Alternate Project 3: Daily Planner

Created by Brian Weinfeld

Using variables, functions, and conditionals in Python, you will create a daily planner program.

Overview

In this project, you will be creating a daily planner program. A daily planner helps a user track what they need to do today and when they need to do it. This can be thought of as a TODO list that pairs each action with a time when the action is done. A planner can help make sure a person doesn't accidently book two things for the same time. Do you add events to your phone's calander?

Details

Behavior

```
Welcome to your Daily Planner
What would you like to do? (add, clear, display, exit) add
What event would you like to add? go to school
What time? 8
Is this event critical? no
Added go to school at 8
['', '', '', '']
['', '', '', '']
['go to school', '', '', '']
['', '', '', '']
['', '', '', '']
['', '', '', '']

What would you like to do? (add, clear, display, exit) add
What event would you like to add? wake up
What time? 6
Is this event critical? no
Added wake up at 6
['', '', '', '']
['', '', 'wake up', '']
['go to school', '', '', '']
['', '', '', '']
['', '', '', '']
['', '', '', '']

What would you like to do? (add, clear, display, exit) add
What event would you like to add? do homework
What time? 6
Is this event critical? no
Conflict with wake up

What would you like to do? (add, clear, display, exit) display
['', '', '', '']
```

```
['', '', 'wake up', '']
['go to school', '', '', '']
['', '', '', '']
['', '', '', '']
['', '', '', '']

What would you like to do? (add, clear, display, exit) add
What event would you like to add? eat breakfast
What time? 6
Is this event critical? yes
Replacing wake up at 6 with eat breakfast
['', '', '', '']
['', '', 'eat breakfast', '']
['go to school', '', '', '']
['', '', '', '']
['', '', '', '']
['', '', '', '']

What would you like to do? (add, clear, display, exit) clear
What time to start clearing? 5
What time to end clearing? 7
Clearing ['', 'eat breakfast']
['', '', '', '']
['', '', '', '']
['go to school', '', '', '']
['', '', '', '']
['', '', '', '']
['', '', '', '']

What would you like to do? (add, clear, display, exit) exit
Goodbye
```

Implementation Details

The planner is a list that has 24 empty strings in it. (The empty string is ''). Each of the 24 indexes reprents 1 hour of time in a day. For example, index 0 represents 12:00AM to 12:59AM, index 1 represents 01:00AM to 01:59AM and index 23 represents 11:00PM to 11:59PM. Your program will allow a user to add events at certain times of the day and prevent conflicts from occuring. All of the events will take exactly 1 hour.

Think back to the previous project where you created a TODO list program. All of the code in that program was put into a single script. This is OK when the program is small, but it gets harder and harder to code as the project gets bigger. Did you find the end of that project to be more challenging than the beginning? You can help mitigate this problem by creating functions to handle large amounts of work.

Begin this project by creating your 24 length list of empty strings and the following three functions. Do your best to ensure that these functions work properly before moving onto the rest of the project.

```
def add_event(planner, event, time, critical):
  # finish function
```

- add_event takes the planner (your list), the event you wish to add (a string), the time you want to do the event (an int between 0 to 23 inclusive) and whether the event is critial (a boolean). If there is no event in your planner for the indicated time, add it to the planner. If there is already an event for that time you have a conflict! (You can't do both things at the same time.) Use critical to figure out what to do. If critical is True, then the new event takes priority. Replace the event in the planner at that time with the new event. If critical is False, the event is not more important and instead an error should be printed. No change is made to the list. Return True if a new event is added to the planner and False if no changes were made.

```
def display_planner(planner):
  # finish function
```

- display_planner takes the planner (your list) and displays it in an easy to read fashion. Print 6 lines, each line containing 4 events. So the first line will contain events from the first four indexes, the second line will contain the next four and so on.

```
def clear_timeframe(planner, timeframe):
  # finish function
```

- clear_timeframe takes the planner (your list) and a timeframe (slice) that you wish to clear. Sometimes you need to remove events that have been cancelled. Clear out all of the events that occur during the given timeframe. Be careful, a slice can contain multiple elements. You need to set all the events in the slice back to the empty string.

- Once you have finished the above functions, create a script that allows a user to add and remove elements from their planner. The user should have four choices **add**, **clear**, **display**, **exit**. **add** is used to add events to the planner, **clear** is used to remove events from the planner, **display** is used to display the day's events and **exit** is used to exit the program. You should also automatically print the planner after any changes have been made to it. See the below running example for details.

Challenge

This section contains additional components you can add to the project. These should only be attemped after the project has been completed.

- Some events take longer than one hour. Modify the add_event function so that time is now a slice representing the beginning and ending time of the event. You will need to place the event in each index in your planner. Be careful, the new event can only be placed if either the event is critical (overwriting any event already in the planner at that time) or every hour in the planner is empty.

- Combine all the information into one line so that it is easier to interact with the planner. For example, you might add a new event by entering "add go to school 8 13 critical" or "add practice lines 14 15 not"

Super Challenge

The super challenge will require knowledge that has not been taught yet. You will need to do additional research on your own. Good luck!

Sometimes the time the event is scheduled for is less important than making sure the event is on the schedule. Add a parameter to the functions that add an element to the schedule called **force**. If force is true, the event must be scheduled. If it can't be scheduled at the expected time, find the closest available timeslot in the future and schedule it there instead. If the event is critical, then the displaced event should be moved forward. Be sure to alert the user that the event has been scheduled, but at a different time than expected! The placement will still fail if there are no available timeslots to place the event.