Simple Point class

```
class Point:
  def __init__(self):
    self._x = 0
    self._y = 0

  def getX(self):
    return self._x

  def setX(self, val):
    self._x = val

  def getY(self):
    return self._y

  def setY(self, val):
    self._y = val




if __name__ == '__main__':
  a = Point()
  a.setX(5)
  a.setY(7)

  b = Point()
  b.setX(a.getX()-8)
  b.setY(4)
```

Robust Point class

```
from math import sqrt                       # needed for computing distances

class Point:
  def __init__(self, initialX=0, initialY=0):
    self._x = initialX
    self._y = initialY

  def getX(self):
    return self._x

  def setX(self, val):
    self._x = val

  def getY(self):
    return self._y

  def setY(self, val):
    self._y = val

  def scale(self, factor):
    self._x *= factor
    self._y *= factor

  def distance(self, other):
    dx = self._x - other._x
    dy = self._y - other._y
    return sqrt(dx * dx + dy * dy)          # imported from math module

  def normalize(self):
    mag = self.distance( Point() )
    if mag > 0:
      self.scale(1/mag)

  def __str__(self):
    return '<' + str(self._x) + ',' + str(self._y) + '>'

  def __add__(self, other):
    return Point(self._x + other._x, self._y + other._y)

  def __mul__(self, operand):
    if isinstance(operand, (int,float)):                # multiply by constant
      return Point(self._x * operand, self._y * operand)
    elif isinstance(operand, Point):                    # dot product
      return self._x * operand._x + self._y * operand._y

  def __rmul__(self, operand):
    return self * operand
```