Встроенные атрибуты класса (перенести туда, где описываются методы класса)

https://www.tutorialspoint.com/python/python_classes_objects.htm

У каждого класса есть:

- **__doc__** - строка документации класса.
- **__dict__** - все атрибуты класса или объекта

Отрезок на плоскости XY

Создадим класс, который описывает отрезок на плоскости XY. Отрезок так же можно задать его началом и концом. Но начало и конец - это не числа на оси X, как было в предыдущем примере, а точки на плоскости XY.

Значит, нужно сначала создать класс, который описывает точки и работу с ними.

Точку полностью описывает ее x и y координата.

Точку можно создать, напечатать, подвинуть. Двигать будем на (dx, dy). Создадим дополнительный класс Direction, который хранит на сколько нужно сдвинуть одну точку, чтобы получить другую точку. (На самом деле это вектор).

Ничего кроме конструкора и печати в классе не нужно:

```
class Direction(object):
    def __init__(self, dx=0, dy=0):
        self.dx = dx    # на сколько надо сдвинуть по оси X
        self.dy = dy    # на сколько надо сдвинуть по оси Y

    def __str__(self):
        return '{} {}'.format(self.dx, self.dy)

    def __repr__(self):
        return 'Direction({}, {})'.format(self.dx, self.dy)
```

Опишем класс - точка на плоскости XY. Назовем его Point.

```
class Point(object):
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def __str__(self):
        return '{} {}'.format(self.x, self.y)

    def __repr__(self):
```

```
        return 'Point({}, {})'.format(self.x, self.y)

    def move(self, dir=None):
        """ Сдвигаем точку в направлении dir"""
        if dir is None:
            return
        self.x += dir.dx
        self.y += dir.dy

    def dir(self, other=None):
        """ На сколько нужно сдвинуть точку self, чтобы получить точку other"""
        if other is None:
            other = Point(0, 0)
        dx = self.x - other.x
        dy = self.y - other.y
        return Direction(dx, dy)    # возвращается объект типа Direction

    def dist2(self, other):
        """Квадрат расстояния до точки other"""
        dx = self.x - other.x
        dy = self.y - other.y
        return dx*dx + dy*dy

    def dist0(self):
        """Квадрат расстояния до точки (0,0)"""
        return dist2(Point(0,0))
```

Отрезок состоит из 2 точек start и finish. Значит, объект класса Segment2 состоит из точек (Point) star и finish:

```
class Segment2(object):
    def __init__(self, start=None, finish=None):
        self.start = start or Point()
        self.finish = finish or Point()

    def __str__(self):
        return '{} {}'.format(self.start, self.finish)

    def __repr__(self):
        return '({}, {})'.format(repr(self.start), repr(self.finish))

    def length2(self):
        return self.start.dist2(self.finish)

    def length(self):
        return math.sqrt(self.length2())

    def move(self, dir=None):
        self.start.move(dir)
        self.finish.move(dir)
```

Напишем отдельно фукнцию `read_segment`, которая по строке делает отрезок и возвращает объект типа Segment2

```
def read_segment2(str):
    x1, y1, x2, y2 = map(int, str.split())
    return Segment2(Point(x1, y1), Point(x2, y2))
```

Задача прочитать отрезки и передвинуть их так, чтобы точки start лежали в точке (0,0). Напечатать полученные отрезки.

```
def move_to_zero(a):
    zero = Point(0, 0)
    for s in a:
        dir = zero.dir(s.start) # вычисляем Direction от точки (0,0) до точки
start отрезка s
        print(s)
        s.move(dir)             # двигаем отрезок s на это dir
        print(s)

import sys

a = []
for str in sys.stdin:
    s = read_segment2(str)
    a.append(s)

move_to_zero(a)
```

Задачи

Rect

Написать класс Rect для хранения прямоугольников на плоскости XY (стороны параллельны осям). Прямоугольник задавать левой верхней и правой нижней точками.

Реализовать:

- конструктор **init**
- печать **str**
- вычисление площади area
- normalize - делает левую верхнюю точку левее и выше правой нижней.
- move(dir)

Протестировать класс.

Площади

Даны прямоугольники (4 целых числа через пробел на строку).

Напечатать прямоугольники и их площади.

```
Input:
1 1 3 -2
0 0 5 -4
-2 8 0 2
```

```
Output:
1 1 3 -2 6
0 0 5 -4 20
-2 8 0 2 12
```

Все lt в (0,0)

Даны прямоугольники (4 целых числа через пробел на строку). Передвинуть их так, чтобы **левая верхняя** точка каждого прямоугольника лежала в (0, 0).

```
Input:
1 1 3 -2
0 0 5 -4
-2 8 0 2
Output:
0 0 2 -3
0 0 5 -4
0 0 2 -6
```

Все левые нижние точки в (0,0)

Даны прямоугольники (4 целых числа через пробел на строку). Передвинуть их так, чтобы **левая нижняя** точка каждого прямоугольника лежала в (0, 0).

```
Input:
1 1 3 -2
0 0 5 -4
-2 8 0 2
Output:
0 3 2 0
0 4 5 0
0 6 2 0
```

Центры в (0,0)

Даны прямоугольники (4 целых числа через пробел на строку). Передвинуть их так, чтобы **центр**(пересечение диагоналей) каждого прямоугольника лежал в (0, 0).

Напишите для этого в классе Point функцию centr(self, other), которая возвращает точку на середине отрезка [self, other].

```
Input:
1 1 3 -2
0 0 5 -4
-2 8 0 2
Output:
-1 1.5 1 -1.5
-2.5 2 2.5 -2
-1 3 1 -3
```

Левые верхние точки в левую верхнюю точку первого прямоугольника

Даны прямоугольники (4 целых числа через пробел на строку). Передвинуть их так, чтобы **левая верхняя** точка каждого прямоугольника лежала в левой верхней точке первого прямоугольника.

```
Input:
1 1 3 -2
0 0 5 -4
-2 8 0 2
Output:
1 1 3 -2
1 1 6 -3
1 1 3 -5
```