Object Oriented Programming in Python

1. Implement a class complex in Python.
    1. Attributes: real, im both of type float
    2. Methods:
        - add(n): Given a Complex number n, returns a new rational number which is the sum of the owner Complex object and n.
        - subtract(n): Given a rational number n, returns a new rational number which is the difference of the owner Complex object and n.
        - multiply(n): Given a rational number n, returns a new rational number which is the product of the owner Complex object and n.
        - divide(n): Given a rational number n, returns a new rational number which is the questient of the owner Complex object and n.
        - neg(): Returns a new rational number which is the negative of the owner Complex object.
        - __str__ : Returns the string representation of the owner Complex object.
2. Implement a class Rational in Python.
    1. Attributes: numerator, denomenator both of type int.
    2. Methods:
        - add(n): Given a Rational number n, returns a new Rational number which is the sum of the owner Rational object and n.
        - subtract(n): Given a Rational number n, returns a new Rational number which is the difference of the owner Rational object and n.
        - multiply(n): Given a Rational number n, returns a new Rational number which is the product of the owner Rational object and n.
        - divide(n): Given a Rational number n, returns a new Rational number which is the questient of the owner Rational object and n.
        - neg(): Returns a new Rational number which is the negative of the owner Rational object.
        - __str__ : Returns the string representation of the owner Rational object.
3. In the calendar program we had implemented earlier, date was implemented as a tuple. Modify the program by implementing a Date class.
    - Attributes: day, month, year.
    - Methods:

1. nextDate(): Returns the next date
2. isLater(d): Returns True if the owner date object is later than the date d.
3. getWeekDay: Returns the weekday of the owner date object.
4. daysInBetween(d): Returns the days between owner date object and the date d.
5. dateRange(d): Given a date d, return the list of all dates from the owner date object to d2, the owner object and d2 included.
6. __str__(): Returns the string representation of the owner date object.

Functionally, the above program should be identical to the earlier calendar program. Therefore, the functions implemented in the original calendar program which have not been transformed into methods in the Date class above should be implemented as global functions as before, except that now they work with the modified version of the date object(earlier: tuple; current: Date class).

4. Write an object-oriented tree in line with tree2 of assignment 2.