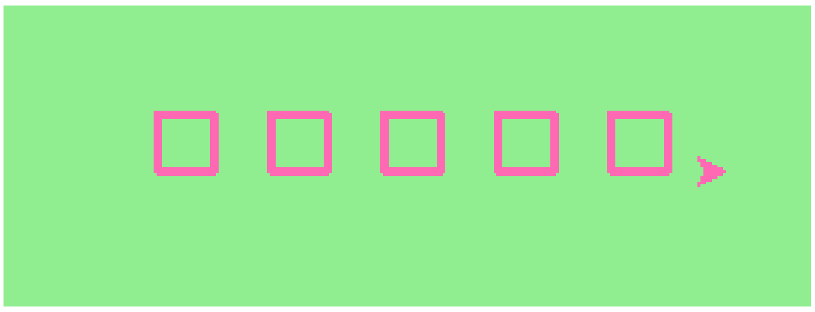
COSC2429 Intro to Programming

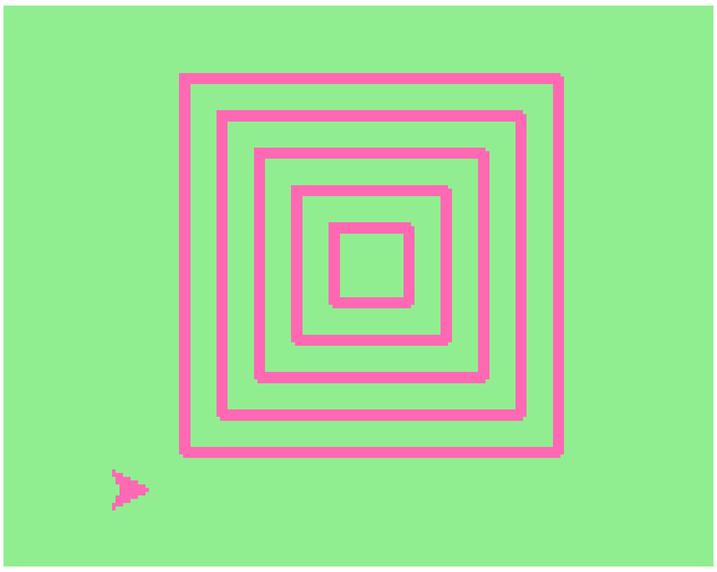
Lab 4: Functions & Iterations

Objective: At the end of this lab, you will be able to apply the divide-and-conquer strategy by creating your own functions to effectively solve a problem.

1. Apply the divide-and-conquer strategy by using the **draw_square** function that we had in the lecture to write a program that draws the image shown below. Assume each side of the square is 40 pixels. Hint: notice that the turtle has already moved away from the ending point of the last square when the program ends.

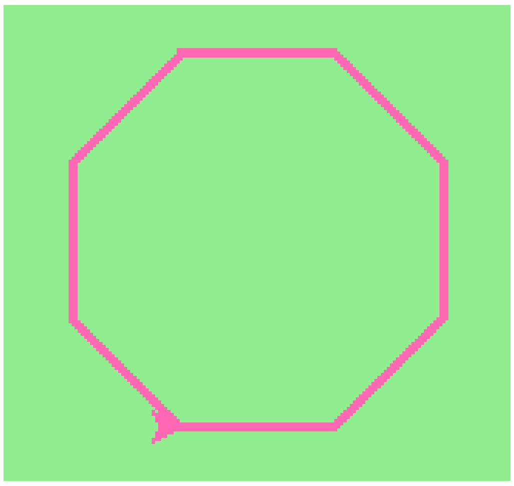


2. Apply the divide-and-conquer strategy to write a program that draws the image shown below. Assume the innermost square is 40 pixels per side, and each successive square is 20 pixels bigger, per side, than the one inside it.

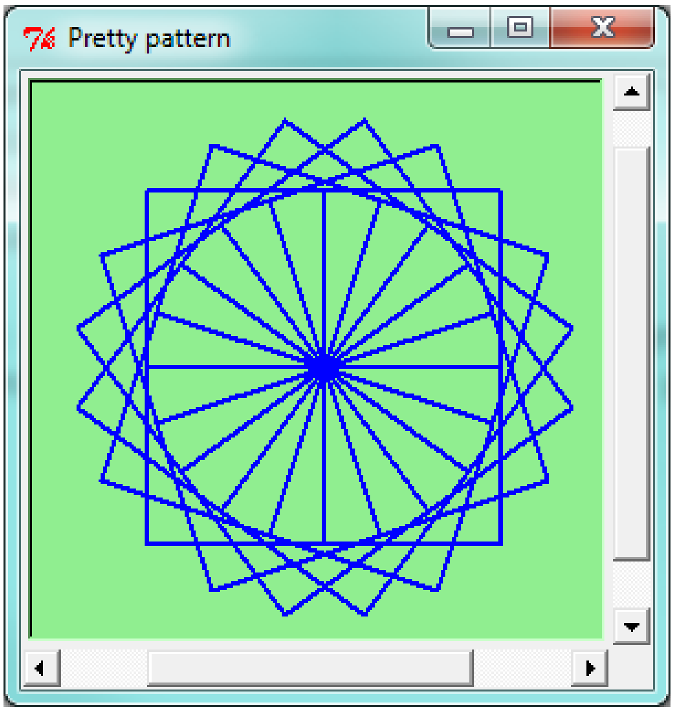


3. Write a non-fruitful function **draw_poly(turtle, sides, length)** which makes a turtle draw a regular polygon (Tip: search the Internet to find out what the angle of a regular polygon is). Then use the above function to write a program that asks the user for the number of sides of a polygon and the length of each side. The program then draws a polygon with those values. This is an example of applying the divide-and-conquer strategy.

   For example, if the user inputs 8 for number of sides and 100 for length, it will draw a shape like this:



4. Apply the divide-and-conquer strategy to write a program that draws this pretty pattern.



5. Write a function **print_triangular_numbers(n**) that prints out the first n triangular numbers (n is a positive integer). A call to **print_triangular_numbers(5)** would produce the following output:

```
1       1

2       3

3       6

4       10

5       15
```

(Hint: search in Google to find out what a triangular number is)

Then apply divide-and-conquer strategy to write a program that asks user for a value of n (n is a positive integer) then calls the function with that value to print out the first n triangular numbers.

6. Apply divide-and-conquer strategy to write a program that asks the user for a positive integer n and then print out a number pattern base on n. For example, when n is 5, the pattern looks like this.

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Hint: print(i, " ", end="") leaves one whitespace after printing i and makes the cursor stay in the same line instead of going to a new line.