

It was suggested by a classmate that my lab reports be in Google doc format, rather than my previous method of adding comments and showing updated code within a copy of the provided Colab. I do not know which method is preferred, but will use this as my lab format for this week, and if you would like and will allow it, I will go back to my earlier weeks and use this format as well.

Like previously I will walk through the code one block at a time, breaking down individual pieces.

```
# Function to draw a fractal tree using recursion
def fractal_tree(length, depth):
    if depth == 0:
        return
    else:
        forward(length)
        left(30)

        fractal_tree(length * 0.7, depth - 1)

        right(60)

        fractal_tree(length * 0.7, depth - 1)

        left(30)
        backward(length)

# Initialize the turtle
initializeTurtle(initial_window_size=(500, 500))

# Set up the turtle's starting position and orientation
jump(250, 450) # Starting position at the bottom center of the canvas
face(0)
color('green') # Set the pen color

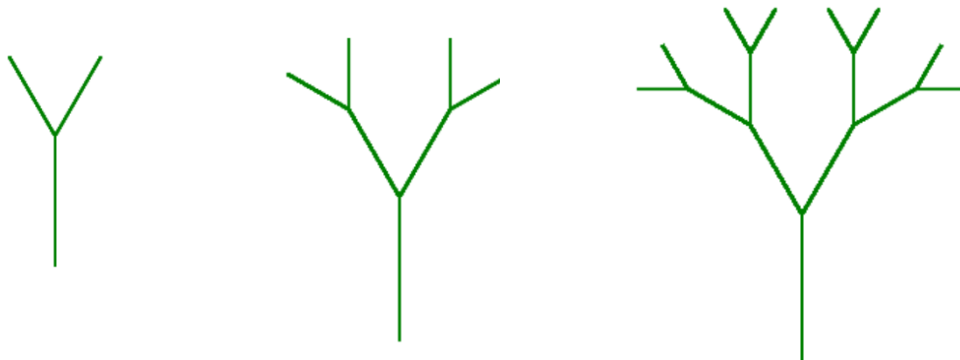
# Set the length and depth for the fractal tree
length = 100 # Initial length of the main trunk
depth = 7 # Recursion depth (number of branching levels)

# Draw the fractal tree
fractal_tree(length, depth)

# Display the drawing
show()
```

The first block of code defines the tree. The turtle moves forward a given length, initially 100. It turns left 30 degrees and moves 70% of the given length. After drawing the left stem, the pointer is returned to the branch tip, so to speak, and is turned right 60% which is net 30% from due north. It then creates the same smaller stem to the right before returning back to the base of the branch.

If the depth is 1 no side stems are created so testing with a value of 2 gives the most basic form, seen below on the left. Changing the depth to 3 shows the first level of recursion in that each of the side branches create further side branches, lengths 70% of the new side branches. You can view the added layer of complexity by looking at the image in the middle. Adding one more layer of complexity shows the new branches developing branches of their own as seen on the right image.



The next two blocks of code are shown here:

```
# Define the fractal tree function
def build_tree(branch_length, shorten_by, angle):
    if branch_length > 3:

        forward(branch_length)

        new_length = branch_length - shorten_by

        left(angle)

        build_tree(new_length, shorten_by, angle)

        right(angle * 2)

        build_tree(new_length, shorten_by, angle)

        left(angle)

        backward(branch_length)
```

```
# Initialize the turtle
initializeTurtle(initial_window_size=(800, 800))

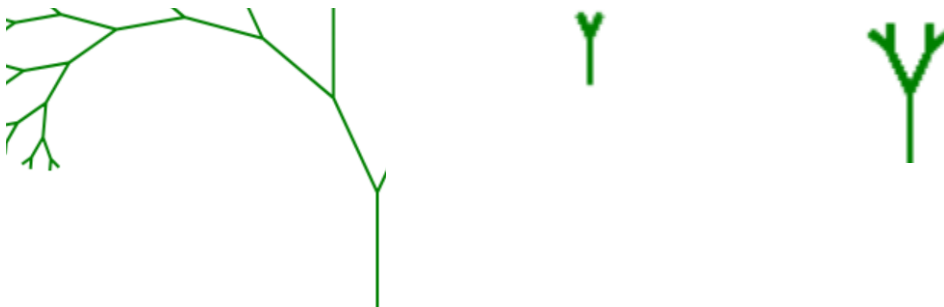
# Set up the turtle's starting position and orientation
jump(400, 600) # Starting position at the bottom center of the canvas
face(0)
color('green') # Set the pen color

# Draw the fractal tree
build_tree(80, 8, 25)

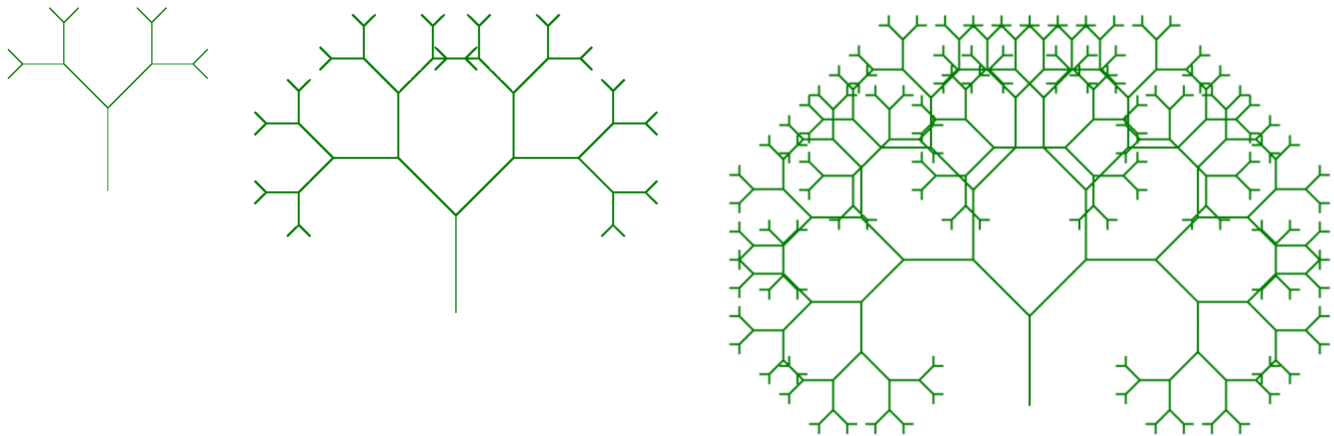
# Display the drawing
show()
```

The first thing to note is that the `build_tree` subroutine requires three values: the branch length, how much to reduce the branch length by on the recursive stems, and the angle to turn each time, which is different from the prior code in that the angles there were given. The pattern for developing the angles remains unchanged in that you turn left the given angle, then right double that amount, and then back before returning to the larger stem.

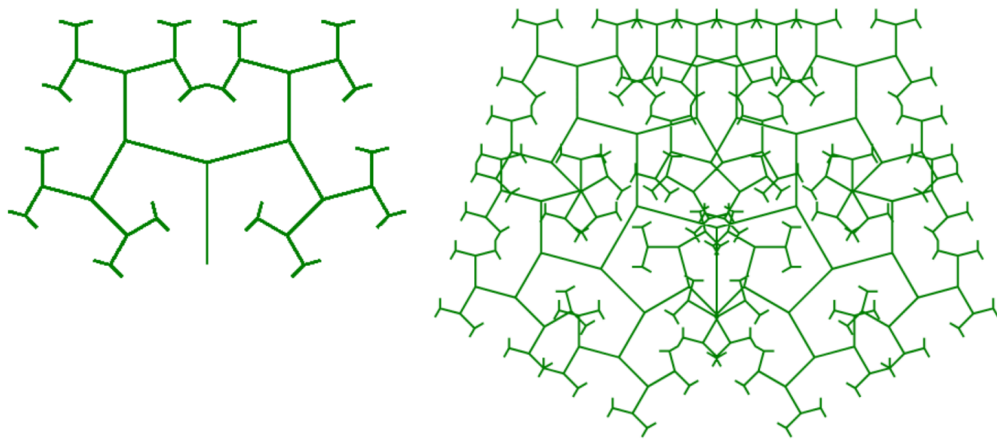
On the initial code there are 10 iterations of the branches of the lengths 80, 72, 64, 56, 48, 40, 32, 24, 16, and 8. This can be seen from any one branching path, the most obvious ones being the outer edges, seen on the left. Changing the branch length to 16 gives only two iterations as seen below in the center, and then expanding one time more gives three as seen on the right.



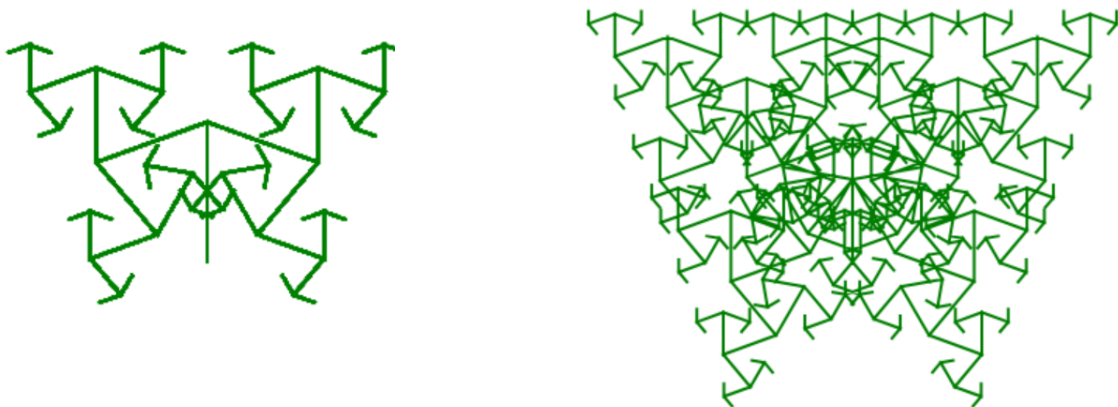
I wanted to play with the angles next. The initial values of 25 degrees are seen above, but changing the value to 45 degrees creates branches that will either be horizontal, vertical, or have slopes of 1 or -1 which is shown below at different levels of complexity.



Seeing the tree start to double up on itself has me curious to see what happens when the angles are between 45 and 90 degrees, 75 degrees in particular this time:



When the angles become obtuse, 110 degrees in this case, the tree loses all semblance of its original form.



I used AI within Colab using the prompt 'create a fractal tree with a different color fractal root' and the following was generated.

```
# prompt: create a fractal tree with a different color fractal root

from turtle_setup import *
# Function to draw a fractal tree using recursion
def fractal_tree(length, depth, color):
    if depth == 0:
        return
    else:
        forward(length)
        left(30)

        fractal_tree(length * 0.7, depth - 1, color)

        right(60)

        fractal_tree(length * 0.7, depth - 1, color)

        left(30)
        backward(length)
# Initialize the turtle
initializeTurtle(initial_window_size=(500, 500))

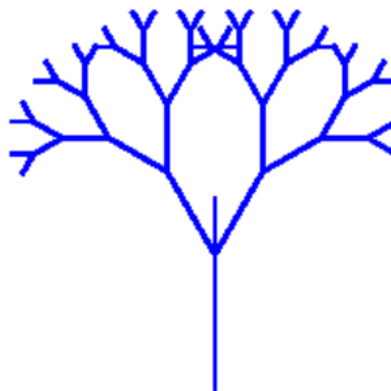
# Set up the turtle's starting position and orientation
jump(250, 450) # Starting position at the bottom center of the canvas
face(0)

# Set the length and depth for the fractal tree
length = 100 # Initial length of the main trunk
depth = 7    # Recursion depth (number of branching levels)

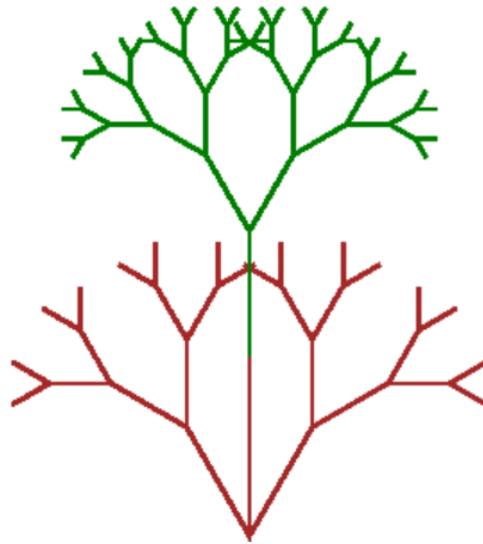
# Draw the fractal tree with a brown root
fractal_tree(length, 1, "brown") # Draw the root in brown
fractal_tree(length * 0.7, depth - 1, "green") # Continue the tree with green branches

# Display the drawing
show()
```

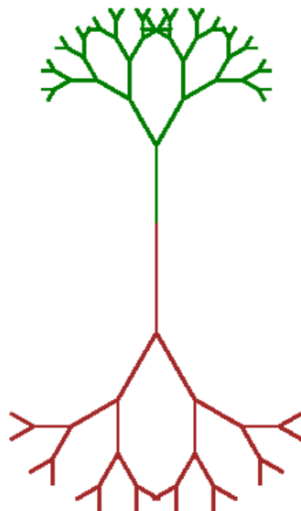
The recursive function maintains the constant angle like the original one, but adds color as an input. The purpose of this as seen at the calls to the function are so that two different tree calls can be made, one for the branch in green and one for the roots in brown. On the initial call though it creates the tree, but in neither color!



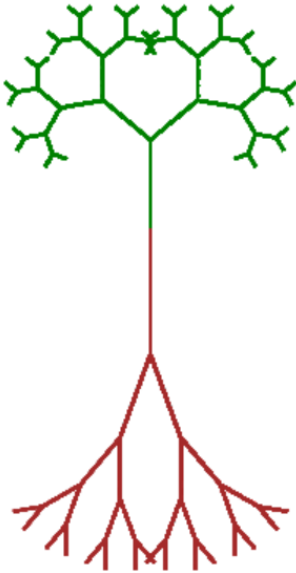
The function is being called, but not working as it should. The first change was to fix the color. I changed the variable 'color' to 'treecolor' and added a line within the function: color(treecolor). The next step is to get the function to create the roots, for which the intent is to start at the same base but create the branches upside down, as a 180 degree reflection. The call for the roots currently does not do that. My initial thought is to create a second function called fractal_root. I was torn between that versus adding conditional statements within the fractal_tree function, but think this method is more efficient despite using more lines of code. In the root command the only change I made was to switch the backward and forward commands, but that does not work as you will see.



It starts off correctly but the angles are going the wrong direction. I added face commands prior to the function calls, and by doing so, was able to get rid of the second function I created. I'm not a fan of not incorporating it into the function itself, but will leave it for now.



Lastly, I like the idea of using adding the angle function so that the branches can spread at a different angle from the roots so I added that as an input value and updated the function calls. I'm really happy with how this came out so below is a sample as well as the code.



```
# prompt: create a fractal tree with a different color fractal root

from turtle_setup import *
# Function to draw a fractal tree using recursion
def fractal_tree(length, depth, treecolor, angle):
    color(treecolor)
    if depth == 0:
        return
    else:
        forward(length)
        left(angle)

        fractal_tree(length * 0.7, depth - 1, treecolor, angle)

        right(angle*2)

        fractal_tree(length * 0.7, depth - 1, treecolor, angle)

        left(angle)
        backward(length)

# Initialize the turtle
initializeTurtle(initial_window_size=(500, 500))

# Set up the turtle's starting position and orientation
jump(250, 250) # Starting position at the bottom center of the canvas
face(0)

# Set the length and depth for the fractal tree
length = 80 # Initial length of the main trunk
depth = 7   # Recursion depth (number of branching levels)

# Draw the fractal tree with a brown root
face(180)
fractal_tree(length, 5, "brown", 20) # Draw the root in brown
face(0)
fractal_tree(length * 0.7, depth - 1, "green", 50) # Continue the tree with green branches

# Display the drawing
show()
```