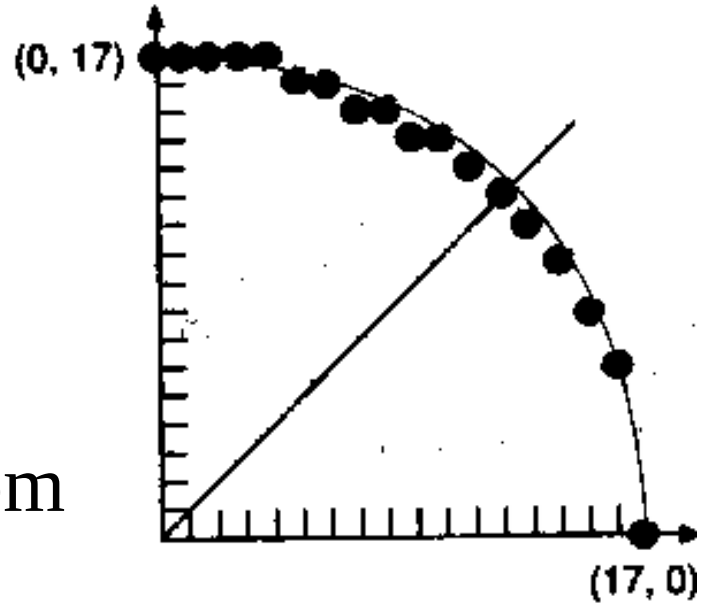
Midpoint Circle Drawing Algorithm



- Explicit: $y = f(x)$

$$y = \pm\sqrt{R^2 - x^2} \quad \ldots (1)$$

Usually, we draw a quarter circle by incrementing x from 0 to R in unit steps and solving for +y for each step.

- Parametric:

$x = R\cos\theta$ - a quarter circle by stepping the angle from 0 to 90

$y = R\sin\theta$ - avoids large gaps but still insufficient.

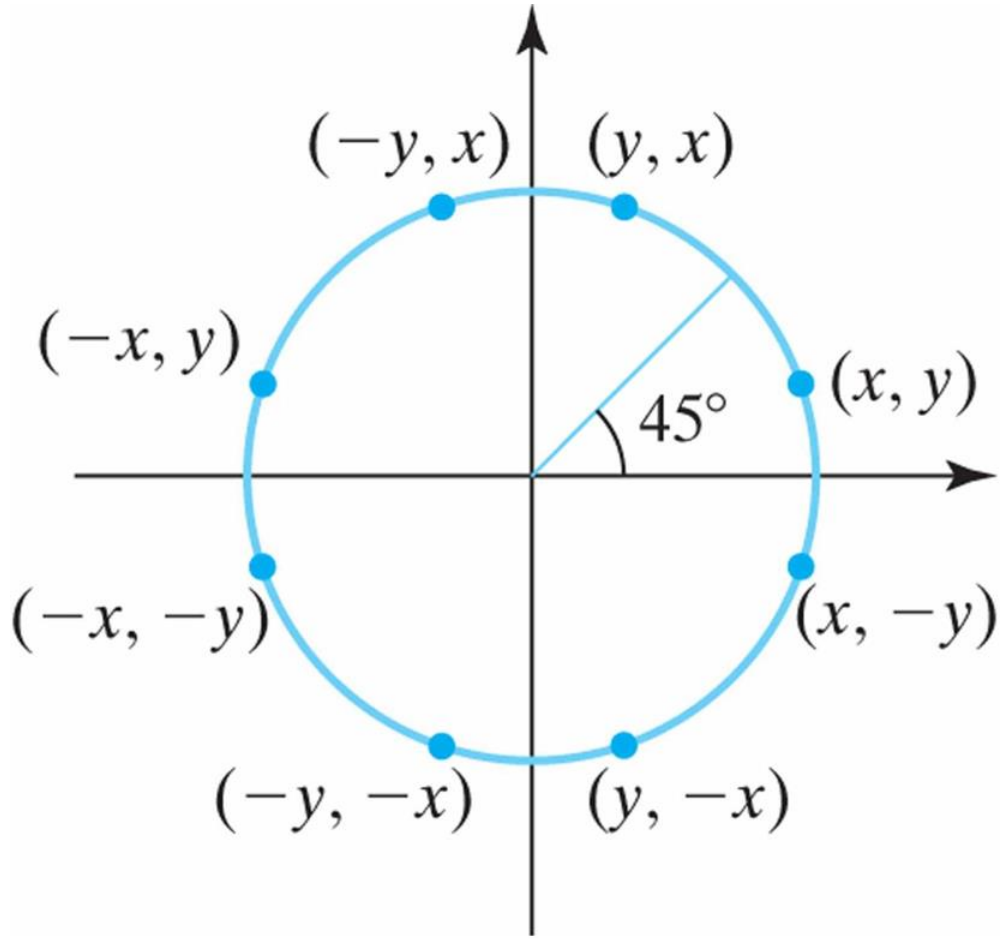
- Implicit: $f(x, y) = x^2 + y^2 - R^2 \ldots (2)$

If $f(x, y) = 0$ then it is on the circle.

$f(x, y) > 0$ then it is outside the circle.

$f(x, y) < 0$ then it is inside the circle.

Eight-way Symmetry



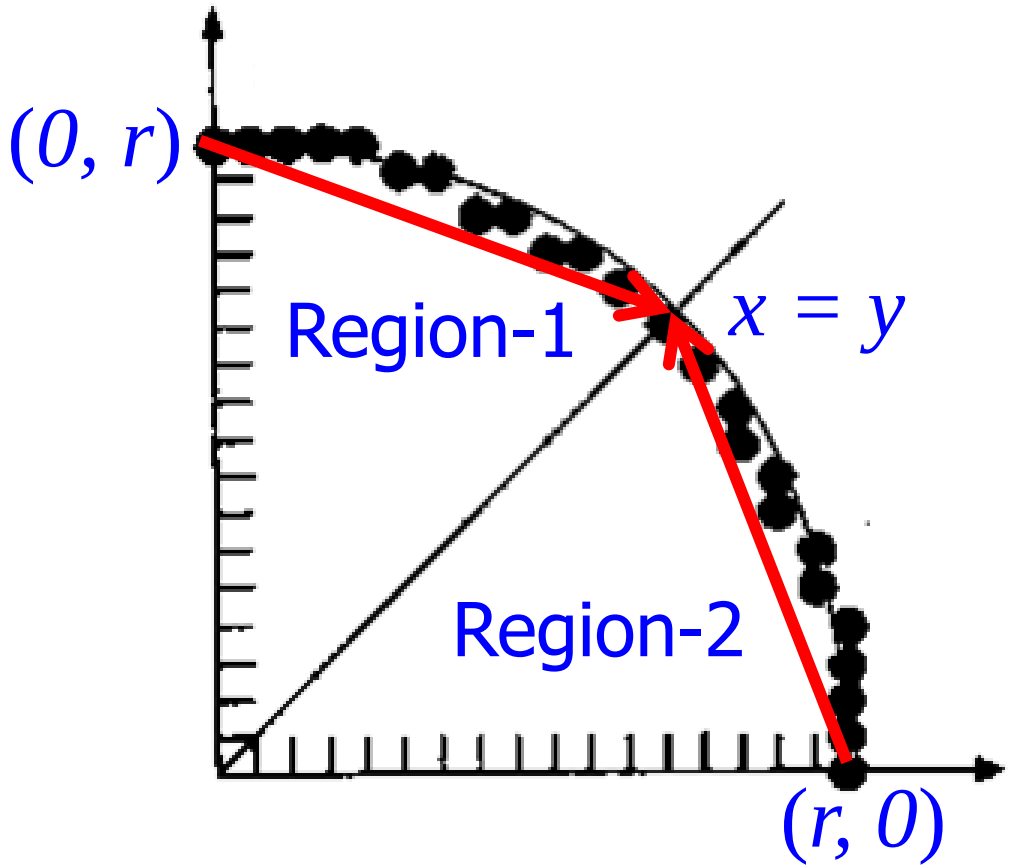

```
def draw8way(x, y):
    drawPixel(x, y)
    drawPixel(-x, y)
    drawPixel(x, -y)
    drawPixel(-x, -y)
    drawPixel(y, x)
    drawPixel(-y, x)
    drawPixel(y, -x)
    drawPixel(-y, -x)
```

Using this symmetry we can complete the circle by drawing an octant only

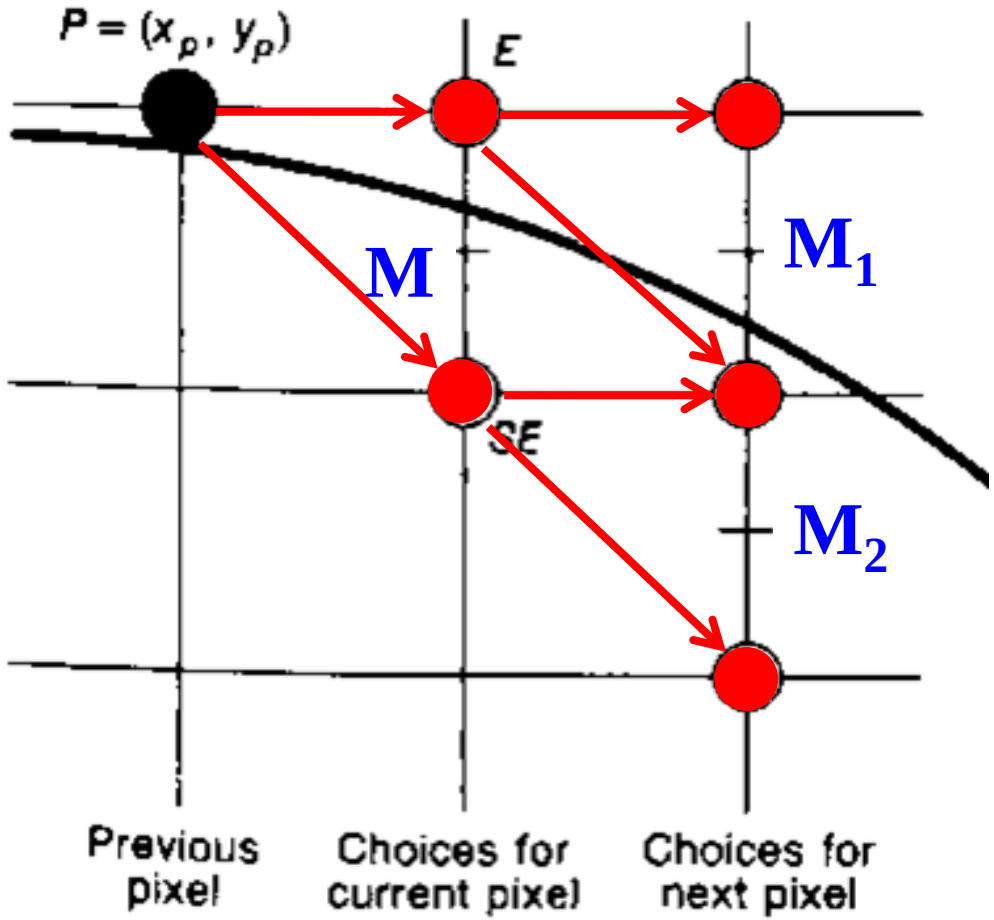
Midpoint Circle Drawing Algorithm (cont.)

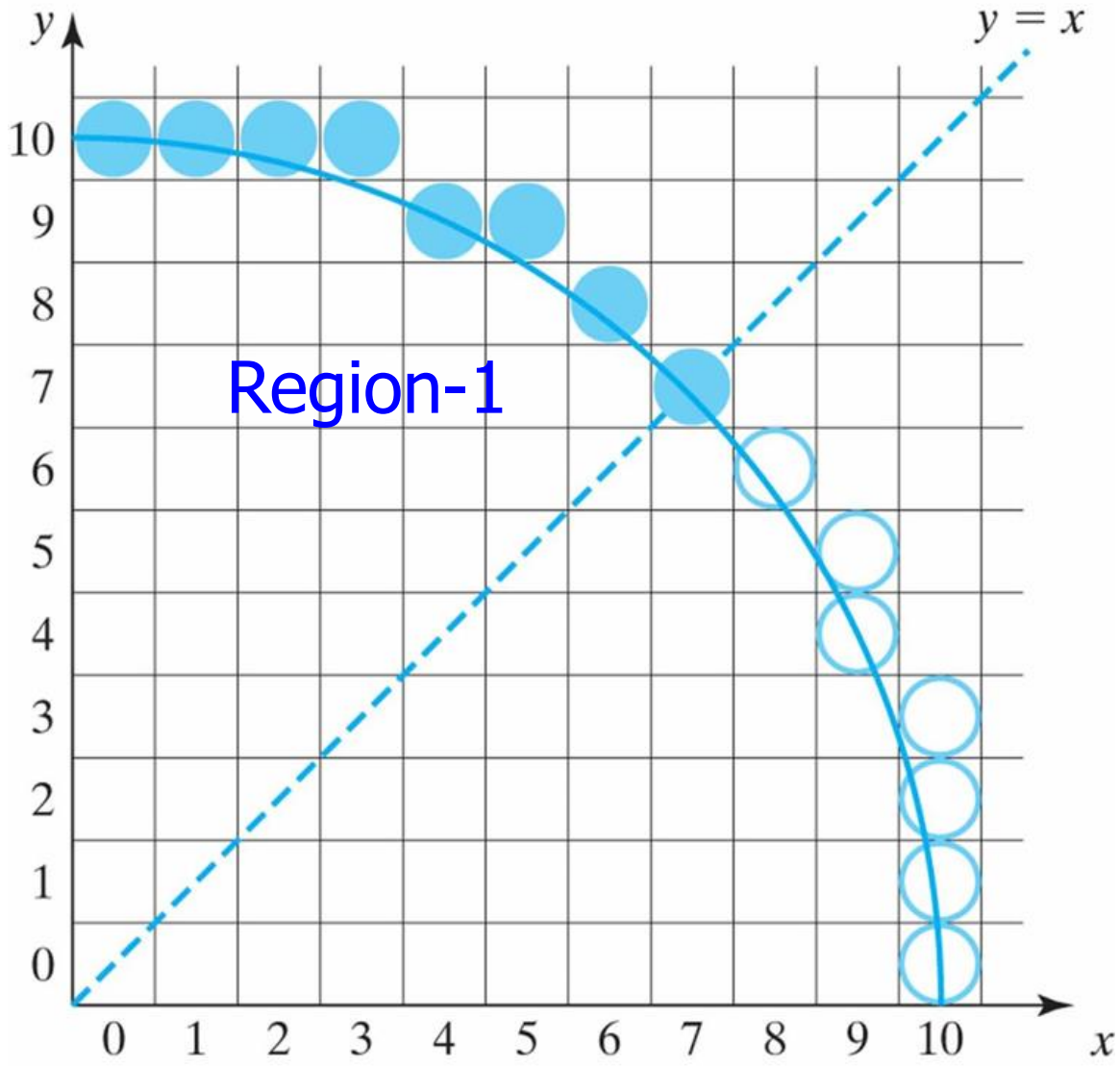
Which octant is best?

- Either the octant from *(0, r)* to *(x = y)* Region-1, which is like Zone-7, i.e., in this region the movements are ΔE and ΔSE, loop controller x.
- Or, from *(r, 0)* to *(x = y)* Region-2, Zone-2, in this region the movements are ΔN and ΔNW, loop controller y.
- We will learn both the regions.



Circle Drawing in Region-1





Derivations

The equation of circle: $f(x, y) = x^2+y^2 - r^2$ ... (2)

- Deviation at M, $d = f(M) = f(x_p+1, y_p-½) = (x+1)^2+(y-½)^2-r^2$
- Deviation at M_1, $d_1 = f(M_1) = f(x_p+2, y_p-½) = (x+2)^2+(y-½)^2-r^2$
- Deviation at M_2, $d_2 = f(M_2) = f(x_p+2, y_p-{}^3/_2) = (x+2)^2+(y-{}^3/_2)^2-r^2$
  - $\Delta E = d_1 - d = 2x_p+3$ ... (3)
  - $\Delta SE = d_2 - d = = 2y_p - 2x_p+5$ ... (4)
- Initial/first value of d
  - d_{init} = d at 1st mid-point $(0, r)$ = $f(0+1, r-½) = 1^2+(r-½)^2-r^2$

$$= \frac{5}{4} - R \quad (5)$$

Code/Algorithm for Midpoint Circle drawing

We know that,

If $f(x, y) = 0$ then the point is on the circle.

$f(x, y) > 0$ then it is outside the circle.

$f(x, y) < 0$ then it is inside the circle.

- In region-1, ΔE means a point going from inside to outside. So, if $f(x\ y)$ or d < 0, (point is inside the circle), so we choose ΔE.
- Else ΔSE

```
def drawCircle_1(r):
    x = 0
    y = r
    d = 5 – 4*r
    draw8way(x, y)
    while y > x:
        if d < 0:  # delE
            d += 4 * (2*x + 3)
            x += 1
        else:  # delSE
            d += 4 * (2*y – 2*x + 5)
            x += 1
            y -= 1
        draw8way(x, y)
```

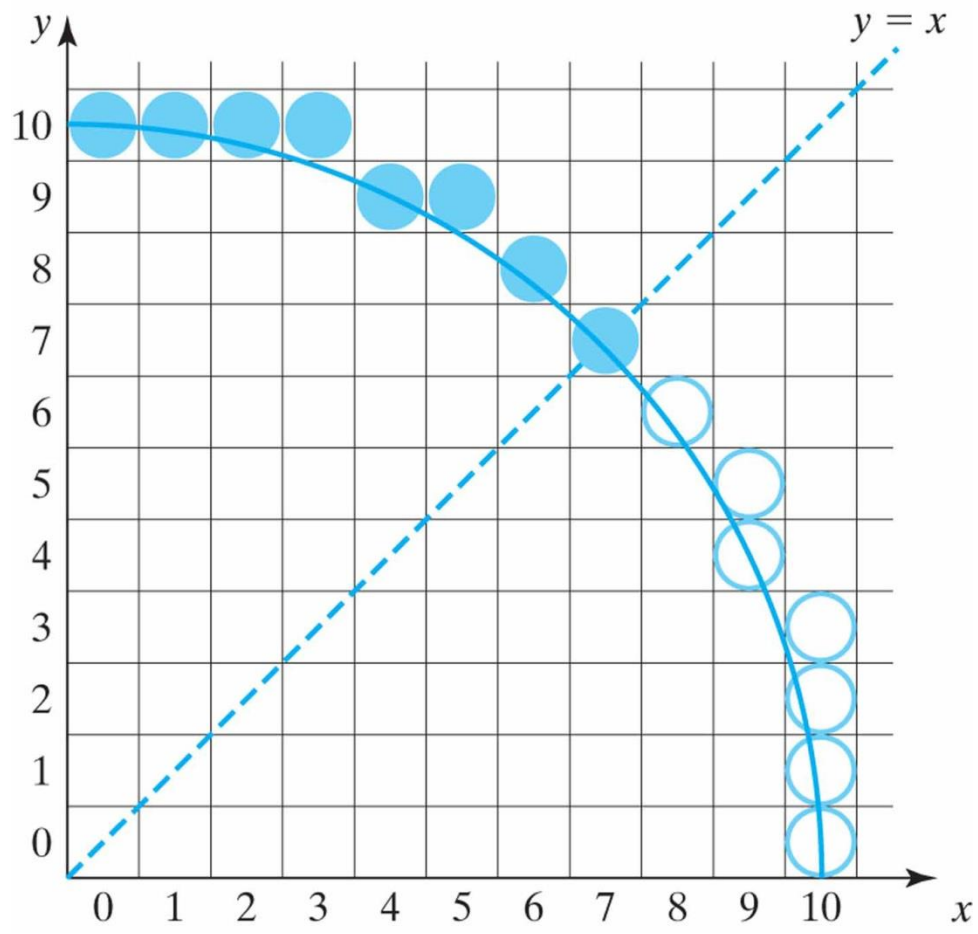
Example: Midpoint Circle Drawing

Given a circle of radius *r=10*, using midpoint circle drawing algorithm, determine the circle points in *region-1*, i.e., *x=0* to *x=y*, assume that the center is at (*0, 0*)

$d_{init} = 4(5/4 - r) = 5 - 4r = -35$, *Starting value of x = 0 and y = 10*

Pixel	*x*	*y*	*d*	*ΔE/ΔSE*
1	*0*	*10*	*-35*	*ΔE*
2	*1*	*10*	*-23*	*ΔE*
3	*2*	*10*	*-3*	*ΔE*
4	*3*	*10*	*25*	*ΔSE*
5	*4*	*9*	*-11*	*ΔE*
6	*5*	*9*	*33*	*ΔSE*
7	*6*	*8*	*29*	*ΔSE*
8	*7*	*7*	*33*	*ΔSE*

Example: Midpoint Circle Drawing (cont.)



Pixel positions (solid circles) along a circle path centered on the origin and with radius r = 10, as calculated by the midpoint circle algorithm. Open ("hollow") circles show the symmetry positions in the first quadrant.