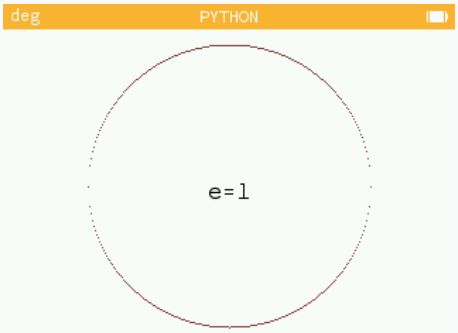
Dessin d'un cercle

Écrire un programme qui trace un cercle de centre $O(x, y)$ et de rayon R.

On va créer une fonction **cercle** permettant optionnellement de remplir ou non l'intérieur avec une couleur. D'un point de vue mathématique, un cercle de centre $O(x_O, y_O)$ et de rayon R est l'ensemble des points $M(x, y)$ tels que $(x - x_O)^2 + (y - y_O)^2 = R^2$, ce qui conduit à $y = y_O \pm \sqrt{R^2 - (x - x_O)^2}$.

```
from kandinsky import *
from math import *
def cercle(x0,y0,r,c1,e,c2):
  for i in range(e):
    x1=x0-r+i
    x2=x0+r-i
    for x in range(x1,x2+1):
      y1=int(y0+sqrt((r-i)**2-(x-x0)**2))
      y2=int(y0-sqrt((r-i)**2-(x-x0)**2))
      set_pixel(x,y1,c1)
      set_pixel(x,y2,c1)
  draw_string('e='+str(e),x0-15,y0-5)

cint=color(210,255,212) #ivoire
cbor=color(139,68,66) #chatain
cercle(160,111,100,cbor,5,cint)
```



Transformée en programme, cette formule trace un cercle avec des trous sur les bords, car il y a trop peu de pixels utilisés lorsque x s'approche de sa valeur minimale $x_O - R$ ou de sa valeur maximale $x_O + R$.

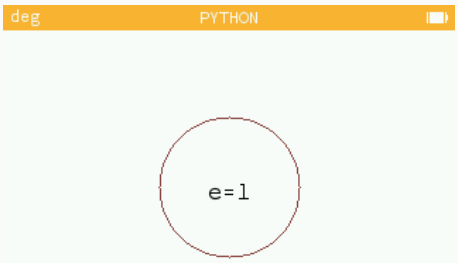
Ce n'est pas satisfaisant, nous voulons un cercle sans trou. Une idée simple à mettre en œuvre est de tracer de cette façon les deux quarts de cercle supérieurs et inférieurs (en termes géographiques NO-NE et SE-SO et, en termes trigonométrique, de $\frac{\pi}{4}$ rad à $\frac{3\pi}{4}$ rad et de $-\frac{3\pi}{4}$ rad à $-\frac{\pi}{4}$ rad) et, pour les deux autres quarts de cercle, de faire subir aux deux premiers un quart de tour direct.

Pour cela, il suffit de se rappeler des formules de trigonométrie : $cos(\alpha + \frac{\pi}{2}) = -sin(\alpha)$ et $sin(\alpha + \frac{\pi}{2}) = cos(\alpha)$.

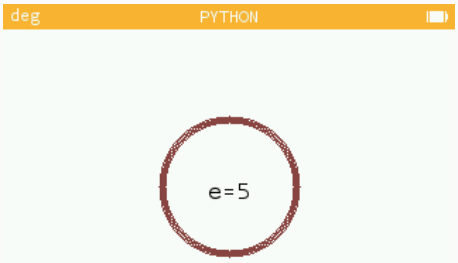
```
from kandinsky import *
from math import *
def cercle(x0,y0,r,c1,e,c2):
  for i in range(e):
    xd=x0-int((r-i)/sqrt(2))
    xf=x0+int((r-i)/sqrt(2))
    for x in range(xd,xf+1):
      x1=x
      y1=y0+int(sqrt((r-i)**2-(x-x0)**2))
      set_pixel(x,y1,c1)
      for j in range(3):
        x2=x0+y1-y0
        y2=y0+x0-x1
        set_pixel(x2,y2,c1)
        x1,y1=x2,y2
  draw_string('e='+str(e),x0-15,y0-5)

cint=color(210,255,212) #ivoire
cbor=color(139,68,66) #chatain
cercle(160,111,50,cbor,1,cint)
```

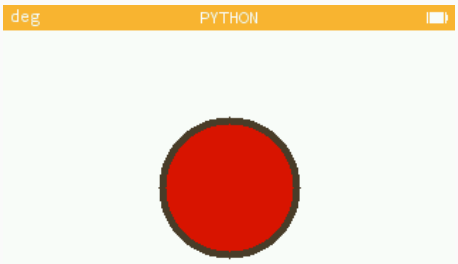
Cela donne de bien meilleurs cercles, sans trou, même s'il subsiste quelques pixels blancs dans le bord.



Ce défaut ne se remarque que lorsque le bord prend de l'épaisseur, je suppose que les arrondis effectués par la fonction `int` conduisent les pixels de deux cercles voisins à se superposer alors qu'ils ne devraient pas. La solution n'est toujours pas satisfaisante car je souhaite un cercle sans trou dans son bord. L'amélioration ultime vient de cet aménagement : je double le nombre e de cercles qui crée l'épaisseur et je divise par deux la diminution du rayon à chaque étape. De cette façon, le problème des pixels qui se superposent, laissant des trous, ne se retrouve plus.



Je profite de cette forme d'achèvement pour implémenter le remplissage de l'intérieur : il suffit de créer la fonction `cercle_plein` qui envoie deux exécutions de la fonction `cercle`, la première pour tracer le bord et la seconde pour remplir l'espace restant qui est considéré comme le bord d'un cercle dont l'épaisseur est égale au rayon.



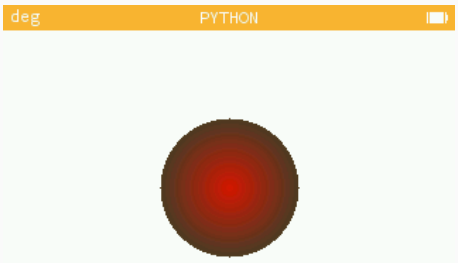

```
from kandinsky import *
from math import *
def cercle(x0,y0,r,c,e):
  for i in range(2*e):
    xd=x0-int((r-i*0.5)/sqrt(2))
    xf=x0+int((r-i*0.5)/sqrt(2))
    for x in range(xd,xf+1):
      x1=x
```

```
      y1=y0+int(sqrt((r-i*0.5)**2-(x-x0)**2))
      set_pixel(x,y1,c)
      for j in range(3):
        x2=x0+y1-y0
        y2=y0+x0-x1
        set_pixel(x2,y2,c)
        x1,y1=x2,y2

def cercle_plein(x0,y0,r,c1,e,c2):
  cercle(x0,y0,r,c1,e)
  cercle(x0,y0,r-e,c2,r-e)

cint=color(219,23,2) #cinabre
cbor=color(78,61,40) #bitume
cercle_plein(160,111,50,cbor,5,cint)
```

J'ai envie de terminer sur la construction d'un dégradé entre la couleur du bord et celle de l'intérieur. Utilisant le principe du dégradé entre deux couleurs mis au point dans la fiche Le module kandinsky[a], la fonction `cercle_grade` réalise cela.




```
def cercle_grade(x0,y0,R,c1,e,c2):
  for i in range(R):
    r=c1[0]+i*(c2[0]-c1[0])//R
    g=c1[1]+i*(c2[1]-c1[1])//R
    b=c1[2]+i*(c2[2]-c1[2])//R
    cercle(x0,y0,i,color(r,g,b),1)

cExt=[219,23,2]
cInt= [78,61,40]
cercle_grade(160,111,50,cExt,10,cInt)
```

a. `content/fr/ressources/python/activites/kandinsky/index.html`