```
#----------LIBRERIAS----------#

import turtle           #Importamos la libreria Turtle
from tkinter import Tk, Label, Button           #De la libreria tkinter importamos Tk, Label y Button



#----------VARIABLE----------#

pinta = 0



#----------CLASES----------#

class Fractal():            #Clase Fractal con los atrubutos necesarios para los fractales y funciones para complear los atributos
    def __init__(self):
        self.longitud = 0
        self.factor = 0
        self.angulo = 0
        self.casoBase = 0
        self.a = turtle.Turtle()

    def DameLongitud(self, longitud):
        self.longitud = longitud

    def DameFactor(self, factor):
        self.factor = factor

    def DameAngulo(self, angulo):
        self.angulo = angulo

    def DameCasoBase(self, casoBase):
        self.casoBase = casoBase

class Tree(Fractal):            #Clase Tree que hereda de Fractal cun funciones de inicializar todos los parametros del padre y los actualiza a los valores deseados y la funcion recursiva del fractal
    def __init__(self, longitud, factor, angulo, casoBase):
        super().__init__(longitud, factor, angulo, casoBase)
        super().DameLongitud(longitud)
        super().DameFactor(factor)
        super().DameAngulo(angulo)
        super().DameCasoBase(casoBase)

    def Arbol(self, longitud, factor, angulo, casoBase):
```

```
        if longitud >= casoBase:
            self.a.forward(longitud)
            longitudNueva = longitud - factor
            self.a.left(angulo)
            self.Arbol(longitudNueva, factor, angulo, casoBase)
            self.a.right(angulo * 2)
            self.Arbol(longitudNueva, factor, angulo, casoBase)
            self.a.left(self.angulo)
            self.a.backward(longitud)

class Koch(Fractal):            #Clase Koch que hereda de Fractal cun funciones de inicializar todos los parametros del padre y los actualiza a los valores deseados y la funcion recursiva del fractal
    def __init__(self, longitud, factor, angulo, casoBase):
        super().__init__(longitud, factor, angulo, casoBase)
        super().DameLongitud(longitud)
        super().DameFactor(factor)
        super().DameAngulo(angulo)
        super().DameCasoBase(casoBase)

    def CurvaKoch(self, longitud, factor, angulo, casoBase):
        if casoBase == 0:
            self.a.forward(longitud)
        else:
            casoBase -= 1
            longitud /= factor
            self.CurvaKoch(longitud, factor, angulo, casoBase)
            self.a.left(angulo)
            self.CurvaKoch(longitud, factor, angulo, casoBase)
            self.a.right(angulo * 2)
            self.CurvaKoch(longitud, factor, angulo, casoBase)
            self.a.left(angulo)
            self.CurvaKoch(longitud, factor, angulo, casoBase)

class Carpet(Fractal):
    def __init__(self, longitud, factor, angulo, casoBase):
        super().__init__()
        super().DameLongitud(longitud)
        super().DameFactor(factor)
        super().DameAngulo(angulo)
        super().DameCasoBase(casoBase)

    def DrawCube(self, longitud, angulo):
        self.a.begin_fill()
        for i in range(4):
            self.a.forward(longitud)
            self.a.right(angulo)
        self.a.end_fill()
```

```
    longitud /= 3
    self.a.penup()

def AlfombraSierpinski(self, longitud, factor, angulo, casoBase):
    if casoBase == 0:
        return
    else:
        casoBase -= 1

        self.DrawCube(longitud, angulo)

        longitud /= 3

        self.a.forward(longitud)
        self.a.left(angulo)
        self.a.forward(longitud)
        self.a.forward(longitud)
        self.a.right(angulo)
        self.a.pendown()

        self.DrawCube(longitud, angulo)

        self.AlfombraSierpinski(longitud, factor, angulo, casoBase)

        self.a.forward(longitud)
        self.a.forward(longitud)
        self.a.forward(longitud)
        self.a.pendown()

        self.DrawCube(longitud, angulo)

        self.AlfombraSierpinski(longitud, factor, angulo, casoBase)

        self.a.right(angulo)
        self.a.forward(longitud)
        self.a.forward(longitud)
        self.a.forward(longitud)
        self.a.left(angulo)
        self.a.pendown()

        self.DrawCube(longitud, angulo)

        self.AlfombraSierpinski(longitud, factor, angulo, casoBase)

        self.a.right(angulo)
        self.a.forward(longitud)
        self.a.forward(longitud)
        self.a.forward(longitud)
```

```
self.a.left(angulo)
self.a.pendown()

self.DrawCube(longitud, angulo)

self.AlfombraSierpinski(longitud, factor, angulo, casoBase)

self.a.right(angulo)
self.a.right(angulo)
self.a.forward(longitud)
self.a.forward(longitud)
self.a.forward(longitud)
self.a.right(angulo)
self.a.right(angulo)
self.a.pendown()

self.DrawCube(longitud, angulo)

self.AlfombraSierpinski(longitud, factor, angulo, casoBase)

self.a.right(angulo)
self.a.right(angulo)
self.a.forward(longitud)
self.a.forward(longitud)
self.a.forward(longitud)
self.a.right(angulo)
self.a.right(angulo)
self.a.pendown()

self.DrawCube(longitud, angulo)

self.AlfombraSierpinski(longitud, factor, angulo, casoBase)

self.a.left(angulo)
self.a.forward(longitud)
self.a.forward(longitud)
self.a.forward(longitud)
self.a.right(angulo)
self.a.pendown()

self.DrawCube(longitud, angulo)

self.AlfombraSierpinski(longitud, factor, angulo, casoBase)

self.a.left(angulo)
self.a.forward(longitud)
self.a.forward(longitud)
self.a.forward(longitud)
```

```
            self.a.right(angulo)
            self.a.pendown()

            self.DrawCube(longitud, angulo)

            self.AlfombraSierpinski(longitud, factor, angulo, casoBase)

            self.a.forward(longitud)
            self.a.forward(longitud)
            self.a.right(angulo)
            self.a.forward(longitud)
            self.a.forward(longitud)
            self.a.left(angulo)
```

```
class Pinta(Tree, Koch, Carpet):            #Clase pinta que hereda de Tree y de Koch y tiene funcion para inicializar todos los parametros de los padres y los atributos necesarios para imprimir el fractal, funciones para rellenar los atributos y funciones para imprimir los dos fractales
    def __init__(self, longitud, factor, angulo, casoBase):
        super().__init__(longitud, factor, angulo, casoBase)

        self.oculta = 0
        self.rumbo = 0
        self.color = "red"

    def DameOculta(self, oculta):
        if oculta == 1:
            self.oculta = 1

    def DameRumbo(self, rumbo):
        self.rumbo = rumbo

    def DameColor(self, color):
        self.color = color

    def PintaArbol(self):
        if self.oculta == 1:
            self.a.hideturtle()
        self.a.setheading(self.rumbo)
        self.a.color(self.color)

        self.Arbol(self.longitud, self.factor, self.angulo, self.casoBase)

    def PintaKoch(self):
```

```
        if self.oculta == 1:
            self.a.hideturtle()
        self.a.setheading(self.rumbo)
        self.a.color(self.color)

        for i in range(3):
            self.CurvaKoch(self.longitud, self.factor, self.angulo, self.casoBase)
            self.a.right(120)

    def PintaAlfombra(self):
        if self.oculta == 1:
            self.a.hideturtle()
        self.a.setheading(self.rumbo)
        self.a.color(self.color)

        self.AlfombraSierpinski(self.longitud, self.factor, self .angulo, self.casoBase)

class Windows():            #Clase windows que crea una ventana de menu grafica con
botones para elegir que fractal quieres y llama a las funciones de pedir informacion para el
fractal
    def __init__(self, principal):
        self.title = "Fractales"
        self.longitud = 0
        self.factor = 0
        self.angulo = 0
        self.casoBase = 0
        self.oculto = 0
        self.rumbo = 0
        self.color = "red"

        self.principal = principal

        principal.title(self.title)

        self.etiqueta = Label(principal, text="Escoge que fractal quieres crear")
        self.etiqueta.pack()

        self.botonArbol = Button(principal, text="Arbol", command=self.VArbol)
        self.botonArbol.pack()

        self.botonCurva = Button(principal, text="Koch", command=self.VKoch)
        self.botonCurva.pack()

        self.botonSierpinski = Button(principal, text="Sierpinski", command=self.VSierpinski)
        self.botonSierpinski.pack()

        self.botonLimpiar = Button(principal, text="Limpiar", command=self.Clear)
        self.botonLimpiar.pack()
```

```
        self.botonCerrar = Button(principal, text="Cerrar", command=principal.quit)
        self.botonCerrar.pack()

    def Clear(self):
        global pinta

        if pinta != 0:
            pinta.a.clear()

    def VArbol(self):          #Funcion VArbol que crea un objeto Pinta y le pasa la informacion necesaria, despues llama a la funcion de pintar pintaArbol
        global pinta

        l = 50
        f = 10
        a = 45
        c = 5
        o = 1
        r = 90
        clr = "red"

        pinta = Pinta(l, f, a, c)
        pinta.DameOculta(o)
        pinta.DameRumbo(r)
        pinta.DameColor(clr)
        pinta.PintaArbol()

    def VKoch(self):          #Funcion VKoch que crea un objeto Pinta y le pasa la informacion necesaria, despues llama a la funcion de pintar pintaKoch
        global pinta

        l = 300
        f = 3
        a = 60
        c = 4
        o = 1
        r = 0
        clr = "blue"

        pinta = Pinta(l, f, a, c)
        pinta.DameOculta(o)
        pinta.DameRumbo(r)
        pinta.DameColor(clr)
        pinta.PintaKoch()

    def VSierpinski(self):          #Funcion VSierpinski que crea un objeto Pinta y le pasa la informacion necesaria, despues llama a la funcion de pintar pintaAlfombra
```

```
        global pinta

        l = 100
        f = 30
        a = 90
        c = 2
        o = 1
        r = 0
        clr = "green"

        pinta = Pinta(l, f, a, c)
        pinta.DameOculta(o)
        pinta.DameRumbo(r)
        pinta.DameColor(clr)
        pinta.PintaAlfombra()



#----------CODIGO----------#

root = Tk()         #Guarda el objeto de ventana en la variable root
miWindows = Windows(root)           #Guarda el objeto Windows al que le pasa la ventana
root en la variable miWindows
root.mainloop()         #Se encarga de crear un bucle en la ventana main para gestionar lo
que pasa en ella
```