In [ ]:

```
#Copyright (C) 2021, Sydney Nwakanma
#May 5th, 2021
# Description: Hello world Python that uses a recursive function to
              #generate a "fractal mountainous skyline" and to display
              #the results of the computation using matplotlib.pyplot
#Inputs: input from the keyboard
#Outputs: displays ASCII text to stdout
#Assumptions: written/tested with Python 3.9.1 on Windows
#Dependencies: matplotlib, time, datetime, and random

import matplotlib.pyplot as plt
from datetime import datetime as dt
import random
import time


#compute A non-recursive function that computes a new midpoint (x', y')
#offset midpoint by an amount that contains an amount that is
#proportional to the distance between x1 and x2,
#and  a random value
def get_midpoint_and_yoffset (x1,y1,x2,y2, scale, sign_bit):
    xnot = (x2-x1)*0.5 + x1
    ynot = (y2-y1)*0.5 + y1

    offset = scale * (x2-x1) * random.random() #
    #if sign_bit: offset = 20
    ynot += offset
    return xnot, ynot

#A recursive function for invoking a midpoint computation function,
#inserting the midpoint into the x- and y-values lists/arrays,
#testing for the base or recursive case,
#and either terminating or recursing as appropriate

def midpoint_divide(x_vals, y_vals, i, scale, recursion_depth, sign_bit):
    print("midpoint_divide: recursion_depth={}".format(recursion_depth))
    xnot,ynot = get_midpoint_and_yoffset(x_vals[i], y_vals[i],
                                         x_vals[i+1], y_vals[i+1], scale, sign_bit)
    x_vals.insert(i+1, xnot)
    y_vals.insert(i+1, ynot)
    if (recursion_depth == 1):
        print("You reached the bottom of the barrel")
        return
    else:
        print("need to do midpoint subdivision")
        midpoint_divide(x_vals, y_vals, i+1, scale, recursion_depth-1,
                        not(sign_bit))
        midpoint_divide(x_vals, y_vals, i, scale,  recursion_depth-1,
                        not(sign_bit))
        return




def main():

    #start of main
```

```
    print("Sydney Nwakanma") #print name

    t = dt.today()
    print(t) #print date

  #Assign values to input variables
    recursion_depth_max = 1
    scale = 0.5
    sign_bit = 1


    # prompt user for a positive integer for recursion depth
    recursion_depth_max = int(input("enter a positive integer for recursion depth"))

 #invoke recursive function using the loop variable as the depth of recursion
    for i in range(1, recursion_depth_max+1):
        print(i)
        x_vals = [0.0, 10.0]
        y_vals = [0.0, 0.0]

        #call the midpoint function
        midpoint_divide(x_vals, y_vals, 0, scale,  i, sign_bit)

       #plot results
        plt.plot(x_vals, y_vals, "green")
        plt.show()

   #then pause (sleep) for 2 seconds and proceed to the next loop iteration
        time.sleep(2)
        # return



if __name__=="__main__":
     main()

#end file
```