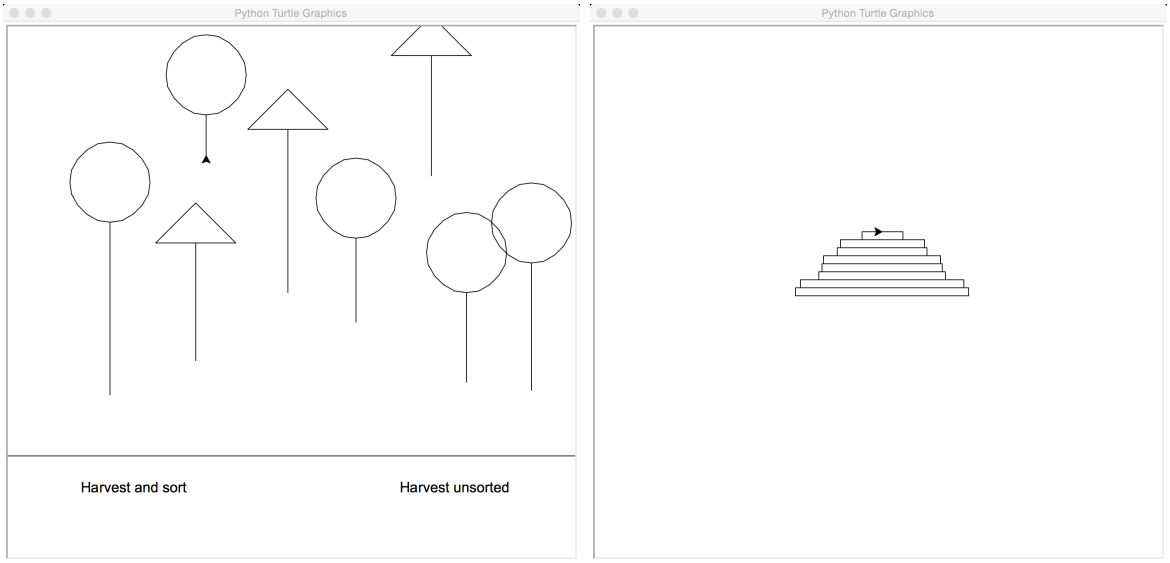
Computational Problem Solving CSCI-603
Cutting Down Trees Lab 2

1 Problem

In this week's assignment, we will first draw a number of trees of different "species" and then harvest them all, drawing the appropriate pieces of lumber and computing the total harvest.

2 Problem Solving Session

Consider the drawing of a forest as shown in the left hand image below. The tree on the far left is our representation of a maple tree and the one to its right is supposed to be a pine tree.



1. Write code for drawing the maple tree. In this assignment, each tree will be drawn as a response to user input at any location within the window. Given that, make sure to specify appropriate preconditions and postconditions.
2. When drawing a pine tree, or indeed some other kind of tree, what pieces of the maple tree drawing function can be reused? Write a general draw tree function that can draw a maple tree as well as some other types, but do not write the code that differs for the other types (just a note as to where it will go is fine).
3. In the right-hand picture above, we can see a pile of logs, nicely centered on top of one another. What would be the preconditions and postconditions of the function that draws a single log? How would you draw a log of 10 x 200 pixels?

At the end of problem-solving, put all group members' names on the sheet, number each item and hand in your work, one copy per team.

3 Implementation

You are strongly encouraged to work in teams of *two* for this lab.

Your solution will consist of several components. In the first phase of operation, you should draw two "buttons" (just pieces of text) as shown in the image above, along with a dividing line. The window above the dividing line should remain empty until the user clicks on it.

When the user clicks above the dividing line, draw a tree starting from that location. To react to user clicks, write a function `doclick` that accepts two arguments, `x` and `y`. Then tell the window that it should call your function when a click happens by calling `turtle.onscreenclick(fun=doclick)` (note that your function actually can be called anything as long as you pass that name to `onscreenclick`). When your `doclick` function is called, the values of `x` and `y` will be the location of the click in window coordinates.

The tree that will be drawn should be of one of three types, chosen at random each time. Two of the types should be the maple and pine shown above (any design close to the pictured ones is fine), while the other should be another tree of your design. The trunk should be of random height between 50 and 250 pixels, except that the top of the trunk should never be less than 50 pixels from the top of the window.

As each tree is drawn, you should keep track of it so that you know how large a log it will make after harvest. Here we only use the trunk length as the log for each tree. We have provided you a class called `LumberYard` in the file `yard.py` which you may download from the "Content" area in MyCourses. This class has a function `addLog(log)` that will add a log length to an internal list for retrieval later.

When the user clicks anywhere below the dividing line, it is time to harvest the trees. If they click on the left side of the screen, you should draw all the logs in a nice neat sorted pile, as shown in the image. If they click on the right side, you should draw the logs in the order that their trees were drawn. In either case, you can use the `allLogs()` function of the `LumberYard` object to get back the list of all log lengths. Each log is 10 pixels high and as long as the trunk of the tree from which it came. You should also print to the console the total length of all logs put together.

Finally, once the wood pile has been drawn, the next click should simply close the window. If you call `turtle.exitonclick` then this will override any other click responses previously specified.

3.1 Useful tips

- The function `randint` in Python's `random` module takes two arguments and returns a random integer between the values of the two arguments, inclusive (so, `randint(3,5)` could return 3, 4 or 5.).
- Python's `sorted` function takes in a list and returns a sorted version of that list (assuming the items in the list can be sorted) from smallest to largest. You can

also call it with an extra argument `reverse=True` to have it sort from largest to smallest.

- You should avoid using any "magic numbers" in your code, but instead use named constants for things like the location of the dividing line.
- You may use the `turtle.goto` function once when responding to each click to position the turtle, but not otherwise.
- You will also need to create a `LumberYard` object outside of any functions. However, you do not need to (and must not, for full credit) use the `global` statement at any time. In order to do this, you should first `import yard` and then call the function `yard.LumberYard()` outside any of your functions — this function will return the `LumberYard` object.
- For full credit, you should reuse code (whether whole functions or pieces of functions) wherever reasonably possible.

3.2 Grading

The assignment grade is based on these factors:

- 20%: attendance at problem-solving and results of problem-solving teamwork
- 30%: three types of trees are drawn at the correct locations
- 15%: correct handling of clicks in the window
- 20%: lumber stacks are drawn correctly and total length printed
- 10%: The design promotes code reuse. This includes, but is not limited to
  - defining and reusing functions (See problem-solving session work.)
  - not using any global variables besides the `LumberYard` instance
- 5%: The code follows the style guidelines on the course web site.

Note: The submitted file must be a **Python 3 program**.

3.3 Submission

Transfer your program to the CS machines. **Each team of two should submit from only one account**, but make sure to have both names in the code to get credit!

Submit your program before the deadline using try:

```
try grd-603 lab2-1 lumber.py
```

You do not need to submit `yard.py` — in fact, you should not modify this file at all, since we will test your code against the provided version.