Name:_____ **Partners:** _____

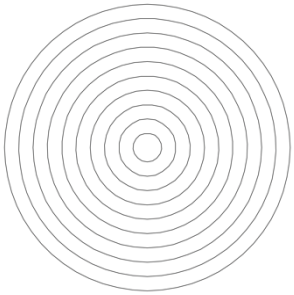
Python Activity 32: Graphical Recursion

Learning Objectives
Students will be able to:
Content:
- Predict what **recursive turtle** code will do
- Define function **invariance**

Process:
- Write code that draws line drawings *recursively*
- Write recursive functions that are **invariant**

Prior Knowledge
- Python concepts: recursion, turtle

Critical Thinking Questions:



1. We want to draw concentric circles recursively, separated by a whitespace `gap`, as follows:
   a. What might be the base case? When should we stop drawing circles? _____
   b. What is the small, repeated step?
   _____
   c. How should we break the journey down into smaller pieces?
   _____
   d. Fill in the partially completed code below with your answers to (a)-(c):

Recursive Code

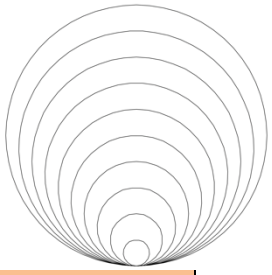
```
from turtle import *
setup(400, 400)

def concentric_circles(radius, gap):
      if base case (a)      :
            return 0
      else:
            # (b) small step:

            # (c) small step on smaller pieces:
            num_circles =


            # we drew one circle here, plus more to come!
            return 1 + num_circles
```

e. This code produces the following output, why might that be?



2. The following code produces our desired output.

a. Circle where the code differs from what we have in Question 1:

Recursive Code

```
from turtle import *
setup(400, 400)

def concentric_circles(radius, gap):
    if radius < gap:
        return 0
    else:
        down() # pen down
        circle(radius) # (b) small step

        up() # pen up, don't want to draw!
        lt(90)
        fd(gap)
        rt(90)

        # (c) small step on smaller pieces:
        num_circles = concentric_circles(radius-gap, gap)

        # we drew one circle here, plus more to come!
        return 1 + num_circles
```

b. What does this new code do?

Why is it necessary?

c. What does this code return? (*Hint: It's different from what is displayed!*)

Highly recommend working through the Application Questions!

Application Questions: Use the Python Interpreter to check your work.

1. We now want to adjust our `concentric_circles(..)` code to create the following designs, filling each circle with an alternating color (in this case, `"purple"` and `"gold"`).

a. First, write a helper function, `draw_disc(radius, color)`, that will draw a circle of the given `radius` and fill `color`. You'll want to assume that the pen is up when the function is called, and to retract the pen when you're done:

```
from turtle import *
setup(400, 400)

def draw_disc(radius, color):
    # put pen down

    # set the color

    # draw the circle with the color



    # pull pen up
```

b. Next, we need to modify `concentric_circles(..)` by adding two parameters to our function: `color_outer` and `color_inner`. Our base case will be the same as before, but we'll need to add something about alternating colors to recursive steps:

   i. How do we add color to the recursive call (i.e., small steps)?

   _____

   ii. How do we ensure the colors *alternate* in the recursive call?

   _____

c. Write out the modified function, making use of our helper function, `draw_disc(..)`, and these two additional steps to handle color:

```
def concentric_circles(radius, gap, color_outer, color_inner):
    if radius < gap:
        return 0
    else:
        # small step




        # reposition



        # small step on smaller pieces:
        # remember to handle alternating colors!
        num_circles =




        # we drew one circle here, plus more to come!
        return 1 + num_circles
```

FYI: When building recursive turtle functions, it is a good idea to ensure they are ***invariant***. That is, the position of the turtle is the *same* at the end of the function as it is at the beginning.

d. In the above code, we change the position of our turtle at the `# reposition` comment. *Place an asterisk* where we might add code to return the turtle to its original position, before the `# reposition`.

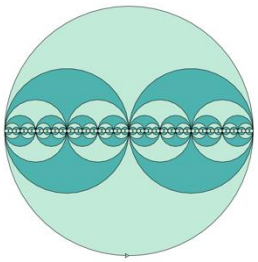
e. Write some lines of code to return the turtle to its original position:
(Hint: We used 3 lines to reposition, so we'll need 3 lines to retrace its steps!)

_____

_____

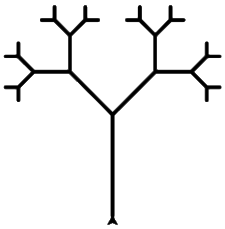
_____

2. Write a recursive turtle function, `nested_circles(radius, min_radius, color_out, color_alt)`, to produce the following visual output with doubly nested recursive turtles. Also, ensure the function returns the number of circles drawn.



- *Hint 1: Follow the steps we've used previously to figure out the pieces of the recursion & base case.*
- *Hint 2: You'll need to have **two** recursive calls as part of your breaking the journey down!*
- *Hint 3: Whereas in the previous example, **invariance** of the function likely would not impact our output, it matters in the case of this example, with multiple recursive calls!*

3. Write a recursive turtle function, `tree(trunk_len, angle, shrink_factor, min_len)`, to produce the following visual output recursively, while also returning the number of branches drawn.



- *Hint 1: Follow the steps we've used previously to figure out the pieces of the recursion & base case.*
- *Hint 2: You'll need to have **two** recursive calls as part of your breaking the journey down!*
- *Hint 3: **Invariance** of the function will likely matter again!*