Program: 2D Points

Using the template provided on the class web site, your task in this programming assignment is to implement a 2D point class in Python. You must provide/address the following in the class:

- A 2D point is made up of an *x*-component and a *y*-component. Each component is a floating point value;
- A constructor should initialize a point either with specified values for the *x*- and *y*-components or the point (0.0,0.0) as default;
- Instance variables should be appropriately named (i.e., beginning with underscores);
- Accessors and mutators should provide read access of and write access to the instance variables;
- A function named `dist` should take **two points as input** and calculate and **return the floating point distance** between the two points;
- A function named `midpt` should take **two points as input** and calculate and **return the midpoint** of the two points; and
- A *magic* function should provide a string representation of a point in the format `(x,y)`.

Note that you must **not** modify the main part of the program; therefore, your output should look **exactly** like the following:

```
jgourd@pi:~$ python 2Dpoints.py
p1: (0.0,0.0)
p2: (3.0,0.0)
p3: (3.0,4.0)
distance from p1 to p2: 3.0
distance from p2 to p3: 4.0
distance from p1 to p3: 5.0
midpt of p1 and p2: (1.5,0.0)
midpt of p2 and p3: (3.0,2.0)
midpt of p1 and p3: (1.5,2.0)
p4: (1.5,2.0)
distance from p4 to p1: 2.5
```

You should probably use the main part of the program and the provided output to help you design the class properly.

To help clarify, here are some specifics and/or constraints:

(1) Make sure that your 2D point class addresses all of the requirements listed above;
(2) Pay attention to the input(s) and output(s) of the functions `dist` and `midpt` (i.e., make sure that your code precisely matches their descriptions above);
(3) Use the decorator method discussed in class for accessors and mutators to properly *wrap* the instance variables;
(4) You must include a meaningful header, use good coding style, use meaningful variable names, and comment your source code where appropriate;
(5) Your output should be **exactly** like the sample run shown above; and
(6) You must submit your source code as a single .py file.