Lab 10

Exercise 1

```
'''This function uses recursion to reverse te order of a list.'''
def reverse(l):
    if len(l) == 0: #Base case
        return []
    else:
        return l[-1:] + reverse(l[:-1]) #Recurisve call

list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print("The original list:", list)
print("Reversed list:", reverse(list))
```

Outputs:

The original list: [1, 2, 3, 4, 5, 6, 7, 8, 9]
Reversed list: [9, 8, 7, 6, 5, 4, 3, 2, 1]

The original list: [1, 2, 3, 4, 2, 54, 7, 8, 9]
Reversed list: [9, 8, 7, 54, 2, 4, 3, 2, 1]

The original list: [1, 2, 3, 4, 2, 54, 7, 8, 9]
Reversed list: [9, 8, 7, 54, 2, 4, 3, 2, 1]

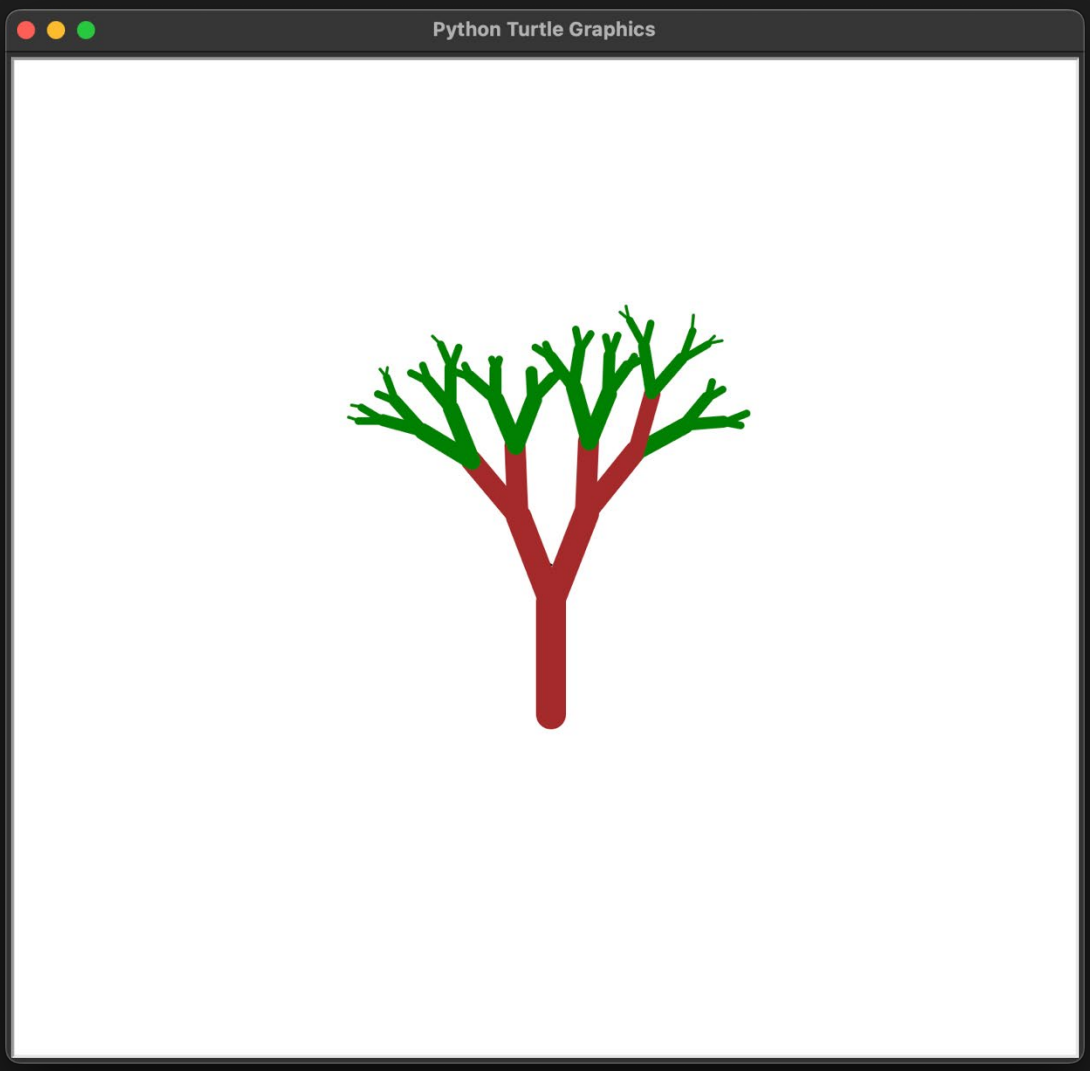
The original list: [1, 22, 234235, 4, 324, 54, 7, 58, 2349]
Reversed list: [2349, 58, 7, 54, 324, 4, 234235, 22, 1]

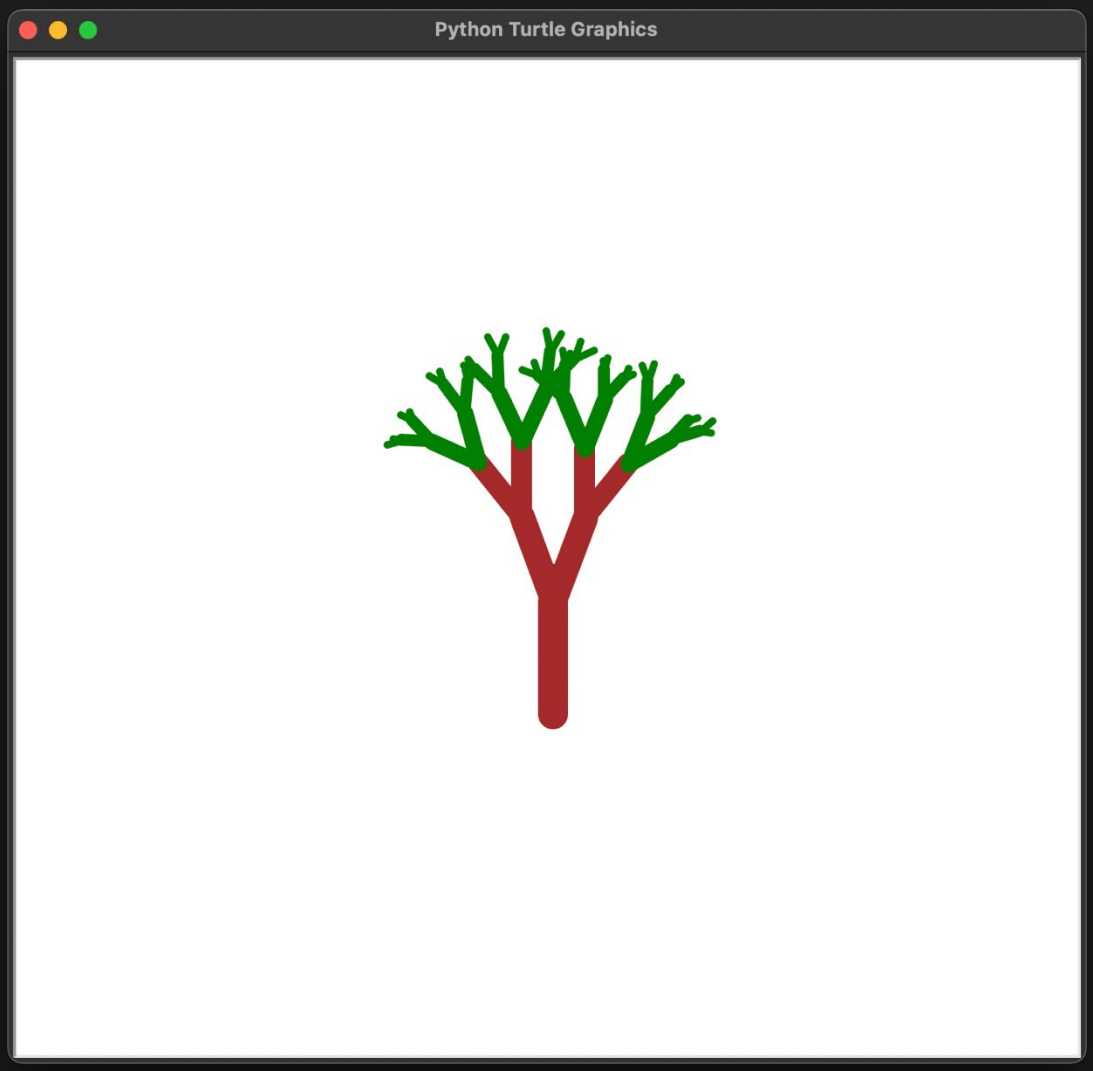
Exercise 2

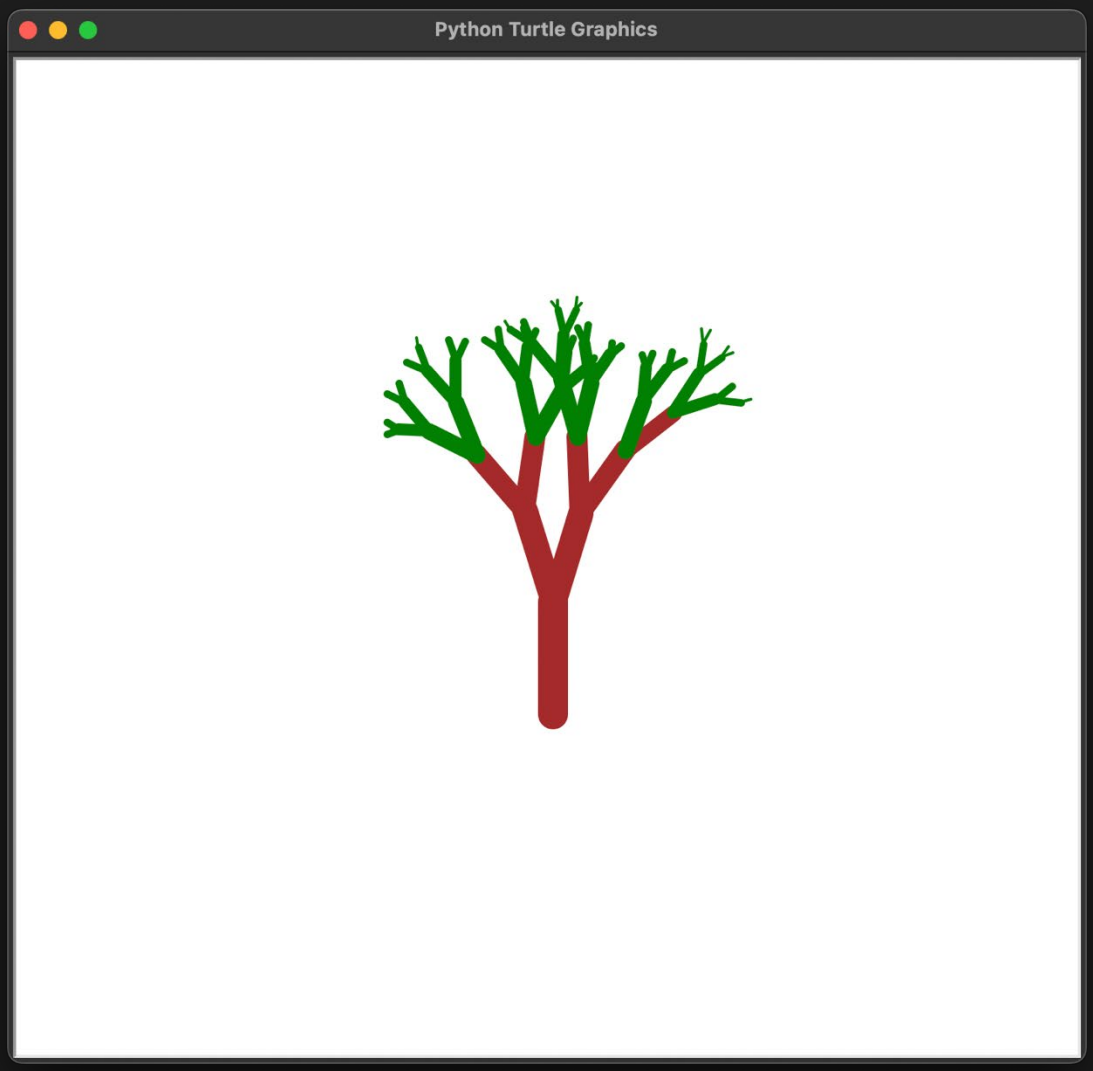
```
import turtle
import random

def tree(branchLen,t,w, n):
    if branchLen > 5:
        '''We added this if statement to check if the branch length is less than a certain amount, so we can change the
color.'''
```

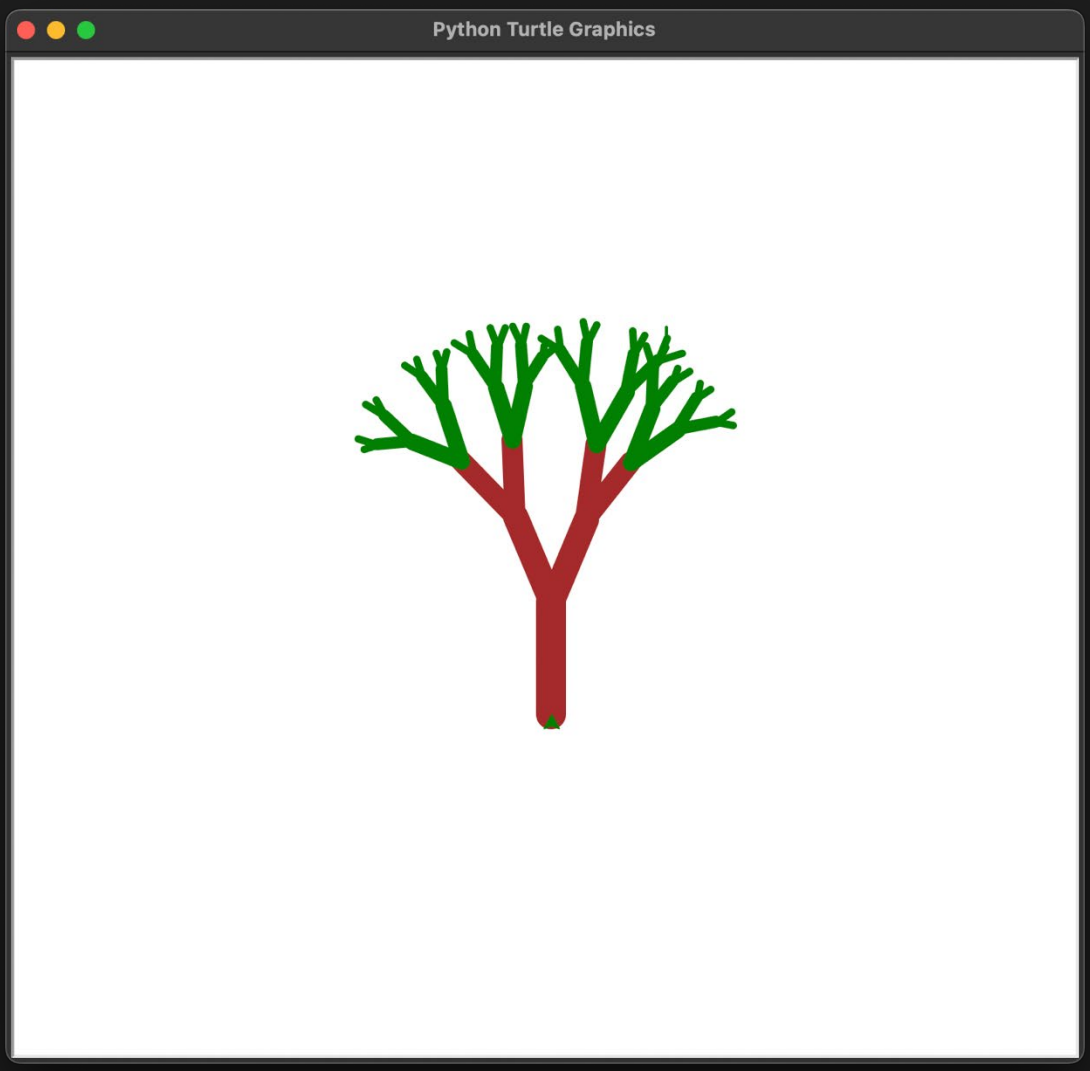
```
        if branchLen > 40: #Base case
            t.color("brown")

        else:
            t.color("green")
        '''We need the turtle to go down here, as later we tell it to go up to avoid going back with the wrong color later'''
        t.down()
        t.width(w)
        t.forward(branchLen)
        t.right(n)
        tree(branchLen-random.randint(10,15),t, w-3,random.randint(15,25)) #Recursive call
        t.left(n * 2)
        tree(branchLen-random.randint(10,15),t, w-3,random.randint(15,25)) #Recursive call
        t.right(n)
        '''We need this up command here, as to avoid creating issues with the line going backwards and drawing the
wrong color.'''
        t.up()
        t.backward(branchLen)

def main():
    t = turtle.Turtle()
    myWin = turtle.Screen()
    t.left(90)
    t.up()
    t.backward(100)
    t.down()
    '''We added a variable called w for width. We will use this for adjusting the width as the tree is created.'''
    w = 20
    t.color("brown")
    tree(75,t,w, random.randint(15,25))
    myWin.exitonclick()

if __name__ == '_main__':
    main()
```

Outputs


Python Turtle Graphics


Python Turtle Graphics

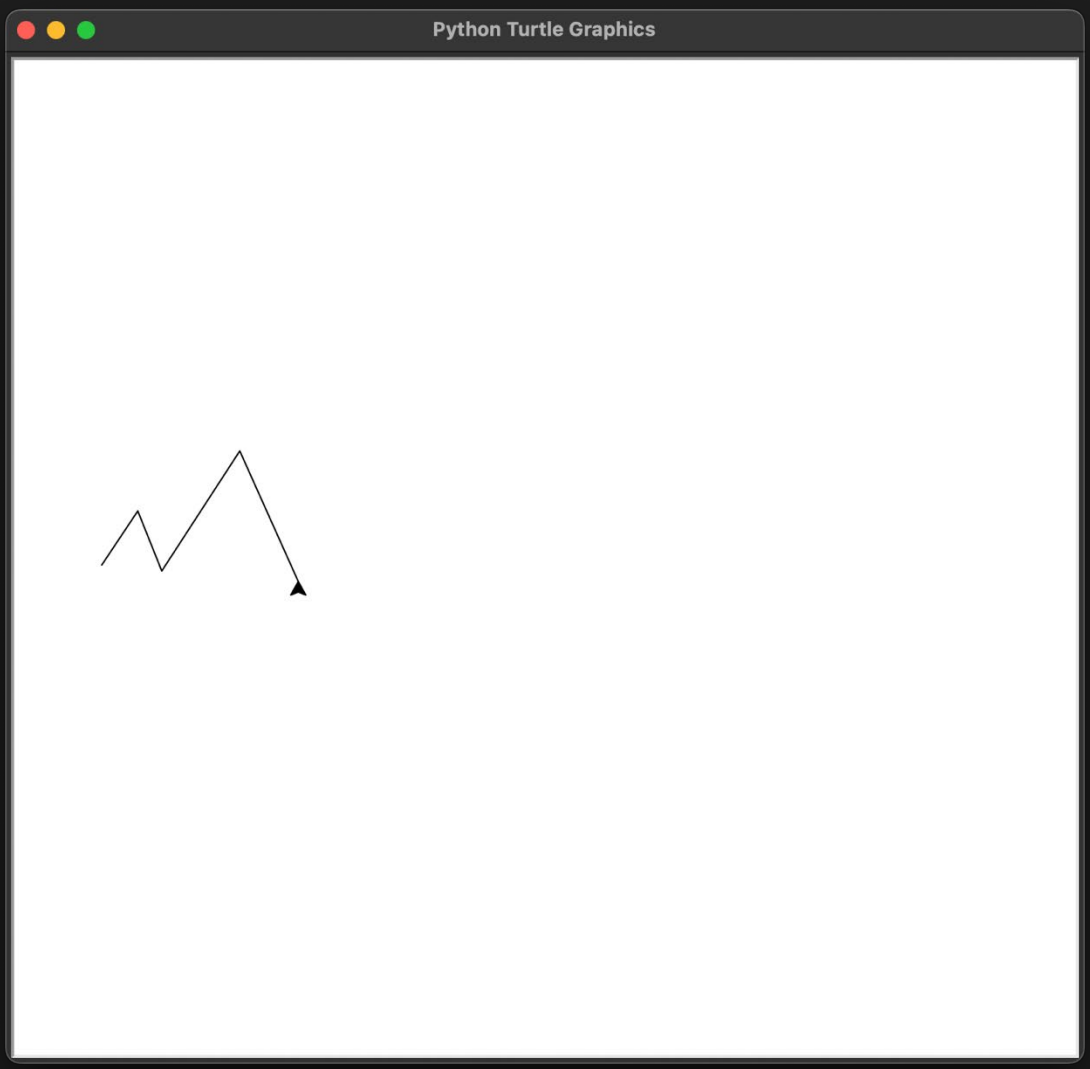

Python Turtle Graphics

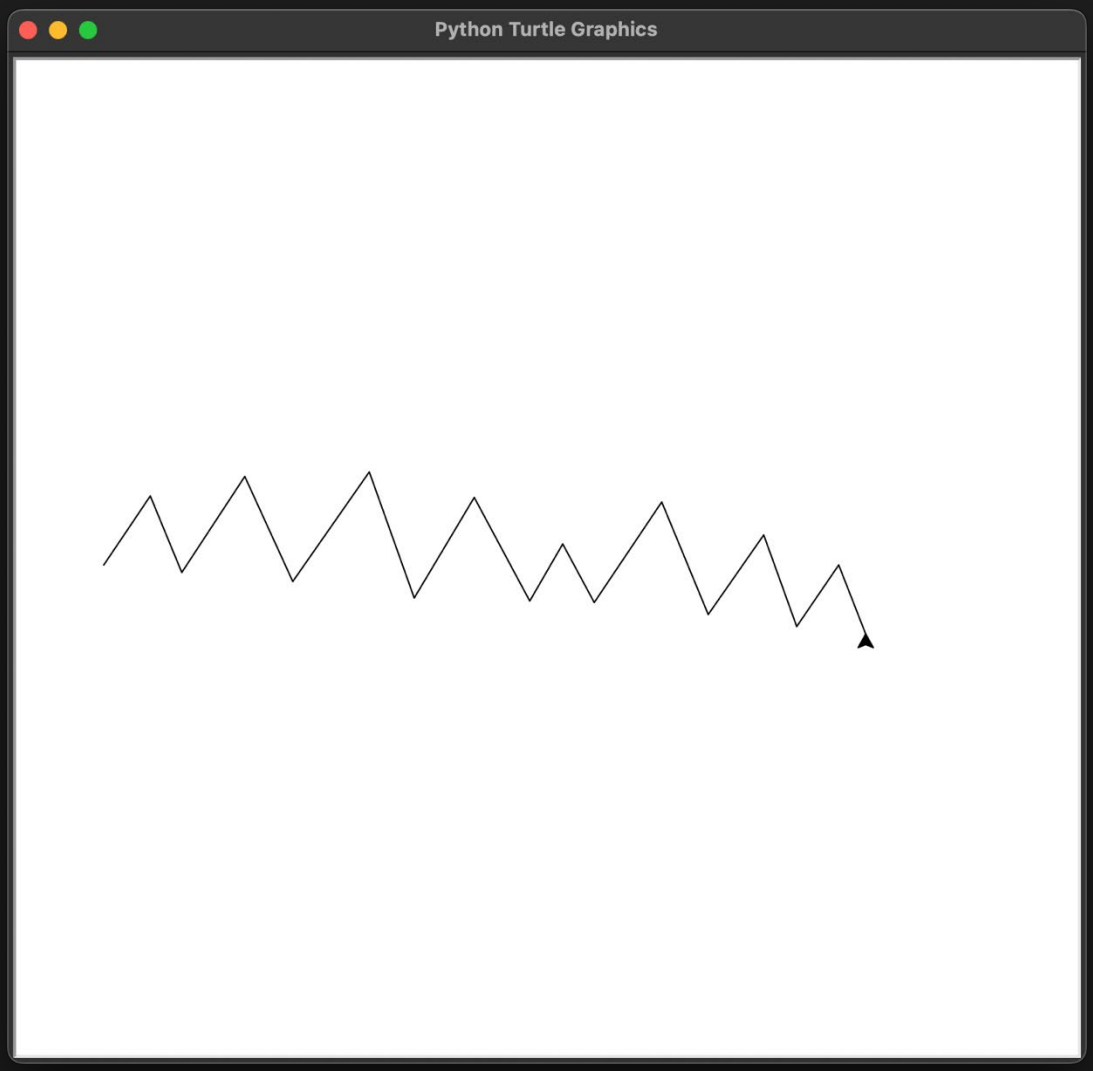

Python Turtle Graphics

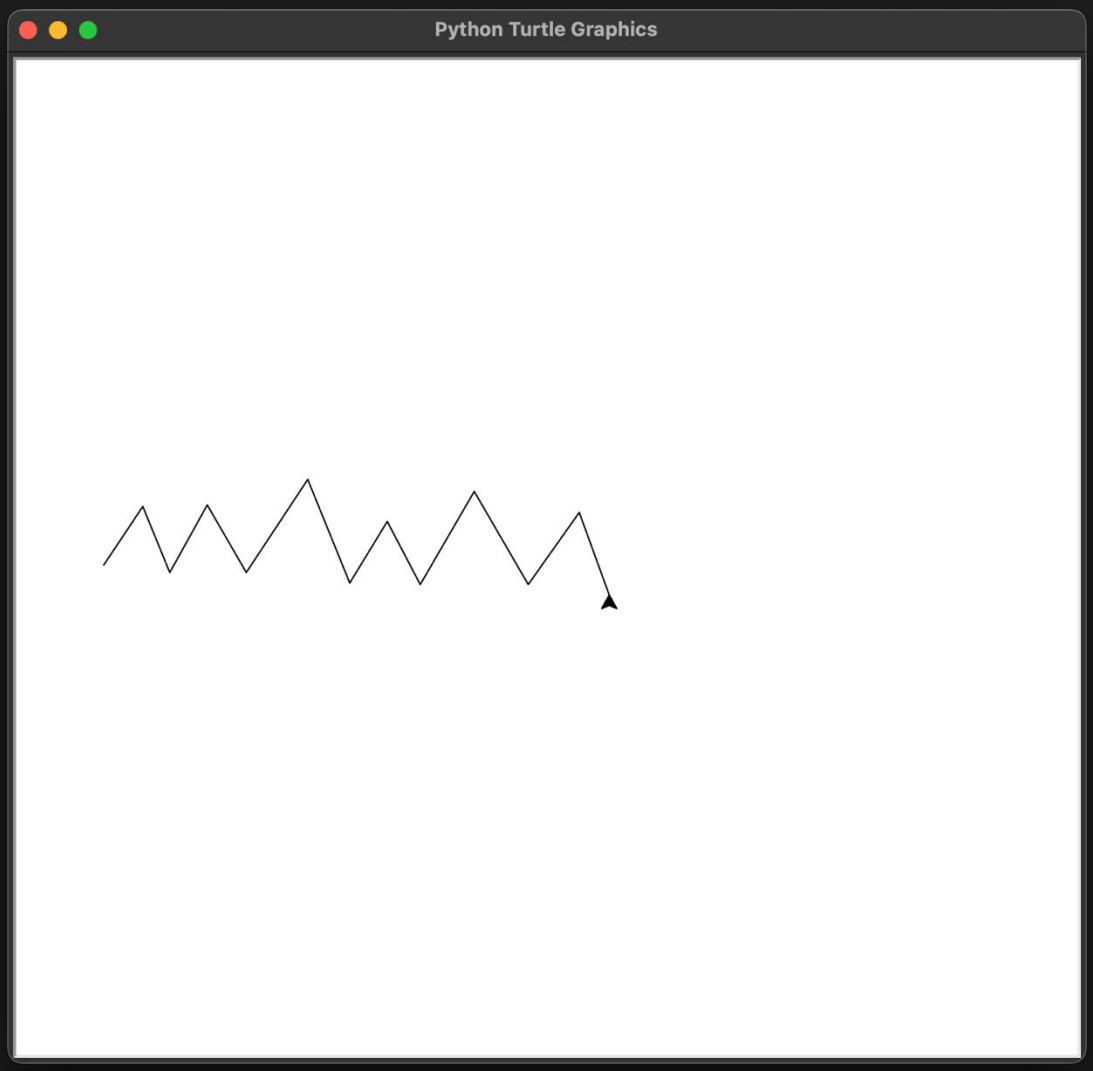
Exercise 3

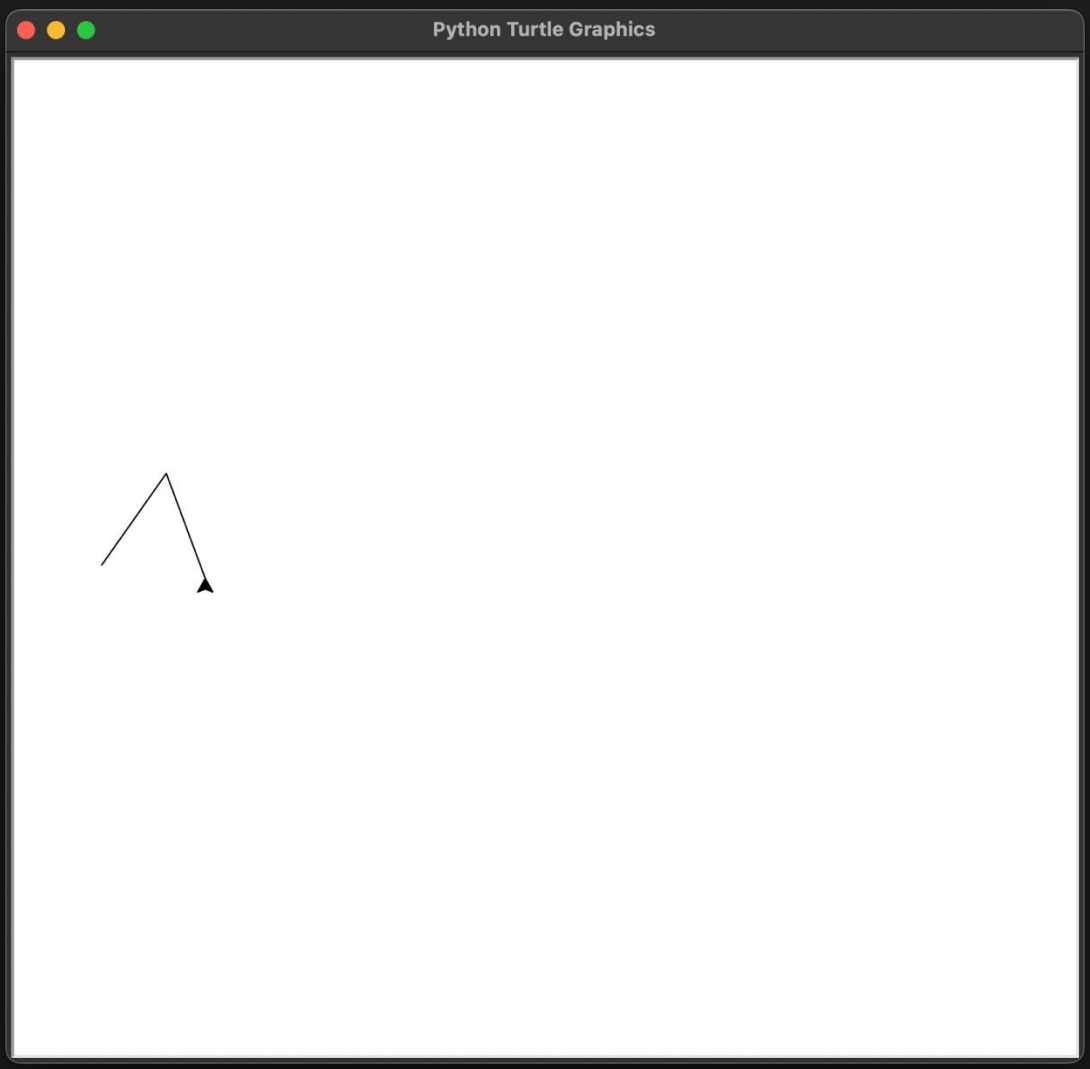
```
'''Imports'''
import turtle
import random

'''This function cretes a mountain using recursion.'''
def mountain(l, t,n):
    if l > 38: #Base case
        t.right(n)
        t.forward(l)
        t.right(n + 90)
        t.forward(l)
        t.left(n * 2 + 90)
        mountain(random.randint(30,100), t, random.randint(30,35)) #Recursive call

'''The main function of the program.'''
def main():
    '''This area just gets the turtle in the right spot to draw'''
    t = turtle.Turtle()
    myWin = turtle.Screen()
    t.up()
    t.backward(300)
    t.down()
    t.setheading(90)
    '''Mountain call. Uses a slightly random angle, as well as a random length.'''
    mountain(random.randint(30, 100), t, random.randint(30,35))
    myWin.exitonclick()

if __name__ == "__main__":
    main()
```

Outputs:


Python Turtle Graphics


Python Turtle Graphics


Python Turtle Graphics


Python Turtle Graphics

Exercise 4

```
'''This is the fibonocci number generator.'''
def fib(n, o, x, number):
    if x == number: #Base case
        return n
    elif x % 2 != 0:
        '''We found we need to have this these if statements to update each number.'''
        return fib(n+o, o, x+1, number) #Recursive call
    else:
        return fib(n, n + o, x + 1, number) #Recursive call

'''Main function of the program.'''
def main():
    number = int(input("Enter a number: "))
    print(fib(0,1,0, number))

if __name__ == '__main__':
    main()
```

Outputs:

Enter a number: 30
832040

Enter a number: 7
8

Enter a number: 100
354224848179261915075

Enter a number: 420
26627102054807356173464520221007550748090234072083744418019196048455636386781458494514 40