Programming Project 02

Assignment Overview
This project focuses on rendering different colors and some simple graphics using Python libraries. You will use both selection (*if*) and repetition (*while, for*) in this project. This assignment is worth 20 points (2.0% of the course grade) and must be **completed and turned in before 11:59 on Monday, September 17, 2012.**

You will use Turtle graphics to draw a picture containing multiple shapes of multiple colors and arranged to be visually pleasing. Although you are free to decide the precise shape(s) and layout for your picture, some aspect of the picture must depend on a numeric value input by the user. For example, the input might determine the size of the shapes, the number of shapes, or their spacing.

Background
Originally written as a part of the *logo* programming language, Turtle graphics (http://en.wikipedia.org/wiki/Turtle_graphics) is one of the oldest graphics programs. It 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:
1. Output a brief descriptive message of what the program does.
2. Repeatedly prompt for the input until the user supplies values of the correct form (discard incorrect inputs). Your prompt should say what form of input is needed.
3. Draw a picture containing multiple shapes of multiple colors, where the input value(s) is (are) used to determine some aspect of the shapes and/or their layout.

In programming your solution, you must:
1. Use at least two repetition (*while* or *for*) statements.
2. Use at least one selection (*if*) statement.

We show example output produced by two different programs that meet these requirements at the end of this write-up. You may be creative and create your own program, or you may choose to mimic one of these two examples. The second example shows error checking being tested.

Deliverables
`proj02.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 "`proj02.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 using handin.

Notes and Hints:

1. Do error checking on input last, i.e. after the rest of the program is working.
2. Class example #9 is a useful model for checking input.
   `http://www.cse.msu.edu/~cse231/Examples/CoursePack/Python/09-inputCheck/`

3. There is a method to check if a string is a number: `isdigit()`
   It returns `True`, if the string is made up of only digits.
   You can use it like this:

```
num_str = input("Enter a number: ")
if num_str.isdigit():
    num_int = int(num_str)
```

Creating Colors:
There are many ways to create a color but a common one used in computer graphics is the process of additive color (see http://en.wikipedia.org/wiki/Additive_color). Combining different amounts of red, green and blue can create most (but not all) colors. In turtle, you can specify a color by giving three floating-point values, each in the range from 0.0 to 1.0, indicating the amount (fraction or percent) of each color. For instance, (1.0, 0.0, 0.0) is red, (0.0, 1.0, 0.0) is green, and (0.5, 0.5, 0.0) is brown. You can find the codes for many colors on a color chart (e.g, "% code" column on http://www.december.com/html/spec/colorcodes.html).

A convenient way to generate different colors is to repeatedly call the random function in the random module to obtain values for the color amounts. First, import the random module: `import random` Then, each call to `random.random()` returns a pseudo random (floating-point) number in the range 0.0 to 1.0. A sample program using this method to create a color and draw a figure is provided in the project directory: `turtleSample.py`

Using turtle graphics:
In order to use turtle graphics in Python you must first import the turtle module. You can then use the help function in Idle to find out what methods this module includes and what they do. Just type `import turtle` in the Python Shell window, hit `enter`, and then type `help(turtle)` and scroll up through the list and information. For more details Google "Python 3 turtle." A sample Python program, `turtleSample.py`, is provided in the project directory. The comments in this file describe methods that you might want to use for this project. When running your program in Idle, you may need to look under the other Idle windows to find the turtle drawing window.

Keeping the window up
If the drawing window has a tendency to disappear too quickly, you can "hold" the window by using the sleep function in the time module, as follows:

```
import time
time.sleep(seconds)
```

The program will wait the number of seconds indicated before it ends.

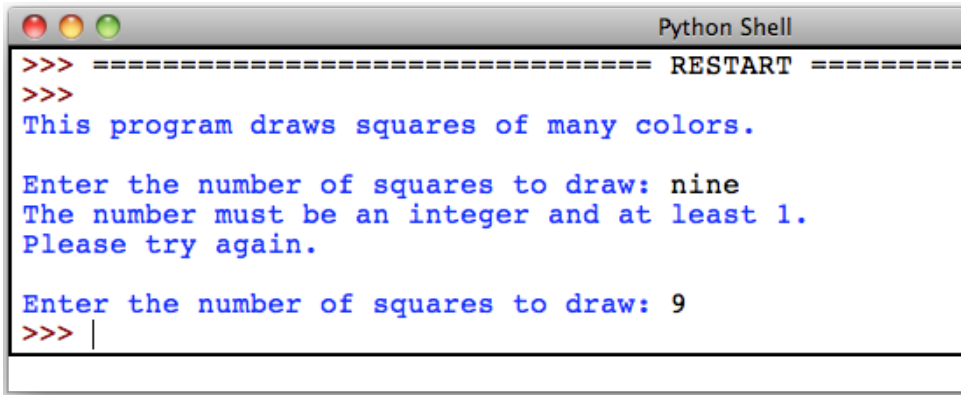
Working nicely with IDLE
Some versions of IDLE do not work well with other graphics windows. If you have trouble with windows hanging (freezing), you can use the _exit function in the os module.

```
import os
os._exit(1)
```

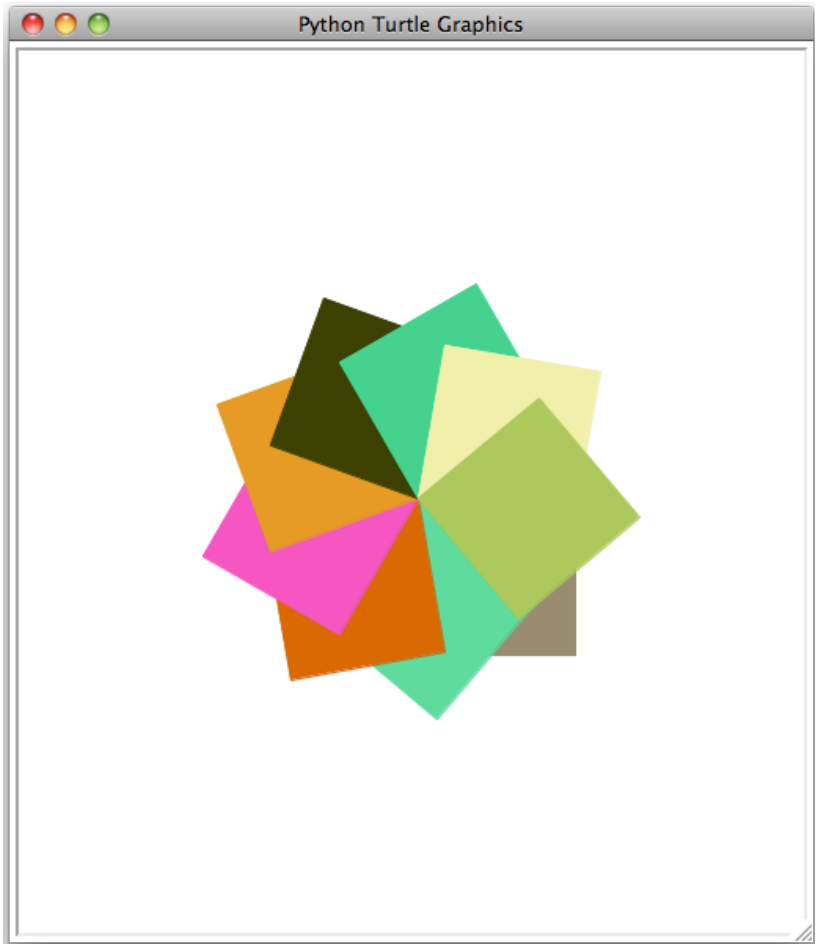
Note the underline in the name. Make the call to the `_exit` function the last line in the program.

Examples:
To illustrate, we show results of executing two programs, both of which meet the requirements.

The first program draws squares of a fixed size; they all start at the origin, but are arranged in a circular manner by manipulating the orientation of the turtle. An example interaction:



An example of the picture it draws:



The second program draws concentric circles of many colors. The user specifies the number of circles and the radius of the largest circle.

An example interaction:

```
Python Shell
>>> ================================ RESTART ================================
>>>
This program draws concentric circles of many different colors

Enter the number of circles to draw: 8.0
The number must be an integer and at least 1.
Please try again.

Enter the number of circles to draw: 8

Enter the radius (>=50, <=200) of the largest circle: 25
The radius must be an integer between 50 and 300.
Please try again.

Enter the radius (>=50, <=200) of the largest circle: 150
>>>
```

An example of the picture it draws: