Lab06 Turtle graphics (10.07.19) CS103 Fall 2019

author john k johnstone jkj at uab dot edu
course CS103 Fall 2019

version Fall 2019

---

materials

- lab06_19fa103.pdf (this document)
- lab06_19fa103.py (with functions you will implement)
- lab06_tests_19fa103.py (with test calls)
- lab partner: please work with the lab partner that you chose or were assigned in Lab02.

purpose

- to learn turtle graphics, using Python's turtle module
- to practice using the functions of a module
- to practice iteration using for loops
- to practice your Cartesian coordinates

preparation

- set up your screen with Canopy terminal on left and Canopy editor on right
- from the Canopy terminal, enter the Canopy interpreter (>>>) and verify that your environment is Python3 and Enthought Canopy (both are required for today's lab particularly)
- download the Canvas materials and put in your 19fa103/lab/lab06 directory
- open a website with the (excellent) Python documentation of turtle graphics:
  https://docs.python.org/3/library/turtle.html
  you may also find this page by googling 'python 3.5 turtle'
- visit lecture12 notes for a list of the important turtle functions that you will use in your code today
- explore the Python documentation for the turtle module,
  paying particular attention to the functions mentioned in Thursday's lecture
- note: all of the functions you write today are unusual: they do not have return values;
  this is unusual for Python functions;
  but a lack of return values is more common in computer graphics, where the rendering of a shape may be the useful task of the function, rather than computing a value.

preamble

You will draw line segments, squares, and triangles. For each problem, you will write a function that draws an arbitrary line/square/triangle, then call it with test data to draw a specific line/square/triangle. First, I suggest exploring in the interpreter, then generalizing and cleaning up in the function.

None of the geometric objects you draw with these test calls should overlap. To help this,

- draw the line segments in the first quadrant, with top right corner (200,200)
- draw the squares in the second quadrant, with top left corner (-200,200)
- draw the triangles in the third quadrant, with bottom left corner (-200,-200)
- draw the challenges in the fourth quadrant, with bottom right corner (200,-200)

definitions

rectilinear square a square is *rectilinear* if its sides are aligned with the coordinate axes
(so two sides will be horizontal, two sides vertical)
equilateral triangle a triangle whose sides are all the same length;
equivalently, a triangle whose angles are all the same (so $\pi/3$)

hints

- don't forget to lift your pen if you want to move without drawing
- don't forget to drop the pen once you are prepared to draw
- don't forget to add done() at the very bottom of your Python script, so that your window does not disappear immediately after drawing
- when possible, use a for loop
- remember the special style of importing the turtle module: `from turtle import *`
  compare the following pieces of code to see how this simplifies your code

```
import turtle
turtle.pencolor ('red')
turtle.goto (100, 0)
```

vs.

```
from turtle import *
pencolor ('red')
goto (100, 0)
```

In-class exercise

Draw a rectilinear square using forward, left and a for loop.
Call it to draw a large black square.

Exercises (with your lab partner)

line segments

- function to draw a line segment using forward;
  call it to draw a long black horizontal line segment
- **function to draw a line segment using goto**;
  call it repeatedly using a for loop to draw 10 equally-spaced short red vertical line segments
- call either function to draw a green line segment that is not horizontal or vertical, and almost touches the black/red lines (but doesn't)

squares

- function to draw a rectilinear square using forward, left, and a for loop;
  call it to draw a large black square
- **function to draw a rectilinear square using goto and a for loop**;
  call it to draw a small red square
- function to draw a filled rectilinear square using forward, right, and a for loop;
  call it to draw a green square
- function to draw a non-rectilinear square using a for loop
  (that is, an arbitrary square; forward is much easier than goto here)
  call it to draw a magenta square

triangles

- **function to draw an equilateral triangle using forward, left, and a for loop**;
  call it to draw a large black triangle
- function to draw an arbitrary triangle using goto;
  call it to draw a small red triangle
- function to draw a filled equilateral triangle using forward, right, and a for loop;
  call it to draw a green triangle

If you finish these exercises, please continue on to one of the challenges.

challenges

- function to draw a regular polygon with n sides; a polygon is regular if all of its sides are the same length; this function generalizes the equilateral triangle function you wrote above
- function to draw a line (not just a line segment); the way you implement this is to find the intersections of the line with the window and draw that line segment
- draw a tic-tac-toe board using a for loop, then generalize to a chess board, then (trivially if you have written the function properly) to a go board
- render the random points in a square challenge from lab05: draw the points inside the circle red and the points outside the circle black
- randomize the test data (lines/triangles/squares) you generate for today's functions, while still avoiding overlap and keeping the geometric objects visible on the screen

toroidal square challenge

Here is a fun one.
Write a function **toroidalTurtle** to move a turtle in a toroidal square.
Toroidal squares were common in early video games, where a plane would fly out one side of the square and immediately appear on the opposite side.

In a toroidal square, the left and right sides are identified, and the bottom and top sides are identified. So if you leave through the right side, you re-enter through the left side. And if you leave through the top, you immediately re-enter through the bottom. If you finish this, try moving in a standard square, bouncing off the walls. You can try various bouncing strategies.

input: starting point inside the square (a 2-tuple), heading (an angle)
rendering: in the toroidal square, animate the turtle moving from start in the direction specified by the heading.

Don't worry about an infinite loop: just keep moving until the program is killed.

Deliverables

Since this is a busy week (HW3 due Wednesday, midterm Thursday), I have relaxed the deliverables for this lab. Happy coding! Happy studying!

A attendance (that is, full participation throughout lab) + in-class exercise
A+ attendance + in-class exercise + boldfaced line/square/triangle exercises