```
__author__ = 'Charlotte Nachtegael'
__date__ = '21 octobre 2015'

# -*-coding:Utf-8 -*

"""Programme qui dessine un pavage d'hexagones, divisés
 en 3 losanges de couleurs, auquel on applique une déformation"""

import turtle

inf_gauche = -300
sup_droit = 300
longueur = 20
col1 = 'red'
col2 = 'blue'
col3 = 'black'
centre = (-100, -100, -100)
r = 200
EPS = 1.0e-5

turtle.title("Vasarely doit se retourner dans sa tombe")

def deformation(p,c,r):
    """reçoit le point p =(x_p,y_p,z_p), le centre c=(x_c,y_c,z_c),
    et le rayon r tous les trois avant déformation,
    renvoit un point p2 après déformation
    """
    def distance(p,q):
        """distance entre les 2 points p et q"""
        return ((q[0]-p[0])**2 + (q[1]-p[1])**2 + (q[2]-p[2])**2)**0.5
    d = distance(p,c)
    if d >= r:
        res = p
    else:
        if d > EPS:
            facteur = r/d
        else:
            facteur = 1
        x2 = c[0] + (p[0]-c[0]) * facteur
        y2 = c[1] + (p[1]-c[1]) * facteur
        z2 = c[2] + (p[2]-c[2]) * facteur
        res = (x2,y2,z2)
    return res
deform = lambda p: deformation(p,centre,r) # définition de la fonction deform
envoyée à pavage

def list_pos(c, longueur, vas):
    """Reçoit le centre c avant déformation et donne la liste des coordonnées
    des points de l'hexagone de rayon longueur sans déformation"""
    res = []
    vas.penup()
    vas.hideturtle()
    vas.goto(c)
    vas.forward(longueur)
    vas.left(120)
    for i in range(6):
        vas.forward(longueur)
        vas.left(60)
        res.append(vas.pos())
    return res
```

```
def hexagone(vas, c, longueur,col1, col2, col3, deform):
    """Reçoit la coordonnée du centre avant déformation et dessine un hexagone
    de rayon longueur autour de celui-ci, dont les points et le centre ont été
déformés
    selon deform. L'hexagone est composé de trois losanges de couleurs (col 1,
col2 et col3)."""
    z = 0
    list_positions = list_pos(c, longueur, vas)
    c_new = deformation((c[0], c[1], z), centre, r)
    p1 = deformation((list_positions[0][0], list_positions[0][1], z), centre, r)
    p2 = deformation((list_positions[1][0], list_positions[1][1], z), centre, r)
    p3 = deformation((list_positions[2][0], list_positions[2][1], z), centre, r)
    p4 = deformation((list_positions[3][0], list_positions[3][1], z), centre, r)
    p5 = deformation((list_positions[4][0], list_positions[4][1], z), centre, r)
    p6 = deformation((list_positions[5][0], list_positions[5][1], z), centre, r)
    vas.showturtle()
    vas.penup()
    vas.goto(c_new[0], c_new[1])
    vas.pendown()
    vas.color(col1, col1)
    vas.begin_fill()
    vas.goto(p6[0], p6[1])
    vas.goto(p1[0], p1[1])
    vas.goto(p2[0], p2[1])
    vas.goto(c_new[0], c_new[1])
    vas.end_fill()
    vas.color(col2, col2)
    vas.begin_fill()
    vas.goto(p4[0], p4[1])
    vas.goto(p5[0], p5[1])
    vas.goto(p6[0], p6[1])
    vas.goto(c_new[0], c_new[1])
    vas.end_fill()
    vas.color(col3, col3)
    vas.begin_fill()
    vas.goto(p2[0], p2[1])
    vas.goto(p3[0], p3[1])
    vas.goto(p4[0], p4[1])
    vas.goto(c_new[0], c_new[1])
    vas.end_fill()
    vas.setheading(0) # met la tortue Vasarely en place pour le prochain hexagone


def pavage(vas, inf_gauche, sup_droit, longueur, col1, col2, col3, deform):
    """Trace un pavage composé d'hexagones dont le coin inférieur est donne par
    inf_gauche et le coin supérieur droit par sup_droit."""
    vas.speed(800)
    centre_inf_gauche = (inf_gauche, inf_gauche)
    centre_sup_droit = (sup_droit, sup_droit)
    i = 0
    x_c = centre_inf_gauche[0]
    y_c = centre_inf_gauche[1]
    while x_c <= centre_sup_droit[0] and y_c <= centre_sup_droit[1]:
      # dessine jusqu'à atteindre le coin sup droit
        hexagone(vas, (x_c, y_c), longueur, col1, col2, col3, deform)
        x_c += 3 * longueur
        if x_c > centre_sup_droit[0] and i == 0:
            x_c = centre_inf_gauche[0] + 1.5 * longueur
            y_c += (3**0.5/2) * longueur
            i = 1
```

```
        elif x_c > centre_sup_droit[0] and i == 1:
            x_c = centre_inf_gauche[0]
            y_c += (3**0.5/2) * longueur
            i = 0
    vas.exitonclick()

pavage(turtle, inf_gauche, sup_droit, longueur, col1, col2, col3, deform)
```