distance:

x, y coordinates

r, θ coordinates

(x, y) = (a, b)

(x, y) = (y, x)

```
def distance (self, p):
    return (sqrt ((p.x - self.x)**2 +
                  (p.y - self.y)**2))
```

getpoint() → (x, y)

```
def distance(self, p):
    (myx, myy) = self.getpoint()
    (px, py) = p.getpoint()
    proceed as above
```

translate (deltax, deltay)

x, y point

```
def translate (self, dx, dy):
    self.x = self.x + dx
    self.y = self.y + dy
```

r, θ point

```
def translate (self, dx, dy):
    (myx, myy) = self.getpoint()
    self.setpoint(myx+dx, myy+dy)
```

Optional/default arguments

```
def int(string, base):
    ≡
```

requires int to be called with 2 args always

Make base optional by providing a default value

```
def int(string, base=10):
    ξ
```

Modify Point so that coords set to (0,0) if not specified while "constructing"

```
class Point:
    def __init__(self,a,b):
        self.x=a
        self.y=b
```

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

Note: Fix r,θ implementation when a=0

"Persistent" names/values

While creating points, remember all points ever added

Compute p.nearest() among currently defined points

Define attributes outside all functions but inside Class

```
class Point:
    allpoints = []

    def __init__(self, a=0, b=0):
        self.x = a
        self.y = b
        Point.allpoints.append(self)

    Define nearest neighbour of self
```

Inside a point

Scan all points in Point.allpoints

Compute distance to each

Return the nearest one