Break Domain Generator

Description:

Imagine that you are working on a program that will be used for the generation of schedules based on several constraints. Namely, you are trying to create schedules for the university classes, hence you need to place some breaks. But you need a way of identifying which breaks should be placed and that is why you decided to create a domain reduction algorithm that will return a list of possible breaks that can be placed.

data.json file structure:

You are given a JSON file of break templates.

```
‘1’: [
        {
                'id': '8af1d0e6808fb86501809239989700 1f',
                'break_duration': 15,
                'start_time_type': 'RELATIVE_TO_CLASS_START',
                'start_times': [
                        '01:30:00',     10:30
                        '01:45:00',     10:45
                        '02:00:00'      11:00
                ]
        },
        {
                'id': '8af1d0e6808fb8650180923998970020',
                'break_duration': 30,
                 'start_time_type': 'RELATIVE_TO_CLASS_END',
                 'start_times': [
                        ‘06:00:00’      07:00
                        '03:00:00',     10:00
                        '03:30:00'      09:30
                ]
        }
]
```

`1` - stands for span_id
Span consists of templates. Each template is a potential break that should be placed. Our example span consists of 2 templates, hence for this case, we would like to place two 2 breaks.

Each break template contains the following information:

`id` - break template id

`break_duration` - how long should break last

`start_time_type` - information about should the break be placed relative to class start time or relative to end time. Can have two values {'RELATIVE_TO_CLASS_START', 'RELATIVE_TO_CLASS_END'}.

*Let's imagine that the class will; start at 9:00 and end at 15:00. Then the first break can be placed at **9:00 + x** and the second one can be placed at **15:00 - x.***

`start_times` - relative time that break can be placed.
For example, based on the above-mentioned example the first break can be placed at [9:00:00 + 01:30:00, 9:00:00 + 01:45:00, 9:00:00 + 02:00:00] = [10:30:00, 10:45:00, 11:00:00]

Note that start_times for the templates can overlap.

Constraints:

1. While implementing a task take into consideration the following constraints. Breaks cannot be placed too close to each other. Choose **2 hours** as an initial starting point for the break spacing. If there is a value inside the domain that obeys the spacing rule then return those values as a domain for the break if not decrease the spacing by **15 minutes** until you get a domain value that will obey the rule. Always return a list of lists of domain values. If there are several values that obey the rule return all those values.
2. The break cannot be placed before or after class start time. The required thing is that the break should be fully inside of the class start and end times.

Input:

As an input, your program will receive **start and end times** for the class and the **span_id**.

Output:

As an output, your program should return **all possible breaks that obey the given constraints.**

Acceptance criteria:

1. Implement a Break class that will be an actual break that needs to be placed. Break class should contain information about the **break start time**, **break end time**, and **break duration**. Feel free to add more fields if you find them important for your program.
2. BreakDomainGenerator should return list<list<Breaks>>(list of break lists) that will obey the given constraints.

For example:

As an input we receive **span_id =** ***'1', class_start_time=`09:00:00`, class_end_time=`17:00:00`***

Span is the above-provided example.

> *The first break can be placed at* ***10:30:00 or 10:45:00 or 11:00:00***
> *The second break can be placed at* ***14:00:00 or 13:30:00 or 11:00:00***

The output of the program should be:

[[Break(10:30:00, 10:45:00, 15), Break(14:00:00, 14:30:00, 30)],
[Break(10:30:00, 10:45:00, 15), Break(13:30:00, 14:00:00, 30)],
[Break(10:45:00, 11:00:00, 15), Break(14:00:00, 14:30:00, 30)],
[Break(10:45:00, 11:00:00, 15), Break(13:30:00, 14:00:00, 30)]]

[Break(10:45:00, 11:00:00, 15), Break(11:00:00, 11:30:00, 30)] is not a valid domain value as it breaks minimum spacing rule. **(11:00:00 + 02:00:00 = 13:00:00 and 13:00:00 > 11:00:00)**

/src
/class
Break.py
Duration.py
Domaingenerator.py
program.py
/utils
/tests