Lesson 5
Fractals & Recursion

Programming Fundamentals in Python

Lesson 4 Recap

- Homework: Pong
- Bonus: Breakout

Class Materials

github.com/DavidYKay/python-fundamentals

Today’s Goal

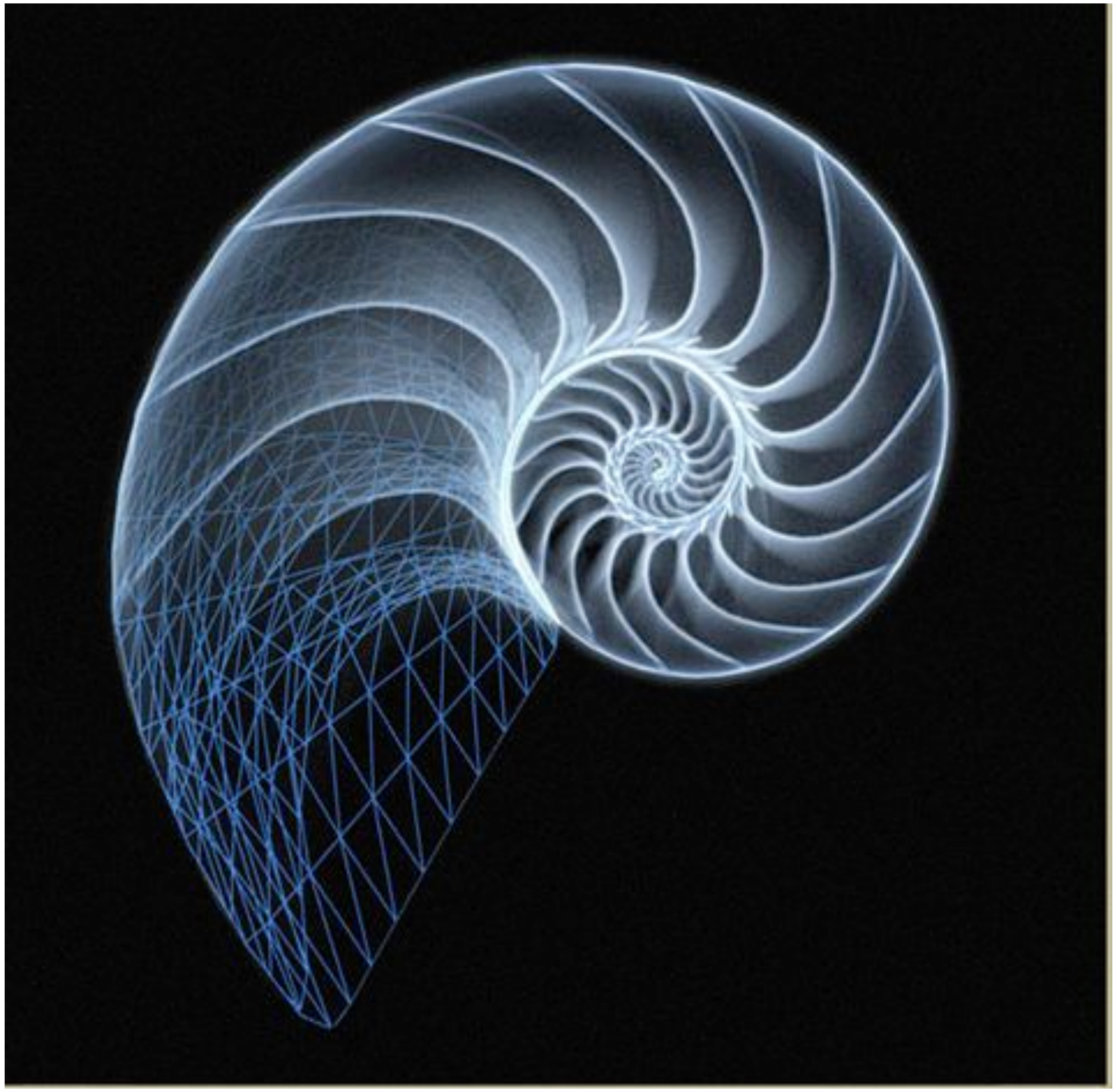
- Implement a game of Pong

Sierpinski Triangle Demo

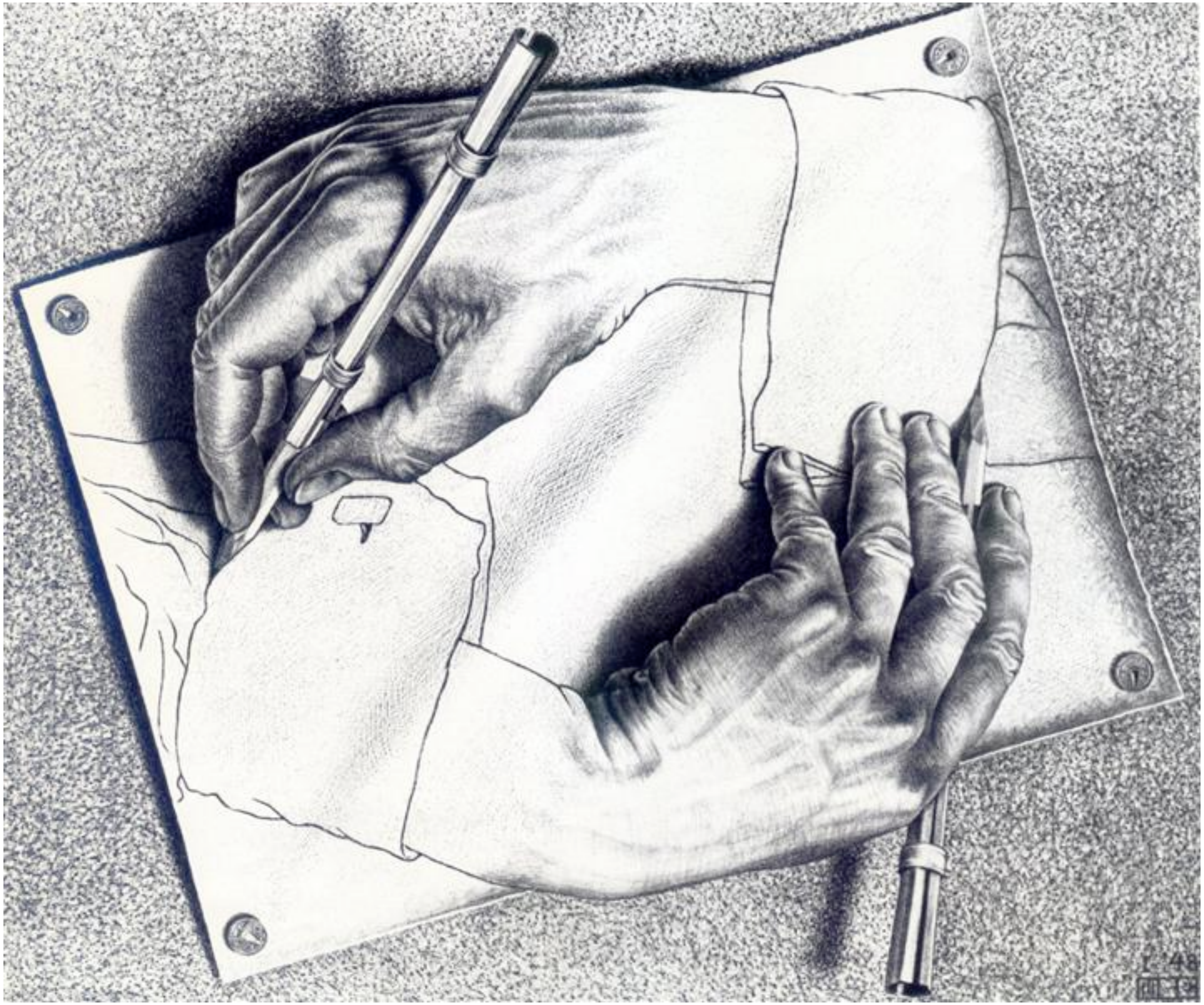
Breakdown

- Recursion
  - Factorial
  - Fibonacci
  - Family Tree
- Turtle

Recursion



Recursion



Recursion

Recursion

- Formally equivalent to iteration
- Often very elegant
- Some limitations in Python

Components of Recursion

- **Recursive definition**: “Go deeper. Here’s how.”
- **Base Case**: Ground Truth / “Where do I stop?”

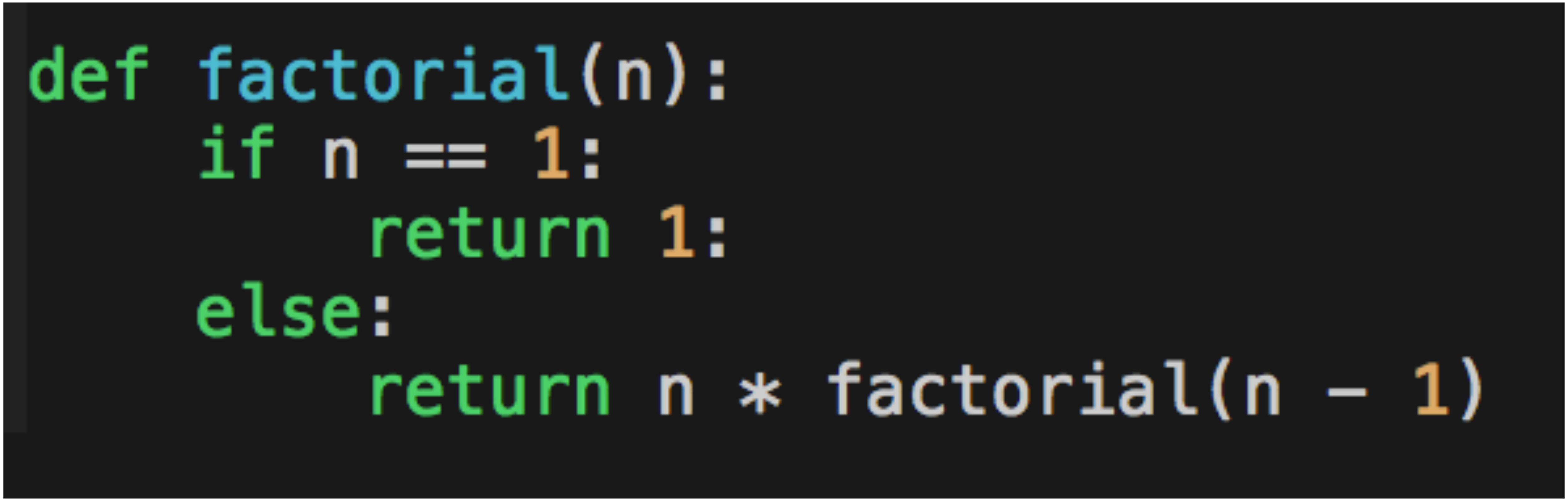
Factorial

$4!$

Factorial

$$4! = 4 * 3 * 2 * 1$$

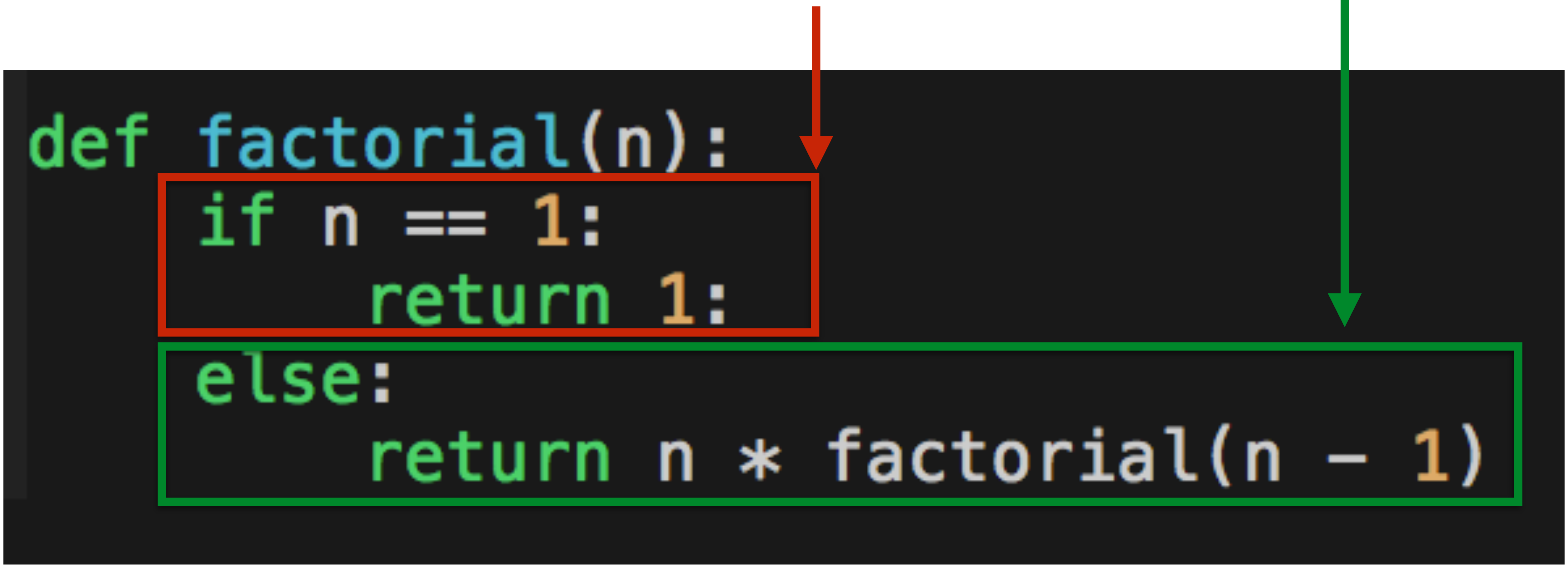
Factorial



```
def factorial(n):
    if n == 1:
        return 1:
    else:
        return n * factorial(n - 1)
```

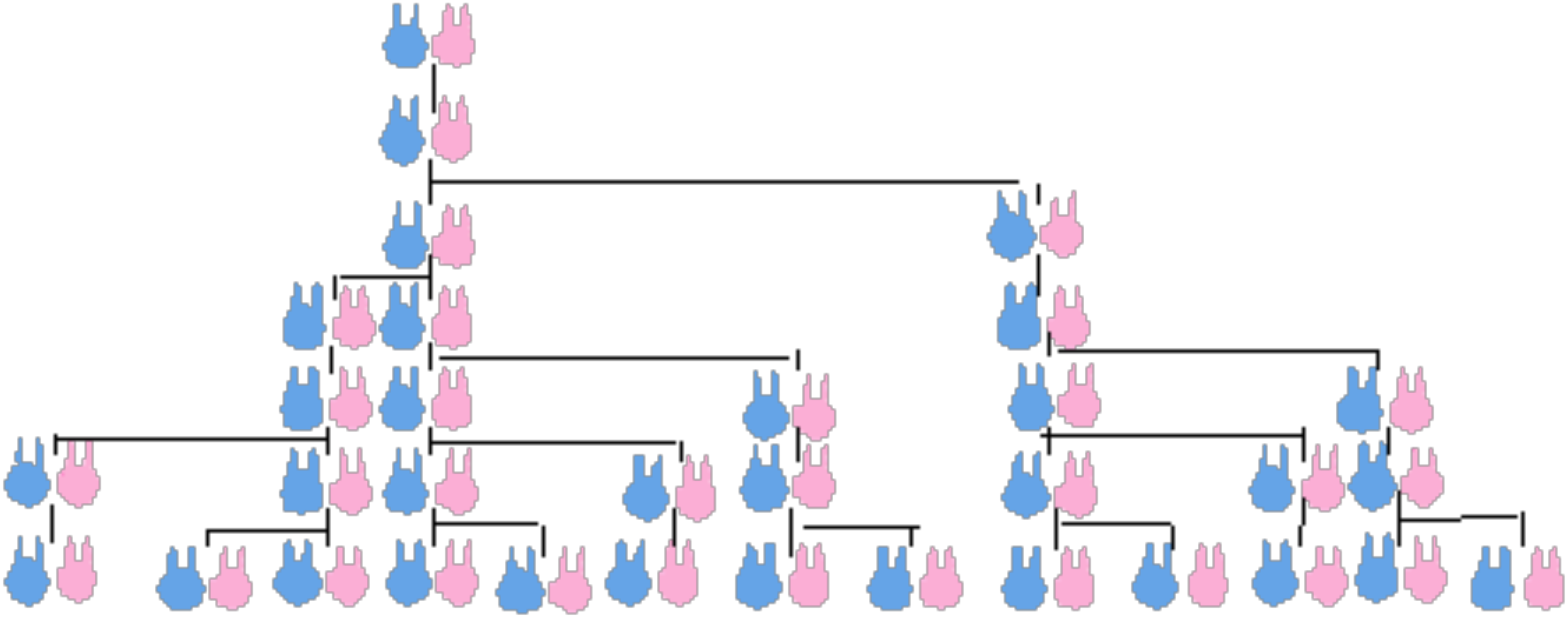
Recursive Definition

Base Case



```
def factorial(n):
    if n == 1:
        return 1:
    else:
        return n * factorial(n - 1)
```

Fibonacci

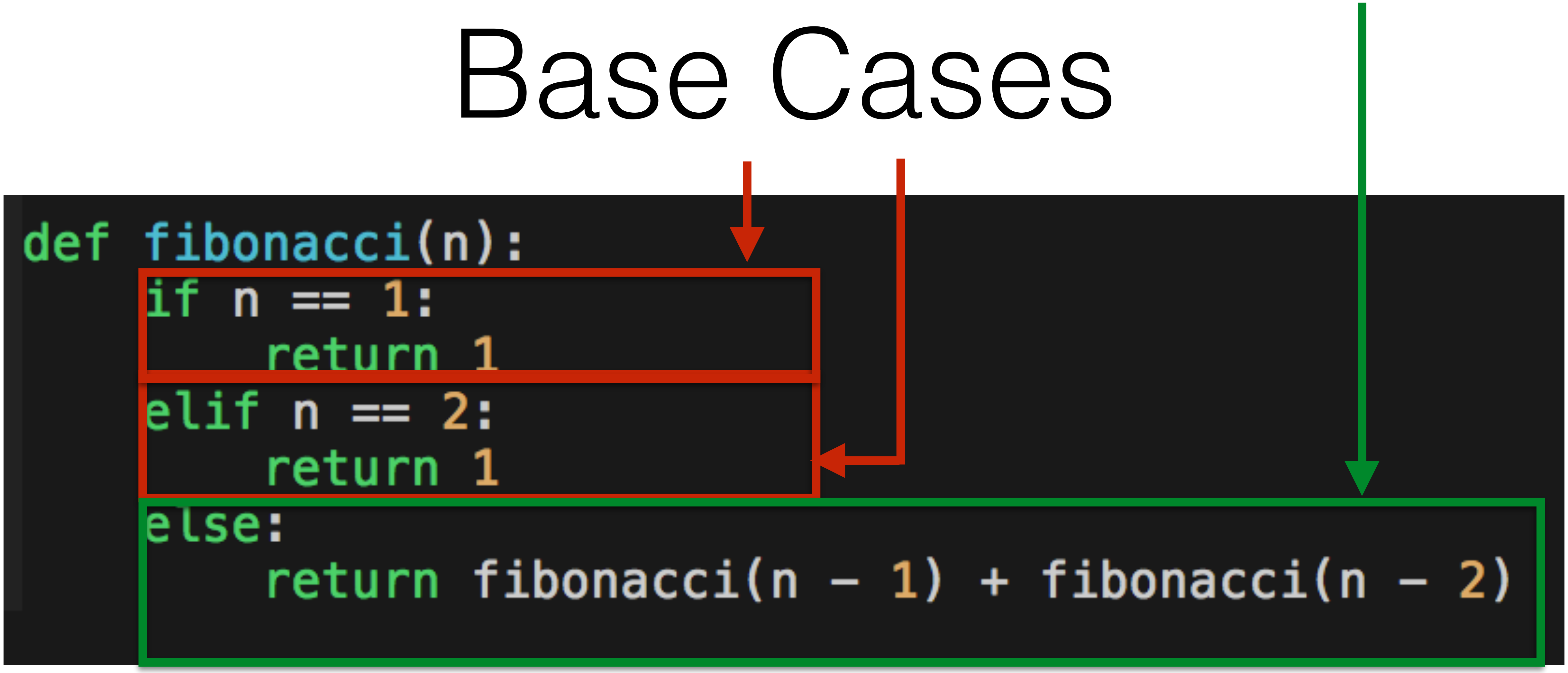


Fibonacci

1, 1, 2, 3, 5, 8, 13, 21

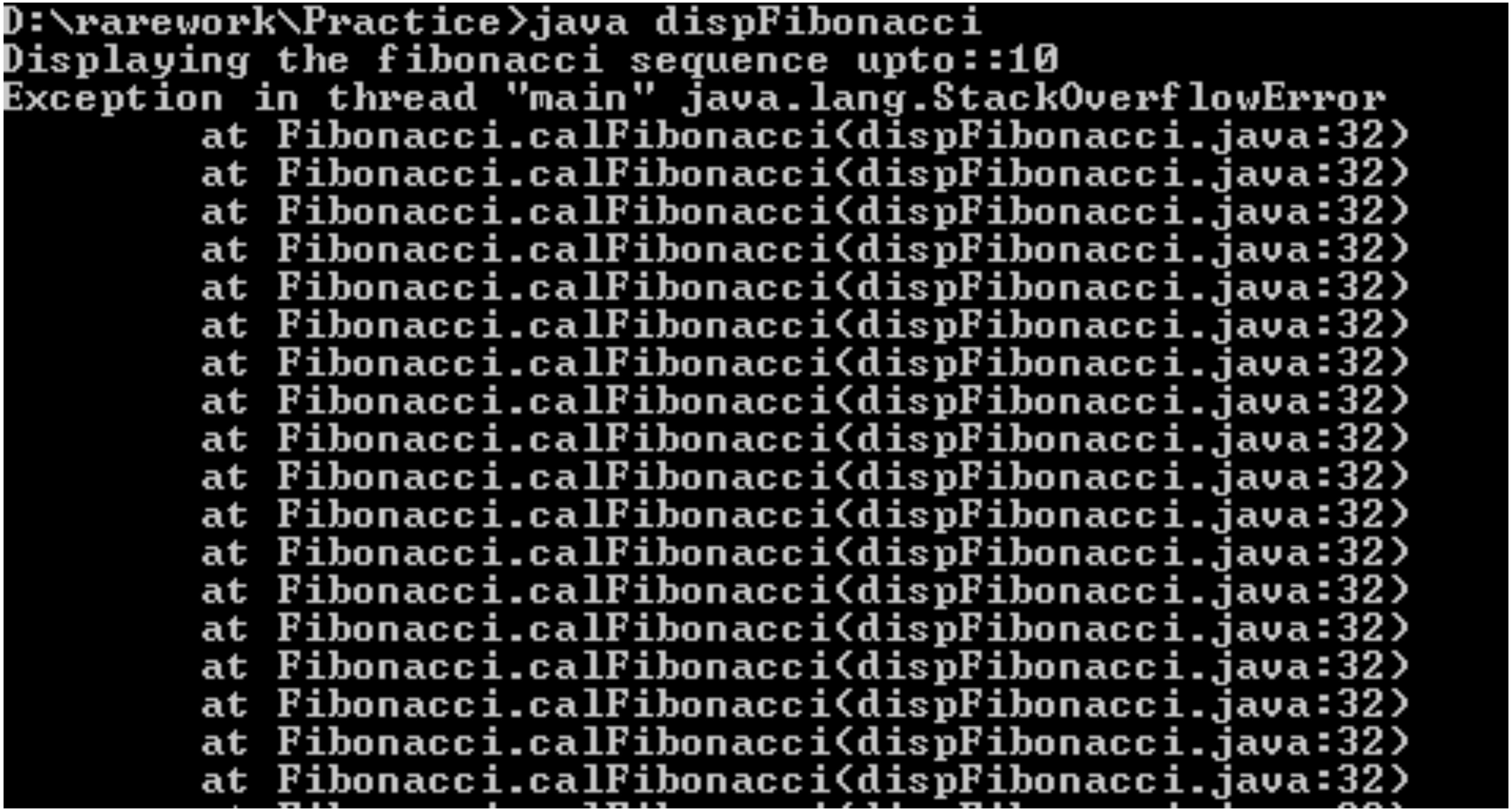
Recursive Definition

Base Cases



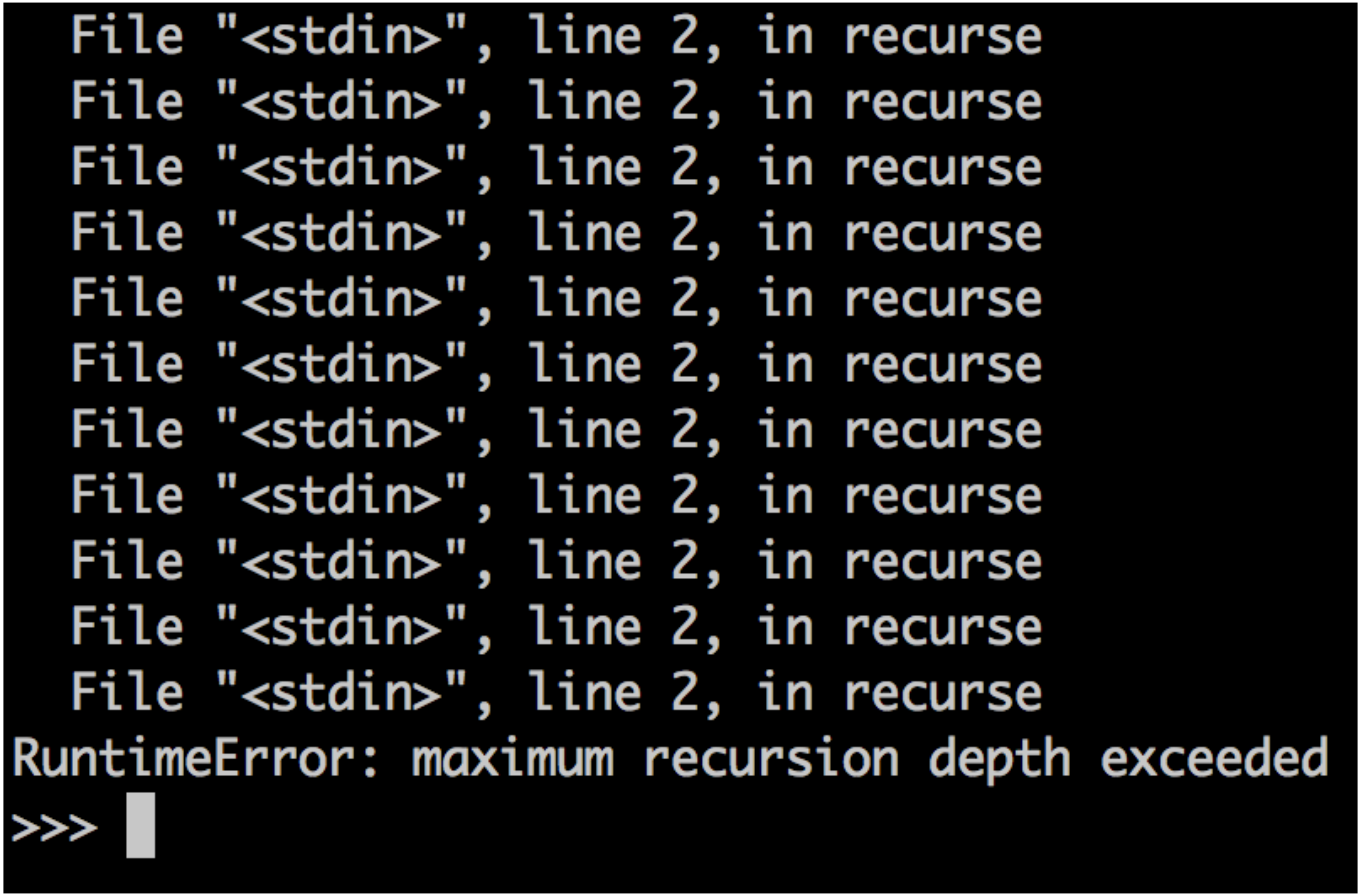
```python
def fibonacci(n):
    if n == 1:
        return 1
    elif n == 2:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
```

Stack Overflow



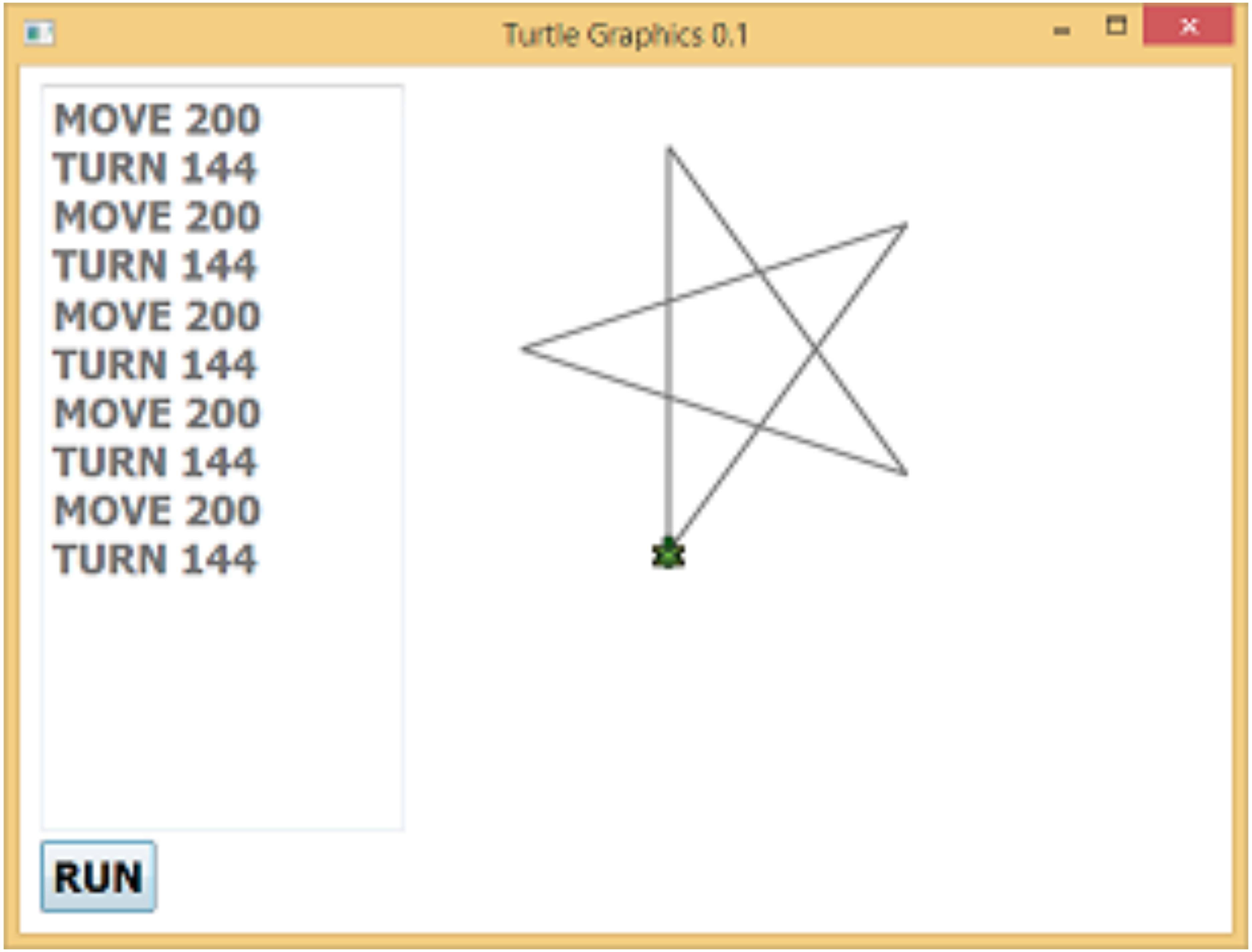
```
D:\rarework\Practice>java dispFibonacci
Displaying the fibonacci sequence upto::10
Exception in thread "main" java.lang.StackOverflowError
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
        at Fibonacci.calFibonacci(dispFibonacci.java:32)
```

Stack Overflow

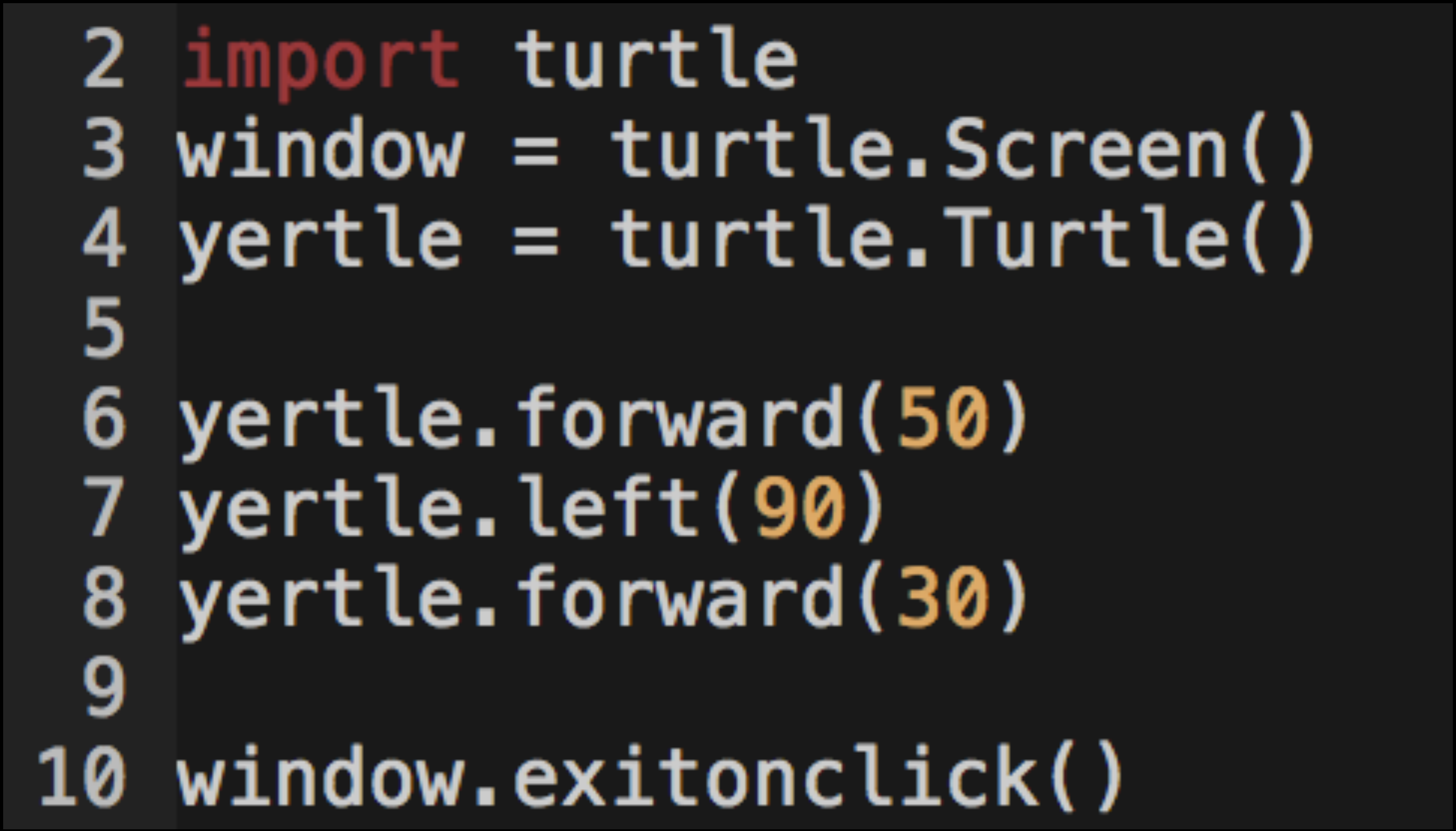


```
  File "<stdin>", line 2, in recurse
  File "<stdin>", line 2, in recurse
  File "<stdin>", line 2, in recurse
  File "<stdin>", line 2, in recurse
  File "<stdin>", line 2, in recurse
  File "<stdin>", line 2, in recurse
  File "<stdin>", line 2, in recurse
  File "<stdin>", line 2, in recurse
  File "<stdin>", line 2, in recurse
  File "<stdin>", line 2, in recurse
  File "<stdin>", line 2, in recurse
RuntimeError: maximum recursion depth exceeded
>>> 
```

Turtle



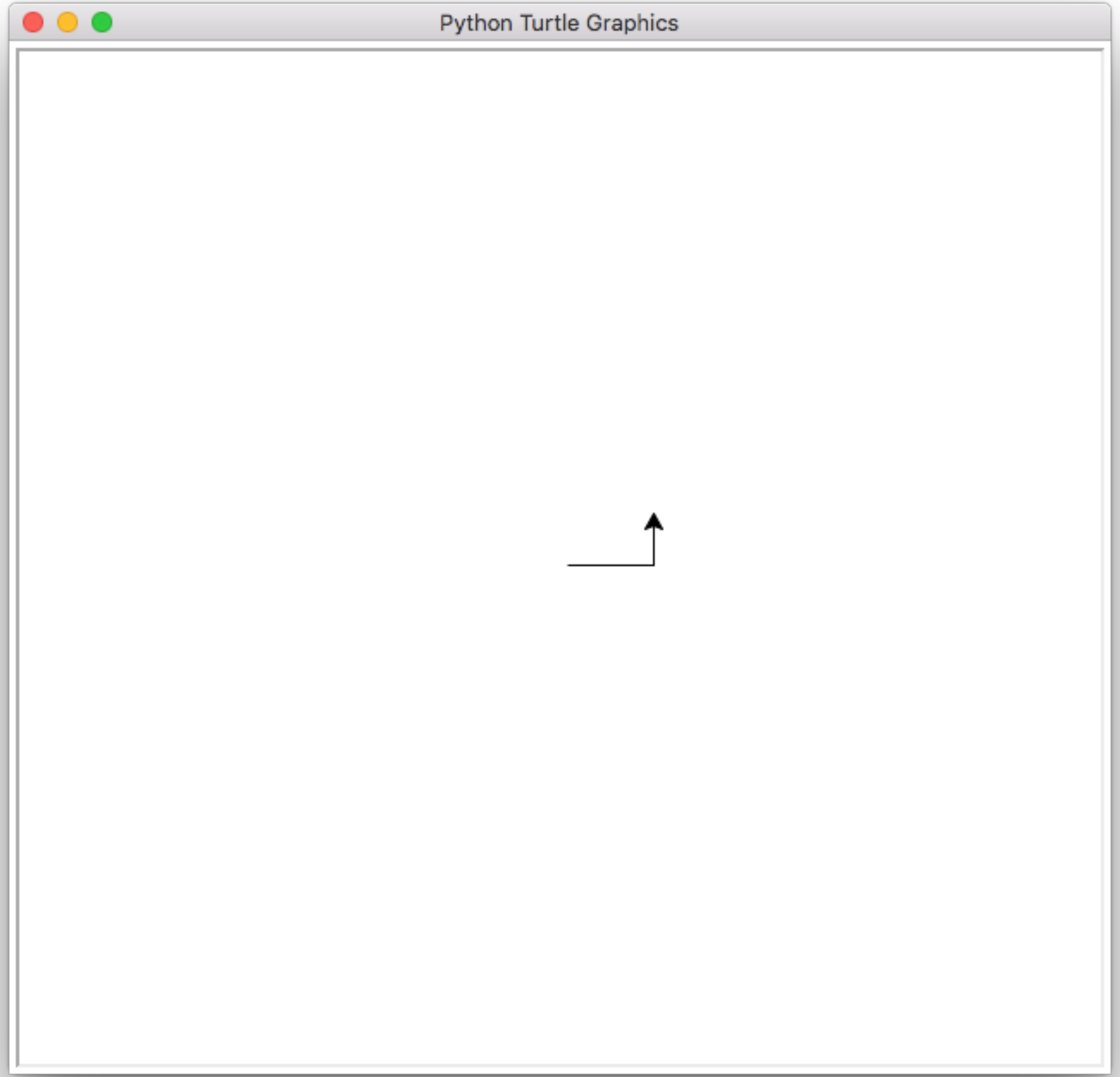
```
MOVE 200
TURN 144
MOVE 200
TURN 144
MOVE 200
TURN 144
MOVE 200
TURN 144
MOVE 200
TURN 144
```



```python
import turtle
window = turtle.Screen()
yertle = turtle.Turtle()

yertle.forward(50)
yertle.left(90)
yertle.forward(30)

window.exitonclick()
```



Python Turtle Graphics

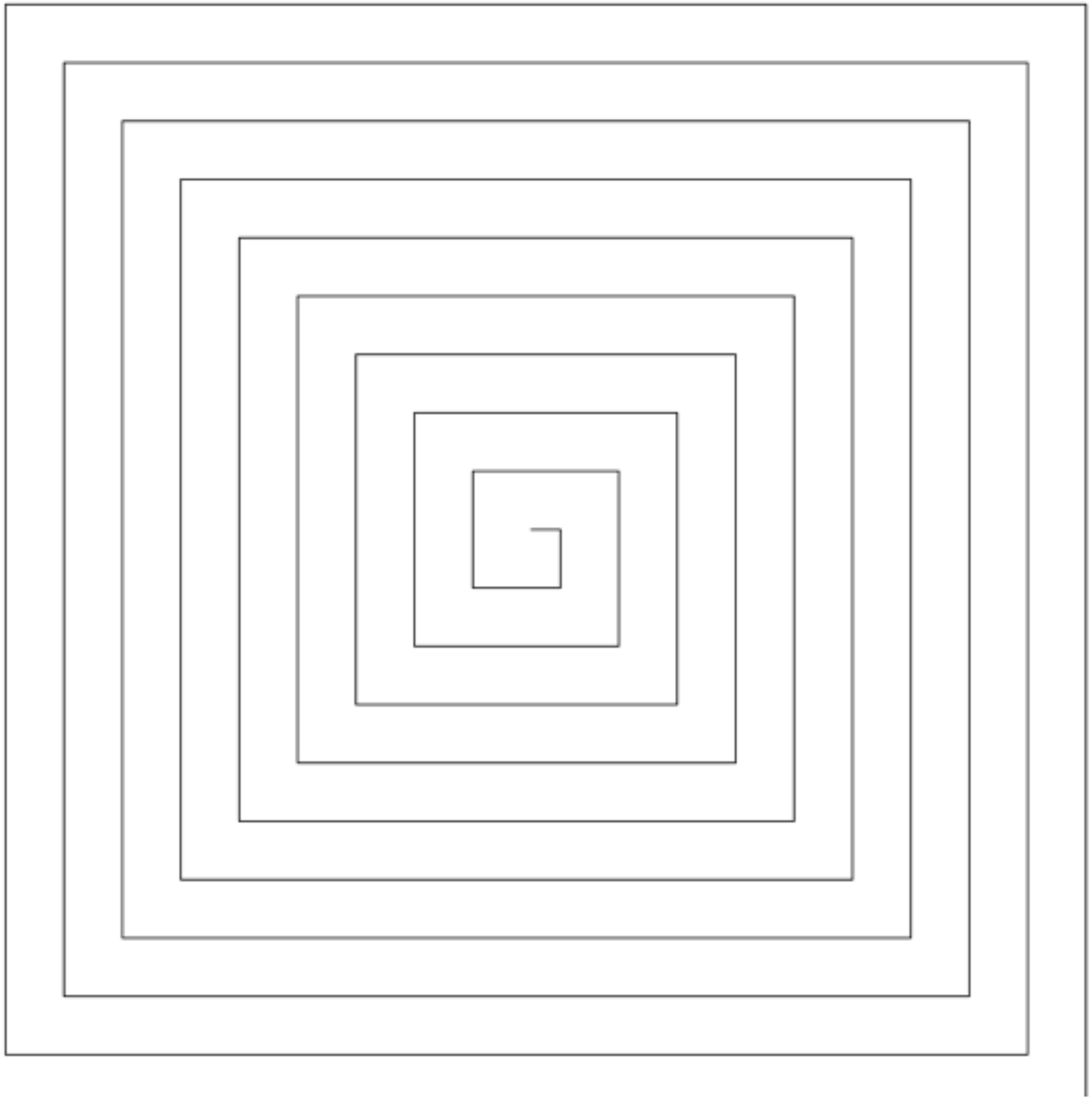
Recap

- Recursion
    - Factorial
    - Fibonacci
    - Family Tree
- Turtle

Easy Assignment

- Implement the Fibonacci Spiral

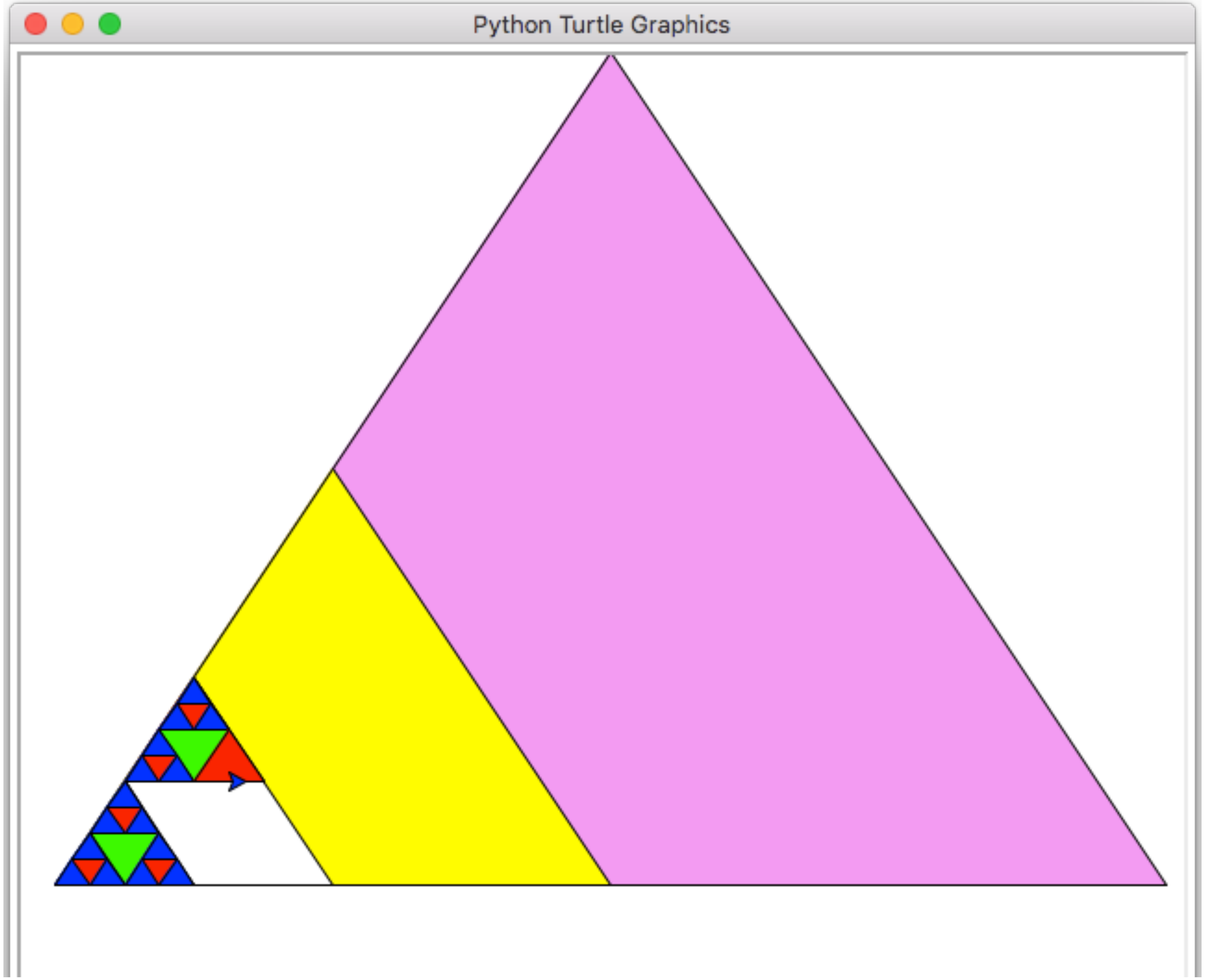
Easy Assignment



Homework Assignment

- Write a working Sierpinski Triangle
- Email it to me
- I have provided a template, sierpinski.py

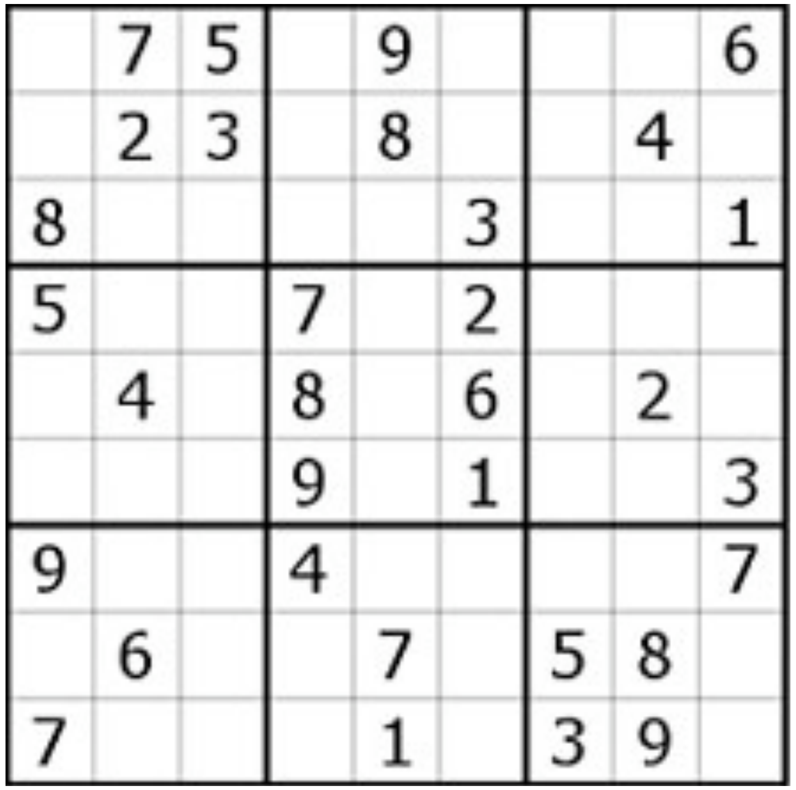
Homework Assignment



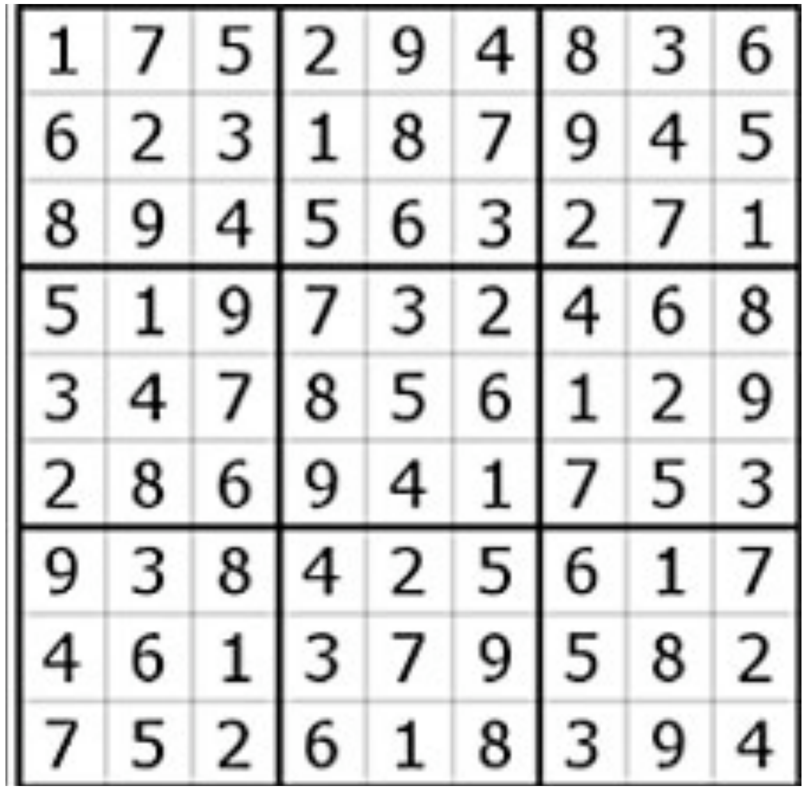
Bonus Assignment

- Implement a function called **sudoku_solve** that:
    - Takes an “empty” Sudoku board and returns the solved board

Bonus Assignment

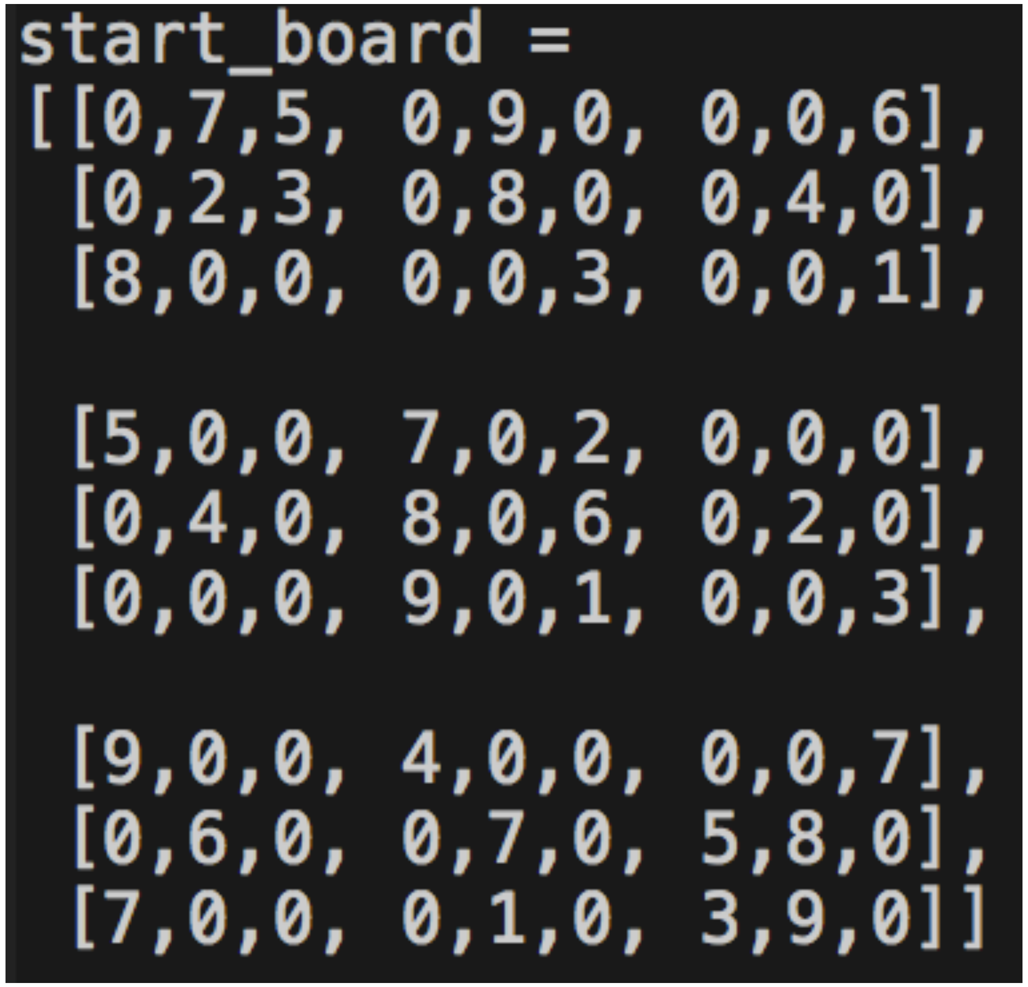


	7	5		9				6
	2	3		8			4	
8					3			1
5			7		2			
	4		8		6		2	
			9		1			3
9			4					7
	6			7		5	8	
7				1		3	9	



1	7	5	2	9	4	8	3	6
6	2	3	1	8	7	9	4	5
8	9	4	5	6	3	2	7	1
5	1	9	7	3	2	4	6	8
3	4	7	8	5	6	1	2	9
2	8	6	9	4	1	7	5	3
9	3	8	4	2	5	6	1	7
4	6	1	3	7	9	5	8	2
7	5	2	6	1	8	3	9	4

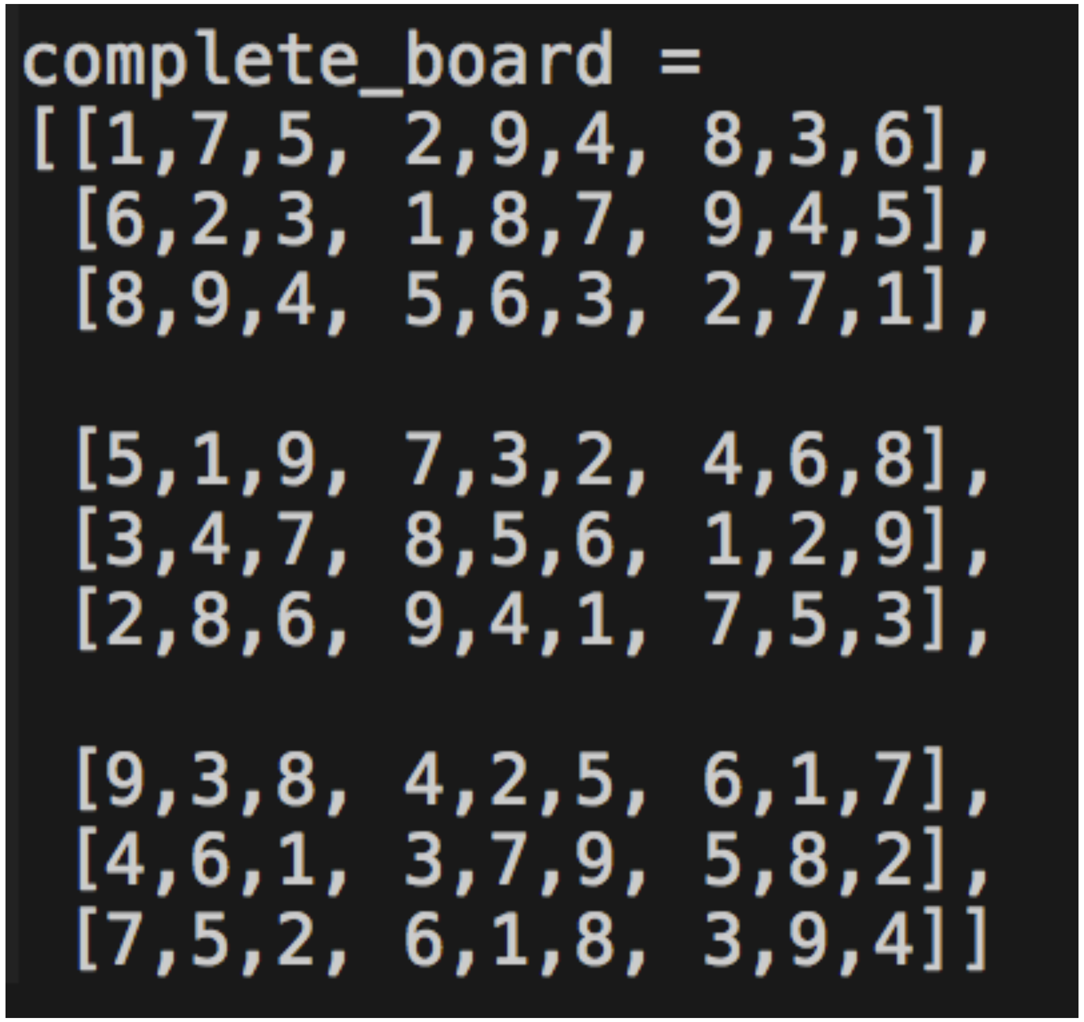
Bonus Assignment



```
start_board =
[[0,7,5, 0,9,0, 0,0,6],
 [0,2,3, 0,8,0, 0,4,0],
 [8,0,0, 0,0,3, 0,0,1],

 [5,0,0, 7,0,2, 0,0,0],
 [0,4,0, 8,0,6, 0,2,0],
 [0,0,0, 9,0,1, 0,0,3],

 [9,0,0, 4,0,0, 0,0,7],
 [0,6,0, 0,7,0, 5,8,0],
 [7,0,0, 0,1,0, 3,9,0]]
```



```
complete_board =
[[1,7,5, 2,9,4, 8,3,6],
 [6,2,3, 1,8,7, 9,4,5],
 [8,9,4, 5,6,3, 2,7,1],

 [5,1,9, 7,3,2, 4,6,8],
 [3,4,7, 8,5,6, 1,2,9],
 [2,8,6, 9,4,1, 7,5,3],

 [9,3,8, 4,2,5, 6,1,7],
 [4,6,1, 3,7,9, 5,8,2],
 [7,5,2, 6,1,8, 3,9,4]]
```

Next Week

Debugging & Testing

(How to reduce bugs)