

Simple Point class

```
1: class Point:
2:     def __init__(self):
3:         self._x = 0
4:         self._y = 0
5:
6:     def getX(self):
7:         return self._x
8:
9:     def setX(self, val):
10:        self._x = val
11:
12:    def getY(self):
13:        return self._y
14:
15:    def setY(self, val):
16:        self._y = val
17:
18:
19:
20:
21:
22: if __name__ == '__main__':
23:     a = Point()
24:     a.setX(5)
25:     a.setY(7)
26:
27:     b = Point()
28:     b.setX(a.getX()-8)
29:     b.setY(4)
```

Robust Point class

```
1: from math import sqrt                                # needed for computing distances
2:
3: class Point:
4:     def __init__(self, initialX=0, initialY=0):
5:         self._x = initialX
6:         self._y = initialY
7:
8:     def getX(self):
9:         return self._x
10:
11:     def setX(self, val):
12:         self._x = val
13:
14:     def getY(self):
15:         return self._y
16:
17:     def setY(self, val):
18:         self._y = val
19:
20:     def scale(self, factor):
21:         self._x *= factor
22:         self._y *= factor
23:
24:     def distance(self, other):
25:         dx = self._x - other._x
26:         dy = self._y - other._y
27:         return sqrt(dx * dx + dy * dy)                # imported from math module
28:
29:     def normalize(self):
30:         mag = self.distance( Point() )
31:         if mag > 0:
32:             self.scale(1/mag)
33:
34:     def __str__(self):
35:         return '<' + str(self._x) + ',' + str(self._y) + '>'
36:
37:     def __add__(self, other):
38:         return Point(self._x + other._x, self._y + other._y)
39:
40:     def __mul__(self, operand):
41:         if isinstance(operand, (int,float)):          # multiply by constant
42:             return Point(self._x * operand, self._y * operand)
43:         elif isinstance(operand, Point):             # dot product
44:             return self._x * operand._x + self._y * operand._y
45:
46:     def __rmul__(self, operand):
47:         return self * operand
```