CS 1110 Spring 2020, Assignment 5: Tre1110, 3rd time's the charm

Updates:

The assignment itself, with updates marked in orange, begins on the next page. On this "page 0", we also document the time, location, and nature of the updates, in reverse chronological order,

- Sun May 3, 6pm Ithaca time:
  - Specification for `Month.__init__` rewritten to avoid ambiguity between references to the month-number attribute `month_num` and the parameter `month`. This has now been fixed in the zip file posted to the course website.
  - Clarification added to Section 4.6 "Thought Question to Answer": we mentioned one problem with having `add_task` being a non-method function. In your answer, you do not need to talk about other problems with `add_task` not being a method.

CS 1110 Spring 2020, Assignment 5: Tre1110, 3rd time's the charm*

1 Motivation: Get all the things done, even if not all at once!

Many to-do items can be scheduled for non-contiguous blocks of time. For example, one might schedule work on revising a paper from 10:00-13:00, and then again from 19:00-22:00. We'll create a simple model of such tasks with a new subclass of class Task from A3[1] called SplittableTask.[2]

Contents

2 Rules (all the same as in A4)

2.1 Resubmission after grading feedback is (conditionally) allowed!

You are welcome to resubmit *one* of A4 or A5 after receiving grading feedback; further details to be arranged.

We are arranging for resubmission because we are concerned that there may currently be huge variation in learning situations of students across the course, and we want to reduce the stress of having the deadlines for two heavily-weighted assignments coming at you. We hope that resubmission will allow you to focus on improvement and mastery of the material over a more gradual period of time.

However, many of the course staff are themselves carrying heavy burdens, and so we aren't in a position to be able to offer regrading of resubmissions for both assignments; hence the limit to one of A4 or A5.

*Authors: Lillian Lee and Will Xiao.

[1]This assignment, A5, does not have any explicit dependencies on A4.

[2]We tried in vain to come up with a shorter name that wouldn't be prone to typos (one of us is prone to mis-remembering"STask" as "STack", for example). Sorry. Typing "SplittableTask" wasn't that fun for us, either.

2.2 You need to re-group (so to speak) on CMS, regardless of prior groupings

You may work alone or with just one other person, who can be someone you've grouped with before in CS1110, or a different person.

If you are partnering, regardless of whether you were grouped in a previous assignment, the two of you must still form a new group *specifically for this assignment* on CMS before submitting.[3]

If your partnership dissolves, see the course Academic Integrity description about "group divorce" for what to do.

2.3 (Dis-)allowed collaborations and documentation requirements

Our policies are laid out in full on the course Academic Integrity page, but we re-state here **the main rules**: where "you" means you and, if there is one, your one CMS-registered group partner,

1. Never look at, access or possess any portion of another group's work in any form.
2. Never show or share any portion of your work in any form to anyone except a member of the course staff.
3. Never request solutions from outside sources; for example, on online services like StackOverflow.
4. DO specifically acknowledge by name all help you received, whether or not it was "legal" according to (1)-(3).

3 Due dates

(a) If you are partnering: well before submission, follow the "How to form a group" instructions on the course CMS Usage Guide. Both parties need to act on CMS in order for the grouping to take effect.

(b) By 2pm on Sun May 10, submit whatever you have done at that point CMS, following steps 1-3 in the "Updating, verifying, and documenting assignment submission" section of the course Assignments page. It is OK if you haven't finished working on it yet.[4]

(c) By **11:59pm**[5] **on Sun May 10**, make your final submission of files `a5.py` and `a5_thought_question.py`, again following the aforementioned steps 1-3.

4 Tasks

4.1 Learning goals

The main concepts tested in this assignment are:

- Writing and working with (sub)class methods
- While-loops.

Hence, *we reserve the right to assign no credit for code that isn't fundamentally based on while-loops, or does not use while-loops in the way we ask for, when we request a while-looped-based implementation*, even if the code fulfills the specification.

4.2 Overview

The files needed for the assignment can be downloaded from the following URL:

`http://www.cs.cornell.edu/courses/cs1110/2020sp/assignments/assignment5/a5.zip`. The zip file includes `a3_classes.py` from A3, since we'll be using the Task class again. You may wish to review the A3 handout and solutions, all available on the course Schedule page.

In our simple model of splittable tasks, when we schedule a SplittableTask for a Day, we fill in as many empty time slots of that Day with the SplittableTask as possible, earliest time slots first.

[3] This links your submission "portals".

[4] The 2pm checkpoints provide you a chance to alert us if any problems arise. Since you've been warned to submit early, do not expect that we will accept work that doesn't make it onto CMS on time, for whatever reason. There are no so-called "slipdays" and there is no "you get to submit late at the price of a late penalty" policy. Of course, if some special circumstances arise, contact the instructor(s) immediately.

[5] **PLUS** a 9-hour grace period (anyone can take advantage of it, but the intent is to account for students in "distant" time zones relative to Ithaca or for connection issues).

Your major task is to write the five incomplete method bodies in `a5.py` for classes SplittableTask and Month, following the implementation directions given as comments in the file.

When done, you'll be able to run the simple scheduling app `a5_app.py`. Here's a sample run, showing how Tasks and SplittableTasks can be scheduled in a calendar. Lines 47-57 show a request to schedule an 8-hour SplittableTask, "odds and ends", for May 12th, and lines 41-45 and 60-67 show that "odds and ends" was automatically scheduled in the gaps around the three other tasks on that date.

```
1  [ljl2@event_horizon A5] python a5_app.py
2  Welcome to the Tre1110 planner redux!
3  Here, you can plan out an entire month.
4  What month would you like to plan for? (Please enter an int in 1..12.)
5  > 5
6  From your implementation, the month you have chosen (May) has 31 days.
7  If this seems off, please check it!
8
9  Type 'print <day_number>' to view your schedule for that day, or 'print all' to view your whole month at once.
10 To add a task to the planner, type 'add task'.
11 Type 'q' to quit.
12 What would you like to do?
13 > add task
14 What type of task would you like to add? Enter either 'r' (for regular) or 's' (for splittable)'.
15 > r
16 What is the name of your task?
17 > on-call hours for volunteer work
18 How many hours will your task take (integers please)?
19 > 5
20 What day number would you like to add this task to?
21 > 12
22 What hour will your task start (24 hour time please)?
23 > 13
24 Successfully added!
25 What would you like to do now? (add task, print <day num>, print all, q)
26 > add task
27 What type of task would you like to add? Enter either 'r' (for regular) or 's' (for splittable)'.
28 > r
29 What is the name of your task?
30 > hackathon
31 How many hours will your task take (integers please)?
32 > 8
33 What day number would you like to add this task to?
34 > 12
35 What hour will your task start (24 hour time please)?
36 > 0
37 Successfully added!
38 What would you like to do now? (add task, print <day num>, print all, q)
```

```
[eliding the addition of two more  regular tasks]
```

```
39 What would you like to do now? (add task, print <day num>, print all, q)
40 > print all
41 Here is your calendar for the month of May:
42 5/12:
43 0:00-8:00      hackathon
44 13:00-18:00    on-call hours for volunteer work
45 19:00-20:00    dinner
46
47 What would you like to do now? (add task, print <day num>, print all, q)
48 > add task
49 What type of task would you like to add? Enter either 'r' (for regular) or 's' (for splittable)'.
50 > s
51 What is the name of your task?
```

```
> odds and ends
How many hours will your task take (integers please)?
> 8
What day number would you like to add this task to?
> 12
The task was successfully fully added!
What would you like to do now? (add task, print <day num>, print all, q)
> print all
Here is your calendar for the month of May:
5/12:
0:00-8:00       hackathon
8:00-13:00      odds and ends
13:00-18:00     on-call hours for volunteer work
18:00-19:00     odds and ends
19:00-20:00     dinner
20:00-22:00     odds and ends

What would you like to do now? (add task, print <day num>, print all, q)
> q
Goodbye and good luck getting all your tasks done!
```

4.3 Classes SplittableTask and Month

Here, for your reading convenience, are the class docstrings/invariants for two classes whose methods you will be implementing.

```python
class Month():
    """
    An instance represents one of January, February, March, ..., December.

    Instance attributes:
        month_num: The number of this month [int in 1..12]
        day_list: The days in this month [list of a3_classes.Day objects]
            Constraints on day_list:
            - The number of Day objects in day_list equals the number of days
              in the month corresponding to this Month.
              Note: in the Land of A5, February always has exactly 28 days.
            - For each Day in day_list, the Day object's name has format
              "<month number>/<day number>".
              Example: if this Month's month_num is 5, the item at index 0 in
                `day_list` should have name "5/1", for the 1st day of May.
            - The Day objects are in ascending order by day number as
              listed in the Day's names; e.g., the Day with name "5/9"
              immediately the precedes the Day with name "5/10".
    """
```

```python
class SplittableTask(a3_classes.Task):
    """
    An instance represents a task that can be scheduled within a Day.

    SplittableTasks do not have to be scheduled for a single block of consecutive
    hours, but rather can be split up and scheduled in multiple smaller intervals.

    Instance attributes (IN ADDITION TO THOSE IN THE PARENT Task CLASS):
        time_unscheduled: the remaining amount of time left in this task that
                          has yet to be scheduled within a Day
                          [int >= 0, and no more than this Task's length]

    Class invariant:
    the amount of time this SplittableTask has been budgeted for in some
       presumed calendar of interest,
    plus this SplittableTask's time_unscheduled,
    equals this SplittableTask's length.
    """
```

And for reference, here's the description of SplittableTask's parent, Task.

```python
class Task:
    """
    An instance is a single task to be completed in one day.

    Instance Attributes:
        name (nonempty string): the name of this Task
        length (int in [1..24]): how long this Task takes to complete (i.e how
            many time slots it occupies)
    """
```

Note that the phrase "how many time slots it occupies" is intended to mean, "how many time slots it *would* occupy if it were to be fully scheduled."

4.4 Test cases

We've provided test cases in `a5_tests.py`. Looking at the test-cases code while you read over the corresponding function's specification may bring you further clarity on what each method is supposed to do.

You are free to create more test cases, and encouraged to do so if you have the time, but we are not asking you to submit your test cases. Nonetheless, *we reserve the right to grade your code in part on its output on different test cases than what we gave you.*

4.5 The methods to write

Again, follow the implementation directions given as comments in the code.

4.5.1 `__init__` for SplittableTask

```python
    def __init__(self, n, time_needed):
        """
        Initializer: a new, completely unscheduled SplittableTask with name `n`
        and length `time_needed`

        Preconditions:
            `n`: nonempty string
            `time_needed`:  int in 1..24
        """
```

4.5.2 `isAllScheduled` for SplittableTask

```python
    def isAllScheduled(self):
        """
        Returns: True if this SplittableTask is fully scheduled, False otherwise.

        A SplittableTask is fully scheduled if it has 0 unscheduled time left.
        """
```

4.5.3 `updateUnsched` for SplittableTask

```python
    def updateUnsched(self, d, hr):
        """
        Schedules one of this SplittableTask's unscheduled hours for Day `d` at
        time `hr`. Note that `d` thus is altered.
        Preconditions:
            This SplittableTask has at least one unscheduled hour left.
            `hr` is a legitimate time for day `d`.
            `d` has hour `hr` free.
        """
```

4.5.4 `scheduleSome` for SplittableTask

```python
    def scheduleSome(self, day):
        """
        Returns: True if a non-zero amount of the unscheduled time in this
          SplittableTask can be scheduled into `day`;
          and if so, this method does the scheduling of as much of the
          unscheduled time into `day` as possible, using the earliest empty
          time slots.
        Returns: True *also* in the case that this SplittableTask already has
          no more unscheduled time left.
        Returns False otherwise (i.e., there's still more of this SplittableTask
        to do but `day` is already full), with no alteration of `day`.

        Parameter `day`: the Day to try to schedule some of this SplittableTask
           into.
        Precondition: `day` is a Day object (not None)
        """
```

4.5.5 `__init__` for Month

Rewritten to avoid ambiguity in references to attributes versus parameters.

```python
    def __init__(self, month):
        """
        Initializer: Creates a Month with the month-number attribute `month_num`
        set to the value of `month` and the attribute `day_list` set as
        specified in the invariant for class Month.

        Precondition: `month` is an int in 1..12.
```

4.6 Thought Question To Answer

In the solutions to `a3_todo.py`, we implemented a function with header

```
def add_task(day, task, start_time):
```

According to its specification, this function is supposed to work for any Task given as second argument. But our `a3_todo.add_task` would fail if given a SplittableTask, because it wouldn't properly update the SplittableTask's `time_unscheduled` attribute.

Worse, if we lost access to the `a3_todo.py` file (or it was owned by someone else), we could never fix that `add_task` function to work correctly on SplittableTasks.

Could these problems have been avoided if we had made `add_task` be a method of class Task? The answer is yes! And this is an advantage of programming with classes and subclasses.

So, suppose `add_task` had been defined in `a3_todo.py` as a Task method, as so:

```
class Task():
    [...]

    def add_task(self, day, start_time):
```

In file in `a5_thought_question.py`, explain in 2-4 sentences how you could make `add_task` act correctly. for SplittableTasks as well as Tasks *without changing* `a3_todo.py`. (We mean, act correctly with respect to the problem we mentioned. To keep this assignment's length down, you do not need to discuss how to fix other issues that we didn't explicitly point out with `add_task` being a non-method function.)

5 Pre-Submission Checklist (all the same as A4) and What to Submit

Files to submit: `a5.py` and `a5_thought_question.py`.

Before submitting, ensure your code obeys the following.[6]

1. Lines are short enough (~80 characters) that horizontal scrolling is not necessary.
2. Functions are separated from each other by at least two blank lines.
3. You have removed any debugging `print` statements.
4. You have removed all `pass` statements.
5. You have removed "instruction" comments, such as "`# IMPLEMENT THIS FUNCTION`".
6. If you added any helper functions, these have good docstring specifications.

Make sure the following are all true before you submit.

7. You've changed the header comments in all files to list the entire set of people and sources that contributed to the code.
8. You (and your partner) have included your NetIDs in the header of all files.
9. The date in the header comments has been changed to when the files were last edited.
10. You have set your CMS notifications settings to receive email regarding grade changes, and regarding group invitations.
11. (reminder) If working with a partner, you have grouped on CMS. (One has invited on CMS, and one has accepted on CMS.)

[6] These requirements up speed up the process of reading/grading hundreds of files.