Lab: Pair Programming with Graphics, Functions, and Local Variables

Pair programming is a technique where two programmers work at a single computer to write code. Person A is called the *driver*, and is the person responsible for typing the code. Person B is called the *navigator*, and is responsible for reviewing each line of code as it is typed in, looking for syntax errors, other bugs, or ways to improve the code to make it clearer, simpler, or more efficient. Before anything happens, both people should agree on the general structure of the algorithm they're going to use. The two people work together as equals, and switch roles frequently. The goal of this is to have one person (the driver) who focuses all their attention on the role of writing the code, while the other person (the navigator) focuses on making sure what the driver is writing is the best possible code they could write.

- Create a program to draw a bullseye (looks like the Target logo) on the screen.
    - First create a `main()` function that opens a canvas, draws one circle, and closes the canvas on a click. (Do this just to make sure your graphics are working correctly.)
    - Modify `main()` so it draws at least four concentric circles using the `draw_filled_circle` function.
    - Alternate between two colors to get a bullseye effect.

- Modify your program to add a function called "`draw_bullseye`" that takes two parameters: the x- and y-coordinates of the center of the bullseye. Add code to the body of `draw_bullseye` so that when called, it draws a bullseye at the (x, y) location specified by the parameters. Then, edit your `main()` function to call `draw_bullseye` twice at different locations on the screen (pick two different (x, y) locations).

  Hints: Your function definition line will look like this → `def draw_bullseye(x, y):`

  When you're done, there should not be any circle-drawing function calls inside `main` anymore, they should only be inside of `draw_bullseye`. All `main` should do is open a canvas, call `draw_bullseye` twice, and close the canvas on a click.

- Modify your program to add a third and fourth argument to your bullseye function called `color1` and `color2`. These arguments will let the caller of the bullseye function choose two alternating colors for the bullseye.

  For instance, if the user wanted a bullseye centered at (100, 100) colored red and black, they should be able to call the function like this: `draw_bullseye(100, 100, "red", "black")`

- Modify your main function so the user can type in the (x, y) coordinates of the center of a bullseye they want to draw, and the colors they want to paint it.
    - This will require four separate input statements: one for x, one for y, and two for the colors.
    - ***The input statements should not go inside the bullseye function; they should be inside the `main` function, and the information typed in should be passed as arguments to the bullseye function***.

- Modify your bullseye function definition to take a fifth argument, the radius of the bullseye. You'll need to do some math to figure out what the radii of the nested circles should be.

- **Challenge**: Write a function called `draw_square(x, y, side)` that takes as arguments the (x, y) coordinates of the center of a square and the length of a side. You should use `draw_line`, `draw_polyline`, or `draw_rect`. Remember: x and y should be the center of the square, not a corner.

- **Challenge**: Modify your program so the user can choose the center of the bullseye using a mouse click. Refer to the graphics library handout.

- **Challenge**: Modify your program so the user can choose the center and radius of the bullseye with two mouse clicks (first click chooses the center, second click chooses the a point on the border of the bullseye from which you can compute the radius). You'll need to use the distance formula for this one:

  The distance from point (x_1, y_1) to (x_2, y_2) is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. To use the square root function in Python, put `from math import *` at the top of your program, then you can use the function `sqrt()`.