Programming Project 05

This assignment is worth 40 points (4.0% of the course grade) and must be **completed and turned in before 11:59 on Monday, February 11, 2013.**

Assignment Overview
This assignment will give you more experience on the use of:

1. functions
2. libraries
3. conditionals
4. iteration

Background
A *tessellation* is created when a shape is repeated over and over again covering a plane without any gaps or overlaps. Another word for a tessellation is a *tiling*. A *regular* tessellation means a tessellation made up of congruent regular polygons. The sides and angles of the polygons must all be equivalent (i.e., the polygon is both equiangular and equilateral) and the polygons that you put together must all be the same size and shape. Only three regular polygons tessellate in the Euclidean plane: triangles, squares and hexagons. [Source: http://mathforum.org/sum95/suzanne/whattess.html] You will use Turtle graphics to draw a regular tessellation of hexagons. (See sample snapshot below.)

Originally written as a part of the *logo* programming language, Turtle graphics (http://en.wikipedia.org/wiki/Turtle_graphics) is a 2D graphics package that uses a Cartesian coordinate system and a "turtle," which you can imagine has a pen attached to its body. The turtle can move around the plane, drawing as it goes. Python has a module that implements the behavior of the original turtle graphics program and this module is simply called "turtle" (see Appendix B of the text and the comments in the sample file `turtleSample.py`).

Project Description / Specification
Your program must meet the following specifications:

1. Print a menu of legal colors and prompt the user to enter two colors (two separate prompts) to be used for coloring the hexagons. Keep re-prompting in case of illegal inputs until a legal color is entered. You can decide what colors to allow, but you must allow at least 3 choices.
2. Prompt the user for the number *N* of hexagons per row, where *N* must be between 4 and 20 inclusive. Keep re-prompting in case of illegal inputs until a legal number is entered.
3. Draw a tessellation that is *N* hexagons wide and *N* hexagons deep, with each hexagon (500/*N*) pixels wide. (So each row is 500 pixels wide, although the whole drawing is a bit wider because the tiling requires shifting successive rows to the right and then the left.) Position it so that the full tessellation fits in the canvas.
4. Alternate the colors of the hexagons in each row.
5. You must define and use at least the following functions in your program.
   a. A function named `get_num_hexagons` to repeatedly prompt for a number of hexagons until the user enters a legal number. Return the number entered (as an `int`). The number must be between 4 and 20 (inclusive) to be legal.
      `get_num_hexagons()`
      - Parameters: None
      - Returns: An `int` in 4, 5, …, 20

b. A function named `get_color_choice` to repeatedly prompt for a legal color until the user enters one. Returns a Tk symbolic name (as a `str`), which Turtle graphics uses to denote colors. See http://www.tcl.tk/man/tcl8.4/TkCmd/colors.htm. For example, `'red'` for red; `'purple'` for purple; `'green'` for green.
`get_color_choice()`
   - Parameters: None
   - Returns: a Tk symbolic name (as a `str`)

c. A function named `draw_hexagon` to draw a hexagon.
`draw_hexagon(x, y, side_len, pen, color)`
   - Parameters:
     `x, y` – coordinates (`float`) of a fixed vertex (e.g., the bottom vertex)
     `side_len` - length (`float`) of each side of the hexagon
     `pen` - "pen" (`turtle`) to draw with
     `color` - Tk symbolic name (`str`) for a color
   - Returns: Nothing

6. Do ***not*** use *global variables* in the bodies of any of your functions. A global variable is a variable that is defined (assigned a value) outside of the function body. Every variable used in the body of a function should be a parameter or defined in the function body (appear on the left side of an assignment statement). This rule distinguishes between variables and constants: a constant is an identifier that is assigned a value once, usually at the start of a program, and is not modified thereafter; whereas, a variable is an identifier whose value might change during execution. To visually distinguish constants from variables, the identifier for a constant is generally all uppercase. A function body may use global constants.

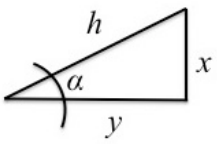
Deliverables

`proj05.py` – your source code solution (*remember to include your section, the date, project number and comments in this file; **do not** include your PID or name*).

1) Be sure to use "`proj05.py`" for the file name (or handin might not accept it!)
2) Save a copy of your file in your CS account disk space (H drive on CSE computers). *This is the only way we can check that you completed the project on time in case you have a problem with handin*.
3) Electronically submit a copy of the file.

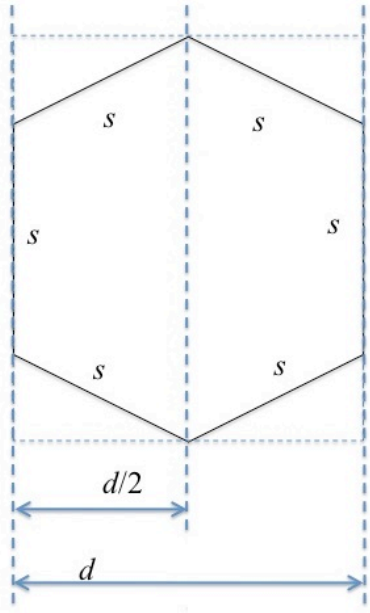
Notes and Hints:

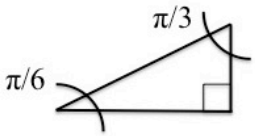
1. You will need to calculate the hexagons' side length. This can be done using elementary geometry and trigonometry. The key trigonometric identities are:



$\sin(\alpha) = x / h$
$\cos(\alpha) = y / h$
$\tan(\alpha) = x / y$

2. The interior angles of a hexagon measure 2π /3 radians (120 degrees), making the angle of incidence at each vertex π /6 radians (30 degrees), as shown in the diagram to the right. Using these measures and a trig equation, you can calculate the side length *s* of a hexagon of width *d* and other information needed to calculate starting coordinates.





3. The Python math module defines:
   a. `math.pi` – the number "pi" (ratio of the circumference and the diameter of a circle)
   b. `math.sin(rad)`, `math.cos(rad)`, and `math.tan(rad)` – the sine, cosine, and tangent functions, where `rad` is the angle measure (`float`) in radians
4. Write your functions first and test each before using them in a main program.
5. Figuring out the colors to use for each tile is non-trivial if you want the effect of diagonal stripes, like you see on many floors that are tiled with hexagons. It is much easier to produce snake-like vertical stripes instead of what you see in our example. Either way satisfies the specifications.
6. Using `turtle.speed(10)` is helpful to draw fast, but we recommend that you begin with slow drawing so you can better observe what is happening. Make your TA happy by including `turtle.speed(10)` in your final version of your program.
7. To keep the canvas from disappearing too quickly, you can import the `time` module and use `time.sleep(sec)` to delay execution of the next statement for (at least) `sec` seconds.

Sample Interaction (user inputs shown in red):

```
>>> ============================ RESTART ============================

Choices for colors to use are:
  red
  blue
  green
  yellow
  orange
  purple
  pink

Please enter your choice: yyellow
'yyellow' is not a legal choice.  Please try again: yllow
'yllow' is not a legal choice.  Please try again: yellow
Please enter your choice: green
Please enter a number of hexagons per row: 3
```

```
It should be between 4 and 20.  Please try again: twelve
It should be between 4 and 20.  Please try again: 12
>>>
```

Snapshot of the drawing produced by this interaction: