Electronic agenda

An electronic agenda keeps a register of entries that represent meetings and tasks. In this assignment you will develop two classes; an `Entry` class that represents an entry in an agenda, and a `Agenda` class that contains a list of entries.

Step 1

Define the `Entry` class. An entry takes place on a specific day. It has a description that can be of any type: a string, a picture, a movie, a number.

Introduce a constructor for initialising an entry with a given description. Use `description` as the name of the argument and specify a default value. Introduce methods `get_description` and `set_description` for manipulating the description of an entry.

Step 2

Add a method `get_day` to the `Entry` class. The day of an entry must be a positive integer number. The day of an entry cannot change, once the entry has been created. Change the constructor for initialising an entry with the given day. Use `day` as the name of the argument for that constructor, and introduce a default value.

Step 3

A meeting must consist of one or more consecutive time slots throughout the day. A day is divided in 24 time slots (ranging from 0 up to 23).

Introduce a constructor for initialising a meeting with a given description, a given day, a given start slot and a given end slot. Use `description` , `day` , `start` , and respectively `end` as the names of the arguments for that constructor, and introduce a default value for each argument.

Introduce methods `get_start_slot` and `get_end_slot` that respectively return the first and last slot occupied by a meeting. Both the start slot and the end slot cannot be changed, once the meeting has been created.

To check whether a slot is a valid number, add a method `is_valid_slot` that takes a number as a parameter and checks whether that number is valid (i.e. ranging from 0 up to 23).

Finally, in the constructor, check if start and end contain logic values.

Step 4

Define a method `occupies_slot` to check whether an entry occupies a given slot.

Define a method `overlaps_with` to check whether some entry overlaps with another entry. Two entries overlap if they are scheduled at the same day and have at least one slot in common.

Step 5

Define a second class named `Agenda` that contains a list of entries as an attribute. Initially, this list is empty.

Then, define a function `add_entry` that can be used to add a new entry to the list. Check whether the new entry overlaps with any other entries in the agenda. If that is the case, don't add the new entry to the list and return False. If all is well, return True.

Add a second function to `Agenda` called `get_entries_on_day` that returns the list of entries that take place on a specified day.

Step 6

Overload the operators `==` and `!=` in the `Entry` class. Make sure that entries are equal only if all their attributes are the same.

Step 7

Add a function `remove_entry` to `Agenda` that removes an entry from the agenda. Make use of the fact that the `==` operator has been overloaded in the previous step.

Step 8

Add a location attribute to the `Entry` class. Introduce methods `get_location` and `set_location` for manipulating the location of a meeting. Meetings are not required to have a location. Meetings that do not have a location should also not have an attribute for storing it. The method `get_location` must return None if the meeting at stake does not have a location. The method `set_location` must remove the attribute if None is supplied as the new location. As for the description, the location of a meeting can be of any type.

Tip: Use the built-in function `hasattr` to test whether an instance has a certain attribute. See https://docs.python.org/2/library/functions.html#hasattr

Step 9

Introduce a method that improves the textual representation of an `Entry` object. In other words, make sure that calling `str()` on an `Entry` object produces a string in the following format: `<description> on <day> from <start> until <end> at <location>`

Step 10

Overload the `__str__` function of `Agenda` so that it prints the list of entries in a chronological order.