only have 5 people allocate the room with max people of 6.

FYI – please see the code below & will attached the code file.

Yasar

770-666-2803

Code - First Part

In [47]:

```
import pandas as pd #importing libraries
```

In [48]:

```
data = pd.read_csv('room.txt', sep=",") #reading from .txt file, room is the data given on the test
```

In [49]:

```
data.head(5) #looking at the top 5 rows of data
```

Out[49]:

	Floor	Room	Max People	Start Time	End Time
0	7	11	8	900	915
1	7	11	8	1430	1500
2	8	23	6	1000	1100
3	8	23	6	1400	1500
4	8	43	7	1130	1230

In [50]:

```
def roomSearch(n,f,s,e):
    #A function that takes:
    # n - number of people
    # f - floor
    # s - start time
    # e - end time

    row = data.loc[(data['Start Time']==s) & (data['End Time']==e)] #locate time slot by start and end time
    if (n < row['Max People'].array):
        return row['Floor'],row['Room'] #return room and floor number
    else:
        print('No room available')
```

In [51]:

```
f,r = roomSearch(5,8,1030,1130) # 5 people on floor 8 between 10:30 and 11:30
```

In [52]:

```
print('Floor: ',f.array[0])
print('Room: ',r.array[0])
```

```
Floor:  9
Room:  547
```

The above method is not refined as it returns the room that is first on the list based on how it is stored. If there were two floors then it would return the floor that was stored first which may not always be the optimized solution. We will address that issue in the next few lines of code

Nearest Floor

In [53]:

```
data1 = pd.read_csv('room1.txt', sep=",") #reading from .txt file, we have added a new line to have two floors with same time slots
```

In [54]:

```
data1
```

Out[54]:

	Floor	Room	Max People	Start Time	End Time
0	7	11	8	900	915
1	7	11	8	1430	1500
2	8	23	6	1000	1100
3	8	23	6	1400	1500
4	8	43	7	1130	1230
5	8	43	7	1700	1730
6	9	511	9	930	1030
7	10	501	9	1030	1130
8	9	511	9	1200	1215

	Floor	Room	Max People	Start Time	End Time
9	9	511	9	1515	1615
10	9	527	4	900	1100
11	9	527	4	1400	1600
12	9	547	8	1030	1130
13	9	547	8	1330	1530
14	9	547	8	1630	1730

Line 8 or index 7 has a new floor 10 with a matching time slot to 9.547

In [55]:

```
def roomSearch1(n,f,s,e):
    row = data1.loc[(data1['Start Time']==s) & (data1['End Time']==e)]
    if (n < row['Max People'].array):
        if row.shape[0] > 1:
            floors = row.Floor.tolist()
            diff_func = lambda x : abs(x - f)
            nearest_floor = min(floors, key=diff_func)
            target = row.loc[row['Floor']==nearest_floor]
            return target['Floor'],target['Room']


    else:
        print('No room available')
```

Scenario 1 - 5 people on floor 8 between 10:30 and 11:30

In [69]:

```
f,r = roomSearch1(5,8,1030,1130) # 5 people on floor 8 between 10:30 and 11:30
print('Floor: ',f.array[0])
print('Room: ',r.array[0])
```

```
Floor:  9
Room:  547
```

Scenario 2 - 5 people on floor 9 between 10:30 and 11:30

In [72]:

```
f,r = roomSearch1(5,9,1030,1130) # 5 people on floor 9 between 10:30 and 11:30
print('Floor: ',f.array[0])
print('Room: ',r.array[0])
```

```
Floor:  9
Room:  547
```

Scenario 3 - 5 people on floor 10 between 10:30 and 11:30

In [73]:

```
f,r = roomSearch1(5,10,1030,1130) # 5 people on floor 10 between 10:30 and 11:30
print('Floor: ',f.array[0])
print('Room: ',r.array[0])
```

```
Floor:  10
Room:  501
```

Scenario 4 - 5 people on floor 11 between 10:30 and 11:30

In [74]:

```
f,r = roomSearch1(5,11,1030,1130) # 5 people on floor 11 between 10:30 and 11:30
print('Floor: ',f.array[0])
print('Room: ',r.array[0])
```

```
Floor:  10
Room:  501
```