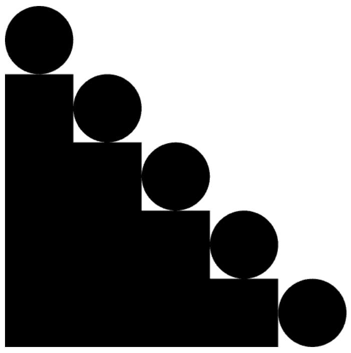
National University of Singapore
School of Computing
CS1101S: Programming Methodology
Semester I, 2024/2025

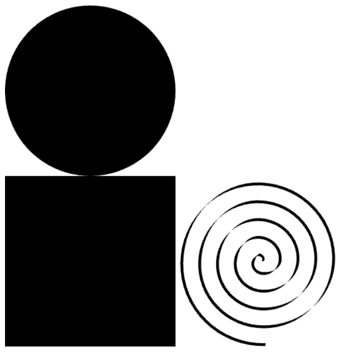
S3-in-class
Recursion; Iterative and Recursive Processes

The goal of this in-class exercise is to write a function `moony` that takes a parameter `n` and produces a rune with `n` circles on the steps of a staircase. The picture below is produced by `show(moony(5))`.



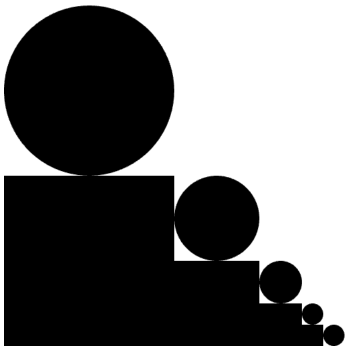
Problems:

1. Begin by writing a function `moony_1` that takes an argument `bottom_right` and produces a rune as shown in this picture.



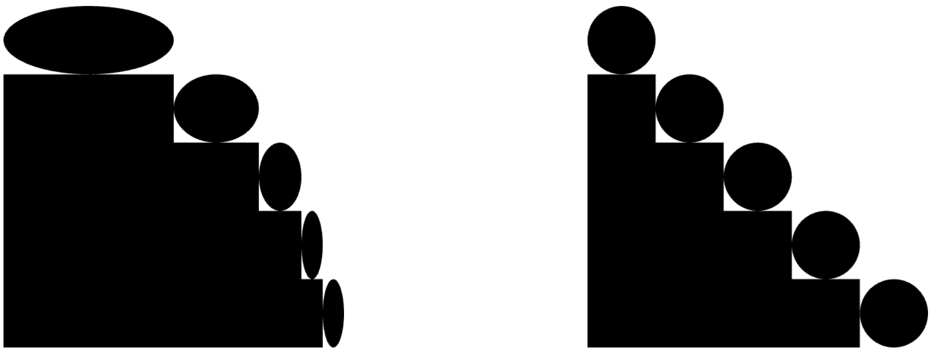
   The picture is the result of `show(moony_1(ribbon))`.

2. Now that we have one circle, we need to produce $n - 1$ more. Use recursion in a function `moony_2` to insert $n - 1$ other circles in the approximately correct location.

   The picture is the result of `show(moony_2(5))`.

3. Use the available primitive combinations on runes to even out the rows and columns, one axis at a time. You may call your final version `moony`.

   The picture on the left show the result of evening out the rows but not yet the columns, and the picture on the right shows the result of `show(moony(5))`.

   Can you explain how your `moony` manages to even out both rows and columns?

4. Do your solutions give rise to recursive or iterative processes?

   Characterize the resource consumption of your function `moony` using the orders of growth notation introduced in Lecture L2B.

   In your description, be clear about what you consider the “size” of the given problem.

   What assumptions are you making on the resource consumption of primitive runes and of primitive operations on runes?