```
class Point:
    """
    A point in 2D coordinate system

    Public Attributes:
    ==================
    :type x: int
        the number of units to the right of origin
    :type y: int
        the number of units above origin
    """

    def __init__(self,x,y):
        """
        Construct a new 2D point self at coordinates x and y

        :param x: number of units to the right of the origin
        :type x: int
        :param y: number of units above the origin
        :type y: int
        """
        pass

    def __eq__(self,other):
        """
        Determine if point self is equivalent to point other

        :param other: a 2D point
        :type other: Point
        :return: whether coordinates of point self is the same as of the other
        :rtype: bool

        >>> p1 = Point(6,7)
        >>> p2 = Point(7,6)
        >>> p3 = Point(6,7)
        >>> p1 == p2
        False
        >>> p1 == p3
        True
        """
        pass

    def __str__(self):
        """
        Produce a user-friendly string representation of point self

        :return: string representation of point self
        :rtype: str

        >>> p = Point(3,4)
        >>> print(p)
        (3,4)
        """
        pass
```

Note 1: no need to define type contract for self in any method.

Note 2: no need to define return type in the constructor

Note 3: no need to define examples for the constructor

Note 4: if a class does not have any public attribute, write None in the corresponding area

Red arrows highlight some examples of the steps specified in Slides 11 (and 19) of Lecture02

```
    def distance_to_origin(self):
        """
        Calculate distance from this point to origin

        :return: square root of x^2 + y^2
        :rtype: float

        >>> p = Point(3, 4)
        >>> p.distance_to_origin()
        5.0
        """
        pass

    def __add__(self, other):
        """
        Sum point self and the other

        :param other: a 2D point
        :type other: Point
        :return: a new point whose coordinates are sum of coordinates of
        point self and the other, respectively
        :rtype: Point

        >>> p1 = Point(3,5)
        >>> p2 = Point(4,6)
        >>> print(p1.__add__(p2))
        (7,11)
        >>> print(p1+p2)
        (7,11)
        """
        pass

if __name__ == "__main__":
    import doctest
    doctest.testmod()
    p1 = Point(20,30)
    p2 = Point(12,13)
    p1 == p2
    p1 + p2
    p1.distance_to_origin()

    x = Point(3,4)
    print("x: ",x)
    print("distance to origin: ",x.distance_to_origin())
```