Diagram

Description automatically generated

*# This is a place to practice creating fractals with python using info from  
# https://towardsdatascience.com/creating-fractals-with-python-d2b663786da6  
# by Dhanesh Budhrani***import** turtle  
  
MINIMUM\_BRANCH\_LENGTH = 5 *# Sets the minimum threshold to create further sub-branches***def** build\_tree(t, branch\_length, shorten\_by, angle):  
 *"""* **:param** *t: Turtle instance* **:param** *branch\_length: the current length of the branch in pixels* **:param** *shorten\_by: determines by how many pixels the sub-branches will be shorter than the parent branch* **:param** *angle: the angles from which the sub-branches emerge from the parent branch* **:return***:  
 """* **if** branch\_length > MINIMUM\_BRANCH\_LENGTH:  
 t.forward(branch\_length)  
 new\_length = branch\_length - shorten\_by  
  
 t.left(angle)  
 build\_tree(t, new\_length, shorten\_by, angle)  
  
 t.right(angle \* 2)  
 build\_tree(t, new\_length, shorten\_by, angle)  
  
 t.left(angle)  
 t.backward(branch\_length)  
  
  
tree = turtle.Turtle()  
tree.hideturtle()  
tree.setheading(90)  
tree.color(**'green'**)  
  
build\_tree(tree, 50, 5, 30)  
turtle.mainloop()

A picture containing shape

Description automatically generated

**import** turtle  
  
  
**def** koch\_curve(t, iterations, length, shortening\_factor, angle):  
 *"""* **:param** *t: Turtle instance* **:param** *iterations: represents the value of n in the image below this list  
 note that n=0 would represent a flat line, which will be the  
 base case in the recursive function* **:param** *length: the length of each side in our current (sub-) snowflake* **:param** *shortening\_factor: determines the factor by which the side length  
 is divided when we create a new sub-snowflake* **:param** *angle: determines the angle from which the new side emerges* **:return***:  
 """* **if** iterations == 0:  
 t.forward(length)  
 **else**:  
 iterations = iterations-1  
 length = length / shortening\_factor  
  
 koch\_curve(t, iterations, length, shortening\_factor, angle)  
 t.left(angle)  
 koch\_curve(t, iterations, length, shortening\_factor, angle)  
 t.right(angle \* 2)  
 koch\_curve(t, iterations, length, shortening\_factor, angle)  
 t.left(angle)  
 koch\_curve(t,iterations, length, shortening\_factor, angle)  
  
  
t = turtle.Turtle()  
t.speed(0)  
t.color(**"blue"**)  
t.hideturtle()

**for** i **in** range(3):  
 koch\_curve(t, 4, 200, 3, 60)  
 t.right(120)  
  
turtle.mainloop()