CSC 110, Spring 2018
Programming Assignment #3: Progress (16 points)
Due: Tuesday, February 6, 2018, 7:00 PM

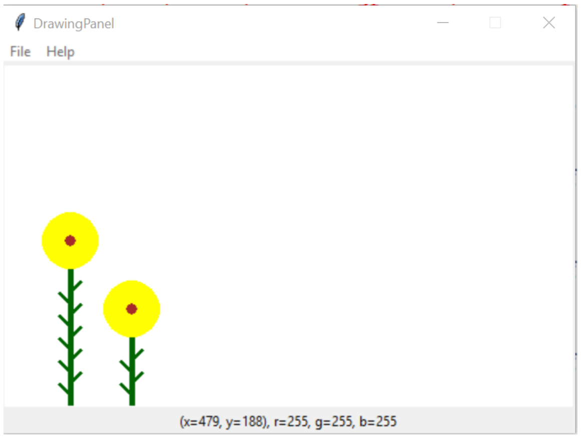
Program Description:

This assignment covers parameters and graphics. Turn in a Python file named `progress.py`.

To run this assignment, you must download the file `DrawingPanel.py` from the Homework section of the class web page and save it in the same folder as your code. Do not turn in `DrawingPanel.py`.

Many programmers who specialize in the subfield of Human Computer Interaction (HCI) work on designs to help users visualize progress and stay motivated. One really common form is flower(s) or tree(s) that grow as the user completes a desired activity. You may have seen these in fitness apps, like the Fitbit on the left, cars, and other places.

This week you will write a program that asks the user for a number of steps and then outputs flowers whose height and number depends on that number. The more a user has walked, the nicer their flower garden will be. Your program should produce an image similar to the one on the right.



The overall `DrawingPanel` should be size **500 x 300**. If there are no visible differences to the naked eye between the expected output and your output, we will consider it correct.

Background:

You may draw any design you like to appear behind your flowers. It must contain at least **five** shapes, use at least **three** distinct colors, **two** functions or loops, be your own work, and not be similar to the flowers you will draw in the foreground.

Required Behavior:

```
How many steps have you walked? 7809
```

When your program starts it should prompt the user for the number of steps they have walked as shown above. You can assume that the user will input an integer value greater than 0.

Once the user has typed in their steps, your program should open a `DrawingPanel` with your background displayed on it. It should then draw flowers on the `DrawingPanel` based on the number the user input. Your program should draw one flower with 10 leaves for every 5000 steps a user has walked. If a user has walked a number of steps that has a remainder when divided by 5000 then the program should draw one more flower. This flower should have two leaves for every 1000 additional steps the user has walked. Therefore, if the user walked 7809 steps, the program should draw one flower with 10 leaves and one flower with 4 leaves.

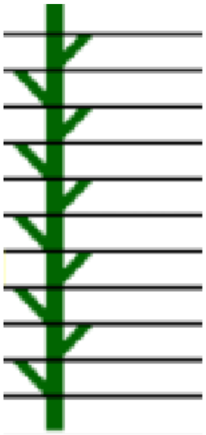
Since each full flower represents 5000 steps, you can figure out the number of full flowers to draw by dividing the total number of steps by 5000. Once you have drawn that number of whole flowers, you can draw one last partial flower representing the remainder of the total number of steps when divided by 5000. For example, if a user walked 1342 steps, you should compute `1342 // 5000` to find the total number of full flowers to draw and `1342 % 5000` to find out how many steps are left over and need to be taken into account when drawing the partial flower.

Flowers should all start at the bottom of the screen. They should be placed at random `x` locations. Flowers should never appear partially off the screen; make sure you choose random locations where the whole flower can fit on the screen.

The stalk should grow 20 pixel taller every time two more leaves are added. Both the stalk and leaves are lines that have a stroke of 5 pixels. The leaves are positioned at intervals of 10 on the stem. Their x starts at the stem and ends 10 pixels away from it. Their y also ends 10 pixels from where it started. See the diagram at right for more details.

The flower petals are a single 50 x 50 oval centered above the stem. The eye of the flower is a 10 x 10 circle centered inside the petal circle.

You may choose any colors you like for the flowers.



10px between each black line and at the top and bottom.

Note that the next leaf starts where the last stops.

Implementation Guidelines:

To receive full credit, you are required to have a particular function described below. This function uses parameter passing and performs some of the program's complex computations.

Function to draw a flower

Your function should draw a single flower. Notice that we specify each flower in terms of its location and how many leaves it has. Different flowers have different positions, and numbers of leaves. Therefore, your function should accept several parameters so that it is possible to call it many times to draw the many different flowers potentially on the screen.

You should also create other functions to capture structure and redundancy. It is up to you to figure out which other functions are necessary.

Place the following statement at the top of your Python file, so that your code can use graphics:

```
from DrawingPanel import *    # so that I can use Graphics
```

Development Strategy (How to Get Started):

This program does not require very many lines of code. However, the numeric computations and parameters are not simple. You might be overwhelmed with the amount of detail you have to handle all at once. As famous computer scientist Brian Kernighan once said, "Controlling complexity is the essence of computer programming." To make things easier, begin with a smaller piece of the problem.

Write your code in stages, repeatedly making small improvements. Start by having your first function draw only one flower of a set height at a set location, then generalize it by **adding one parameter at a time**. For example, add a parameter to change the x position. Test the parameter by passing different values. Once it works, move on to the next.

Style Guidelines:

For this assignment you are limited to the language features discussed in lectures 1 - 10.

We require at least the function described previously. You should use additional functions when they would improve your code. You may receive a deduction if your functions accept too many parameters or unnecessary parameters. An "unnecessary" parameter in this case is one whose value is redundant with another's, could be computed using others' values or simply isn’t used.

Give meaningful names to functions, variables, and parameters. Follow Python’s naming standards as specified in lecture. Limit the lengths of your lines to fewer than 80 characters. Include meaningful comment headers at the top of your program and at the start of each function.