Programming Problems

For each of these problems (unless otherwise specified), write the needed code directly in the Python file, under the comment and print statement that correspond to the problem. Do not delete the provided print statements- we're using them to autograde.

If you find yourself struggling to get your code to work, remember your resources! Office hours in particular are useful for debugging problems.

Before you submit: click 'Run File as Script' to make sure your code runs without raising an error message. Any syntax or runtime errors left in the code will result in a deduction on the assignment grade. You should do this for all future programming assignments as well.

#1 - Printing - 10pts

Can attempt after Programming Basics lecture

Write code at the top level of the file to match the following algorithm.

1. Assign either the string `Kelly` or the string `Franceska` to **`prof`**.
2. Assign a string holding your 110 TA's name to **`ta`**. If you have two TAs, choose one.
3. Write a single print statement that greets both your professor and your TA by name. The statement must use the variables **`prof`** and **`ta`**, as well as at least one additional string.

#2 - Using Function Calls - 10pts

Can attempt after Function Calls lecture

Write code at the top level of the file to match the following algorithm:

1. Import the **`random`** library and the **`math`** library
2. Generate a random integer between [1, 360] and store it in the variable **`x`**.
3. Convert the integer value in **`x`** to a radian number with a function in the math library and store the result in the variable **`r`**. *Hint*: use radians function.
4. Write a single print statement that describes the relationship between the values in **`x`** and **`r`**. The print statement must use both variables, as well as at least one additional string.

#3 - Libraries (Graphics) - 10pts

Can attempt after Function Calls lecture

Add code under the comment `# draw fence here` so that it draws a simple version of the CMU Fence on the canvas. If you have not yet heard of the Fence, learn more here: www.amusingplanet.com/2014/09/the-fence-of-carnegie-mellon-university.html

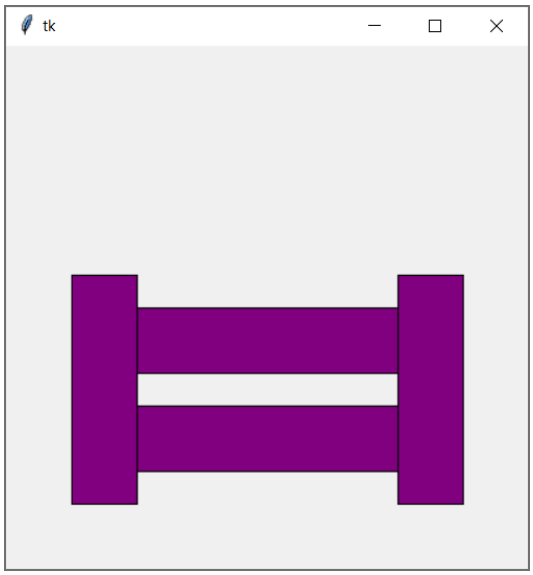
Note: because graphics require set-up code, we have included graphics setup code in the file for you. Do not delete this code!

Your Fence must meet the following basic requirements, but otherwise you may customize it as much as you like.

- The Fence must have at least two columns, with one column ending the left side of the Fence and one ending the right
- The Fence must have two cross-boards that connect all the columns
- There must be a gap above the top cross-board, between the cross-boards, and below the bottom cross-board
- The Fence must be painted at least one color

Otherwise, you're encouraged to 'paint' your fence with your own design or message! You can find bonus instructions on graphics here: www.cs.cmu.edu/~110/slides/week2-graphics.pdf

Here's an example of a simple Fence that meets the requirements:



#4 - Defining Functions - 15pts

Can attempt after Function Definitions lecture

Write a function that implements the following algorithm, which finds the slope of a line between two points.

- The function's name is **slope**
- The function takes four arguments: **`x1`**, **`y1`**, **`x2`**, and **`y2`**.
- The function should follow this algorithm:
    a. First, compute the difference in y values (using subtraction) and assign it to the variable **`diffY`**.
    b. Second, compute the difference in x values and assign it to the variable **`diffX`**.
    c. Third, compute the slope (**`diffY`** divided by **`diffX`**) and assign it to the variable **`m`**.
- The function should return the variable **`m`**.

After the function definition, write a print call that directly calls slope on four different numbers and displays the result.