For each method, implement the body.

```python
import event
class Day:
    """A calendar day and its events."""

    def __init__(self, calendar_day, calendar_month, calendar_year):
        """ (Day, int, str, int) -> NoneType

        Initialize a day on the calendar with calendar_day, calendar_month
        and calendar_year, and no events.

        >>> d = Day(1, 'August', 2015)
        >>> d.day
        1
        >>> d.month
        'August'
        >>> d.year
        2015
        >>> d.events
        []
        """

    def schedule_event(self, new_event):
        """ (Day, Event) -> NoneType

        Schedule new_event on this day, even if it overlaps with
        an existing event. Later we will improve this method.

        >>> d = Day(26, 'March', 2014)
        >>> e = event.Event(11, 12, 'Meeting')
        >>> d.schedule_event(e)
        >>> d.events[0] == e
        True
        """
```
def __str__(self):
    ""
    (Day) -> str
    
    Return a string representation of this day.
    ""

>>> d = Day(9, 'August', 2015)
>>> d.schedule_event(event.Event(13, 14, 'Submit last exercise'))
>>> d.schedule_event(event.Event(19, 23, 'Celebrate end of classes'))
>>> print(d)
9 August 2015:
- Submit last exercise: from 13 to 14
- Celebrate end of classes: from 19 to 23
""

if __name__ == '__main__':
    
    # Create day 1 August 2015.
    
    # Add an event "Sleep in" from 0 to 11 on 1 August 2015.
    
    # Add an event "Brunch" from 11 to 13 on 1 August 2015.
    
    # Print the day 1 August 2015, including its events.