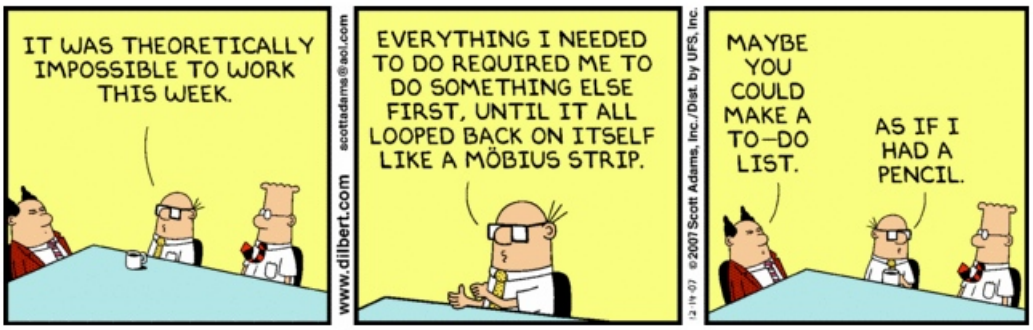
CS150 - Assignment 10

"*To understand recursion, you must understand recursion*"

Due: Wednesday April 30, at the beginning of class



http://dilbert.com/strips/comic/2007-12-14/

For this assignment, we will be playing with recursion. When you're done, make sure you've met all of the specifications required and that you're submitting everything you need to be (two .py files and one .png file).

1 Warming up: lists

Having now discovered recursion, you've decided to become a recursion purist and not use any functions that use iteration/loops. To get you started, you're going to implement some of the list functions recursively. In a file called `"warmup.py"`, write the following functions using recursion:

- A function called `length` that takes a list as a parameter and returns the length of the list. Your function may not use the `len` function (Hint: to see if a list has a length of 0, check to see if it equals `[]`)
- A function called `rec_max` that takes a list as a parameter and returns the largest value in the list. You may assume that the list has at least one entry in it (and, of course, you may not use the `max` function). You can use the `len` function if you'd like.

I've tried to make it clear above what functions are off limits, but please try not to utilize extraneous functions that get around the heart of the problem. If there is any question about what is acceptable, feel free to ask me.

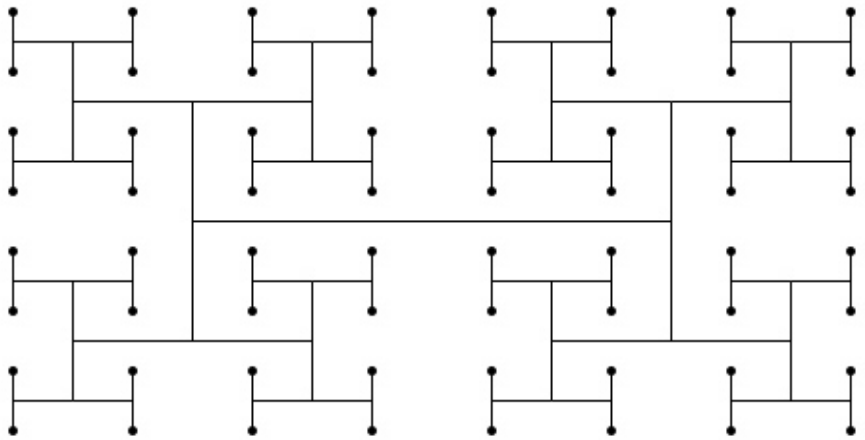
Just for fun

If you want some more practice (*not required*), you can also try this one:

- A function called `rec_count` that takes a list and another object as a parameter. The function returns how many times the object occurs in the list (you may not use the `count`, `in` or `find` functions).

2 Big H

In another file called "`recursive_pictures.py`", write a function called `recursive_H` that takes four parameters: the x and y coordinates of the center of the H, the length of the *vertical bars* (the horizontal bar will be 2 times the length) and number of recursive levels to draw. Level 1 should just draw an H with four dots at the end, level 2 an H with 4 smaller H's at the end of the vertical bars, etc. The smallest H's will all have black dots at the end of their vertical bars. The following is a level 3 recursive H:



See the lab prep for more details.

Advice:

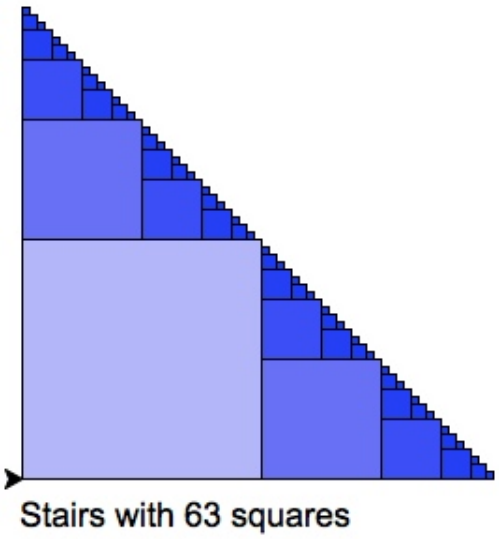
- Walk through the four steps for defining a recursive function we discussed in class. When doing this for graphical components, the key is often trying to finish a statement like: "A recursive H is ...", where somewhere after "is" you should end up using the term "recursive H". For example, a broccoli is a line with three smaller broccolis off of the end of this line. Once you have this, think about what the base case would be.
- It can sometimes help to mentally walk through your recursive steps and draw out the figure by hand.

- As always, think about how you might break off some of the functionality of this function into other functions to make your code easier to debug and read.

http://nrrdgrrl.net/nonsense/category/smellanie/page/2/

3 Stairs

For the last part of this lab add functionality to your "`recursive_pictures.py`" so that *when you run the program/module* (but not when you import it) it generates the following picture:



Stairs with 63 squares

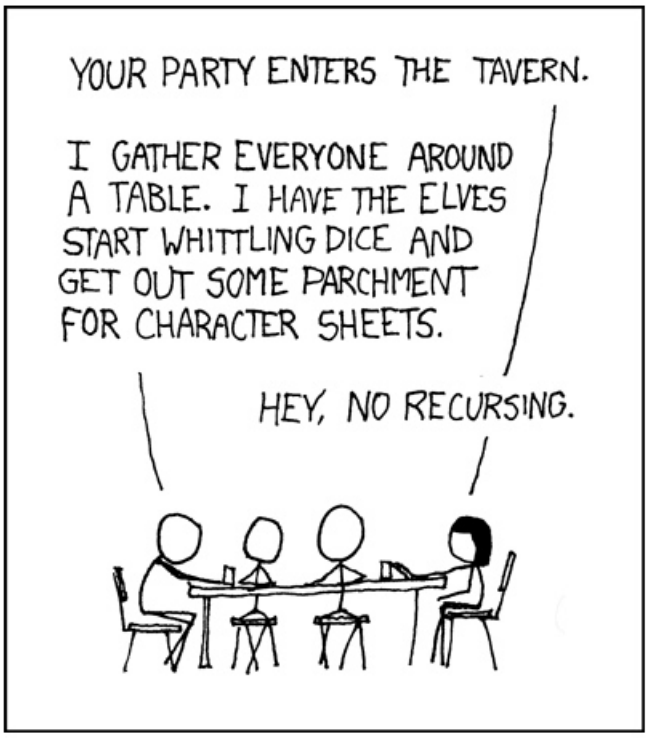
To do this, you must implement a function called `stairs` that takes three parameters: the x and y coordinates of the bottom left corner of the stairs and the *length* of the side of the square in the bottom left hand corner. You should recursively draw stairs as long as the side length is greater than 3. The squares will be colored such that they get darker for smaller squares (not grey). Besides drawing the stairs, the function should also return the number of squares inside the stairs and must be calculated recursively (like we did with counting the lines for broccoli).

Powers of 2 work well for starting lengths. See the lab prep for more details on how the stairs can be viewed recursively.

Implementation:

- Walk through the four steps for defining a recursive function outlined in class. Try and finish the following statement recursively "Stairs are ...".
- Get your `stairs` function working drawing just the outline.
- Add functionality to change the color of the squares based on the length of the side of the square.
- Change your `stairs` function so that it returns the number of squares that make up the stairs.
- Write some code so that it generates the stairs when the program runs (I used a starting length of 128).
- Add the text underneath your picture stating the number of squares. Read the documentation in the turtle module (or use `help`) regarding the `write` function to help you with this. You *must* use the answer returned by your `stairs` function in your text (that is, don't hard-code 63 in your call to `write`).

When you're done, create a screen capture of the output of running your program and submit this file along with your .py files (see Lab 2 for instructions on how to do a screen capture if you've forgotten).



http://xkcd.com/244/

4 Extra credit

You may earn up to 2 points of extra credit on this assignment. Below are some ideas, but you may incorporate your own if you'd like. Make sure to document your extra credit additions in comments at the top of the file.

- Add some color to the recursive H
- Make the color more interesting for the stairs, for example, have it change based on whether it's the top or the right recursive staircase.
- Add another recursive function that draws a different recursive structure (for example out of circles or triangles).

5 When you're done

Make sure that your program is properly commented:

- You should have a docstring comment for each module at the top of the file

- You should have comments after the module doc string stating your name, course (including section number), assignment number and the date.
- Each function should have an appropriate *docstring*
- Other miscellaneous comments to make things clear

In addition, make sure that you've used good ***style***

Submission procedure

For this assignment, you will have two .py files and an image file to submit. To submit multiple files, create a directory with your name followed by the lab number. For example, my folder would be called `davidkauchak10`.

Put all the files inside this directory and the create a .zip file from this directory. On the Macs, right-click on the directory and select "Compress...". If you're working on Windows, right-click on the file and select "Send to" then select "Compressed (zipped) Folder" (or if you don't have this option, use http://www.winzip.com/). You will then see a file with a .zip extension be created.

Submit this .zip file digitally at the usual location:

http://www.cs.middlebury.edu/~dkauchak/classes/cs150/submission/

Enter the relevant information and upload your file. If you have problems with this, please let me know. This link can also be found in the "Resources" section of the course web page at the bottom.

Grading

		points
`warmup.py`		
	`length`	3
	`rec_max`	3
`recursive_pictures.py`		
	recursive H	4
	recursive stairs	4
	returns num squares	2
	stairs coloring	2
	text/generates picture on run	2
Comments, style		5
extra credit		2
total		25 (+2)