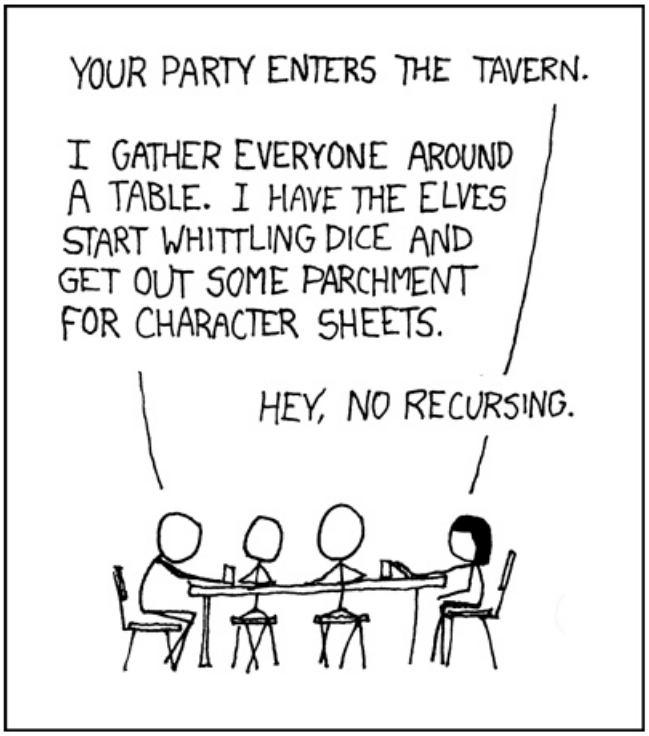
CS51 - Assignment 7
Recursion

Due: April 2, at 11:59pm



https://xkcd.com/244/

For this assignment, we will:

- Become comfortable drawing in Python using the `turtle` module
- Write recursive code

1 The `turtle` module

A crucial programming paradigm is code reuse. Rather than writing your own code, say a `turtle` module, you can use one that someone has already written. An important part of coding, then, is

documentation: both generating it for your programs and understanding other people's documentation.

The documentation for the `turtle` module can be found at:

`https://docs.python.org/3/library/turtle.html`

If you scroll down a bit, you can see an overview of all of the functions available to you.

To get ready for this assignment, you're going to need to read a bit more about some of the functionality of the `turtle` module. Besides looking at the documentation, another way to get information about a function is with the `help` function. The `help` function takes as an argument the name of a function and outputs the docstring (i.e., function description).

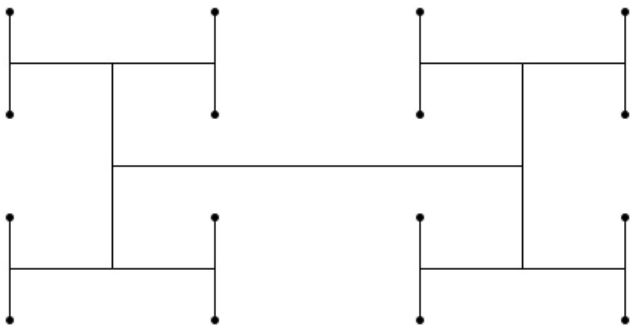
Important: You will need to import the `turtle` module first by typing `from turtle import *` before trying to call help on any of the `turtle` module functions.

2 Planning

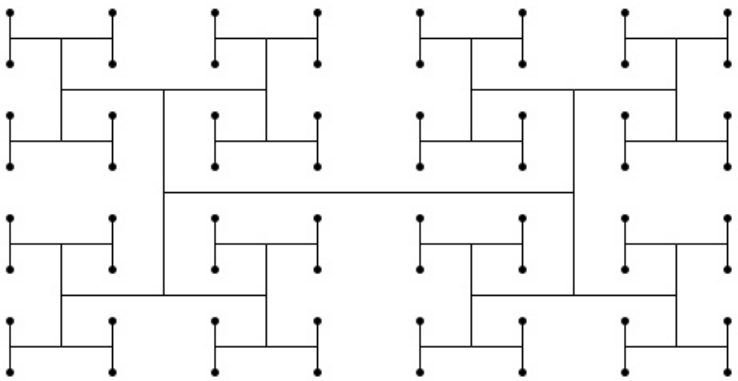
Recursive H Club

A recursive **H** is an **H** where the end of each vertical line of the **H** may have another recursive **H**. The recursive **H** is defined by a level. A level 1 recursive **H** is just an **H** with dots at the ends:

A level 2 recursive **H** has another **H** at the end of each line:



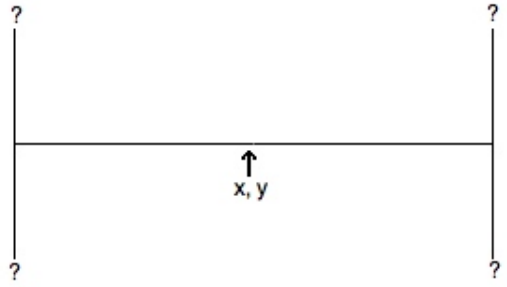
A level 3 recursive **H** has another **H** at the end of the each of these **H**s:



Notice that the smallest level of **H** always ends in a dot.

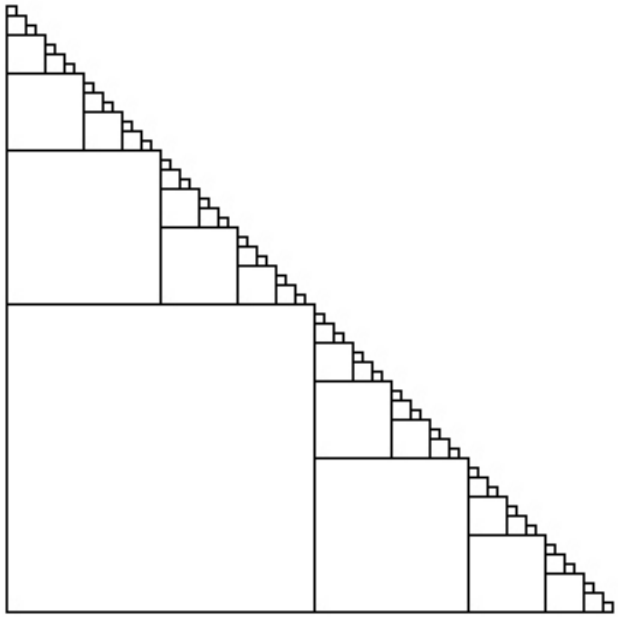
Each of these figures is made up of many **H**s. An **H** is defined by x, y coordinates and a length, l. The x, y coordinates are at the center of the **H**. The *vertical bars* are of length l and the horizontal bar of length $2l$.

Question 1: Given an H at location x, y with length l what are the 4 coordinates of the ends of the vertical bars of that H (that is the upper left, lower left, upper right and lower right of the H) relative to the supplied x, y?

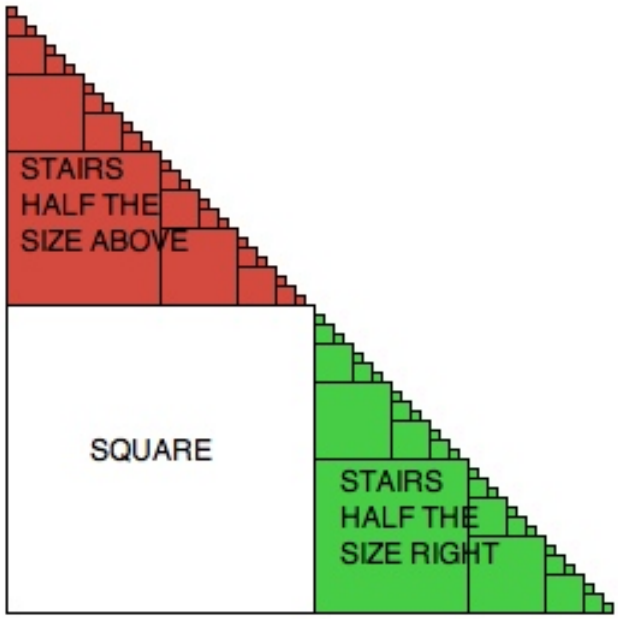


Stairs

We will also be creating a program that draws "stairs":



To draw stairs, we first draw a large square in the lower left. We then draw a set of stairs above it and to the right that are half as large:



Stairs are specified by the x, y coordinates of the bottom left corner of the stairs and the length l of a side of the square in the lower left-hand corner.

Question 2: If we drew stairs starting at x, y with length l, what would be the coordinates of the recursive set of stairs above (colored in red above) and the recursive set

of stairs to the right of the initial square (colored in green above), again relative to x, y?

Color in turtle graphics

We typically specify fill colors in turtle graphics by passing a string to the `fillcolor` function (e.g. `fillcolor('yellow')` or `fillcolor('blue')`). In many situations, this is sufficient; however, in some cases, we need a bit more granularity. Instead of passing a string to `fillcolor`, if you need more precision, you can pass three numbers between 0 and 1.0 specifying the proportion of red, green, and blue that make up the color (called rgb coloring). For example:

```
>>> fillcolor(0, 1, 0)
>>> begin_fill()
>>> circle(50)
>>> end_fill()
```

will draw a green circle. See what happens as you change these three numbers.

Question 3: What would be the rgb values if we wanted a light blue color?

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 built-in functions recursively.

Write the following functions using recursion:

1. A function called `concat_strings` that takes a list of strings as input and returns a string where all of the strings in the list have been concatenated into one with spaces in between the words. For example:

   ```
   >>> concat_strings(['CS51', 'is', 'great'])
   'CS51 is great '
   ```

   Note that my implementation has an extra space at the end. It's fine if this is the case, though it's also fine if yours doesn't.

2. 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 `[]`)

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

```
>>> rec_count([1, 2, 3, 1, 1, 2], 1)
3
```

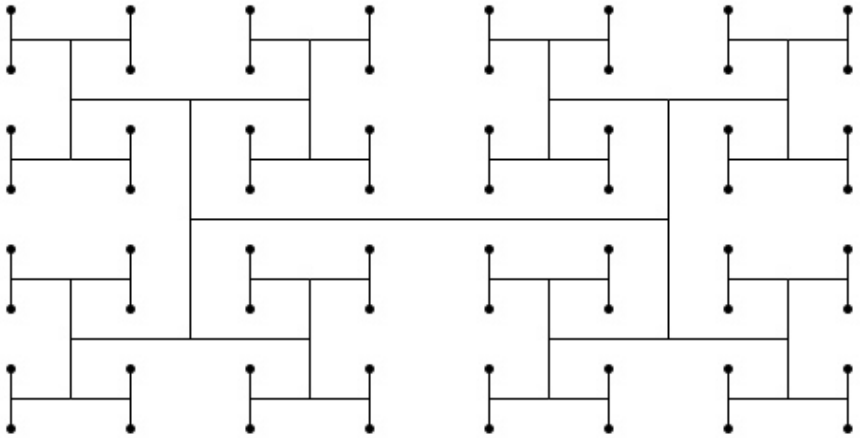
Hint: like the example we saw in class for calculating max, make the recursive call and save that answer into a variable. Then do some calculation and return the overall answer.

4. A function called `remove_spaces` that takes a *string* as input and removes all the spaces from that string. The only string operations you may use are '+' (concatenation) and checking for equality ('=='). For example:

```
>>> remove_spaces("CS51 is great ")
'CS51isgreat'
```

Recursive Turtle

1. 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 the 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:

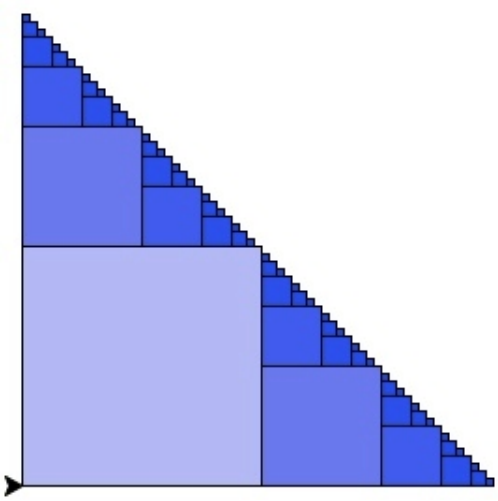


See "Planning" section above for more details.

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 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.
- 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.



2. Write a function called `stairs` that takes *four* parameters: the x and y coordinates of the bottom left corner of the stairs, the *length* of the side of the square in the bottom left hand corner, and a "blueness" parameter indicating how blue the square in the left corner should be. You should recursively draw stairs as long as the side length is greater than 3. The blueness parameter will be a float between 0.0 and 1.0, with larger values indicating lighter blue and lower values darker blue, e.g., 1.0 would draw a white square (no blue) and 0.0 would be dark blue. The squares will be colored such that they get darker for smaller squares (not grey).

   Powers of 2 work well for starting lengths. See the "Planning" section above 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. You can either do this by 1) having the color be based on the length of the side of the square or 2) by adding additional *optional* parameters that allow you to specify the color of the square, which would then change with each recursive call (similar to how the length changes). If you do option 2), make sure that your `stair` function can still be called normally with just 3 arguments.

Extra credit

You may earn up to 1 point 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).

3 Feedback

Create a file named `feedback7.txt` that answers the questions:

- How long did you spend on this assignment?
- Any comments or feedback? What things did you find interesting, challenging, or boring?

4 When you're done

Submit your `assignment7.py`, which should contain your code for the warm up and recursive turtle and, at the top, the answers to the three questions, in comments, and `feedback7.txt` online using Gradescope. Be sure you're uploading them to the right assignment. Note that you can resubmit the same assignment as many times as you would like up until the deadline.

5 Grading

	points
`Planning`	
3 questions	1
Warming up	2
`Recursive Turtle`	
recursive H	3
recursive stairs	3
stairs coloring	1
extra credit	1
total	10 (+1)