1 Administrative Topics

- We take the quiz.

- Either delete the pages or remove the labels from pages you don’t mean to hand in. We use those labels for course organization and it is cluttering things :)

- We have started grading the projects. They look good. But I want to give you a warning about the next project. If you forget to adequately comment your code, then your grade will be penalized by 3%.

2 Rules for Variable and Function Names

A variable name can contain upper case letters (A through Z), lower case letters (a through z), decimal digits (0 through 9), and the underscore character.

A variable name must begin with a letter (upper or lower case).

Example variable names include, A, pic1, pic2, my_string, myString, etc.

Note that this means that the statement “6=A” is illegal in Python.
3 Math Module

There are lots of mathematical functions available in the math module. For example, sin, cos, and other trigonometric functions. Also, there are a few “rounding” functions – floor, ceil, trunc. Finally, there are some interesting one to test whether or not a variable contains an odd value, like infinity or NaN (special symbol for “Not a Number”).

Also,

```python
import math
```

adds an entry in the symbol table with the name “math” and the value is of type `<module>`. The module is a collection of math functions. So we execute them like this:

```python
a = math.sin(0.0)
```

In other words, we provide Python the directions to the function – look up “math” in the main table, then “sin” within the module.

Using this method of importing, it is easy to keep your code organized. But it does require extra writing (i.e. the “math.” before each function). There is another method of importing:

```python
from math import *
```

This tells Python to find a “math” module, open it, and dump all of its contents into the current symbol table. So, instead of having one entry called “math”, we have one entry for each of the functions in the math module. This makes calling math functions more convenient, but makes organization a little tougher because everything is in one table.

Later, you will be writing code that draws things. You may want a variable named “tan” to store info about the color tan. However, the symbol “tan” is already used for the tangent function in math. So, you have two uses for the same symbol - different meanings in different contexts (a color or a math function). But you can have only one value per symbol. The easiest way to resolve that problem is to use the first method of importing. Then, the math function will be called `math.tan` and the color variable will be called `tan`. 
4 Random Module

The random module provides functions that generate pseudo-random numbers. I find this helpful for drawing starry skies and forests.

5 Advice for Project 2

Don’t forget to comment your code. Every function should have a quick comment explaining what it does. For any main code or function with a lot of lines in it, then use comments to explain the different parts of the code. For example, in my tree function, I use a comment to indicate which lines of code draw the trunk and which lines of code draw the leaves.

Simple shapes should have sizes. And it should be possible to draw them at any angle. We talked about a block in class.

Aggregate shapes should have scales. You can always set up your complex shapes so that they are always upright. We talked about a drawing of a house in class.

It is a good idea to design your scenes as if you are a child using simple shapes cut out of colored construction paper. Don’t try to fit perspective into it, unless you really want to.

To figure out what colors are legal, do a web search for rgb.txt. And follow whatever link you want to. The list of colors should be at least partly compatible with your Python installation.

We wrote code to draw a row of blocks, under the assumption that there is a block function already defined and that its parameters are x (x-position of bottom lefthand corner), y (y-position of bottom left-hand corner), and edge (length of edge).

```python
# draw a row of blocks with the bottom lefthand corner at (x,y)
# If scale is 1, then each block will be 200x200.
def rowOfBlocks( x, y, scale ):
    edge = 200
    for i in range(4):
        block( x+i*edge*scale, y, edge*scale )
```

We use a for loop to draw each of the four blocks. There are a few things to
notice in the code

- We use scale to scale the edge length. The “natural” size for a block is 200x200.

- The “loop control variable” $i$ has the value 0 the first time through the loop, 1 the second time through, 2 the third time through, and 3 the fourth time through. So we take advantage of that to determine the x-position of the $i$th block.

- We scale the offset in the x position, but not the x-position itself $(x + i \times \text{edge} \times \text{scale})$. 