# Problem Set 1 Turtles Graphics

## Setup

Open ps1\_template.py.

Note the intializeTurtle() function that sets up the turtle and screen.

In this function, you'll see the speed of the Turtle is set to 100 by default. As you're working, you might find it useful to **reduce this speed value** so the Turtle moves slower, making it easier to track.

*# Set the speed; 0=No animation, 1=slowest, 6=normal, 10=fast, etc.*

speed(100) *# can change this to vary the speed of your turtle*

At the bottom of the starter file is a function run, which invokes initalizeTurtle. The invocations of the functions you write (e.g., row()) will go inside run().

## Task 1. Helper square function

In this task, you will choose one of the versions of the square() function below to use in your recursive\_turtles.py file.

Note that Version A square() is the simplest; Version B contains a for loop and is **not** recursive.

*# Version A: No loops, no recursion*

**def** **square**(length):

"""draws a square of size length maintains

heading and position invariant"""

fd(length)

lt(90)

fd(length)

lt(90)

fd(length)

lt(90)

fd(length) *# back to starting location*

lt(90) *# heading in same direction as start*

*# Version B: For loop, no recursion*

**def** **square**(length):

"""draws a square of size length maintains heading

and position invariant"""

**for** i **in** range(1,5): *# or for i in range(4)*

fd(length)

lt(90)

Choose **one** of the square functions above and paste it into your recursive\_turtles.py file, right where it says # WRITE YOUR FUNCTIONS HERE.

Then invoke square in the run function.

Once you run your file and see how square works, move on to Task 2 which will use this function.

## Task 2: Row of squares

Write a recursive method called row(number, size) that draws a horizontal row of number squares of size size.

**This function must be recursive and maintain a position and heading invariant.**

FYI: In the example screenshots below, the Turtle's pen was set to red ( via pencolor('red')), but your pen color will be black by default.

|  |
| --- |
| row(3,50) http://cs111.wellesley.edu/content/labs/lab07/images/row3.png |
| row(5,25) http://cs111.wellesley.edu/content/labs/lab07/images/row5.png |

## Task 3: Spaced row of squares

Write a recursive method called spacedRow(number,size) that draws a horizontal row of number squares of size size with a constant space of size/4 between each square.

**This function must be recursive and maintain a position and heading invariant.**

|  |
| --- |
| spacedRow(3,50) http://cs111.wellesley.edu/content/labs/lab07/images/space3.png |
| spacedRow(5,25) http://cs111.wellesley.edu/content/labs/lab07/images/sapce5.png |

## Task 4: Decreasing row of squares

Write a recursive method called decreasingRow(number,size) that draws a horizontal row of numbersquares. The first square has size size and each subsequent square is half the size of the previous one.

**This function must be recursive and maintain a position and heading invariant.**

|  |
| --- |
| decreasingRow(3,100) http://cs111.wellesley.edu/content/labs/lab07/images/decrease3.png |
| decreasingRow(5,100) http://cs111.wellesley.edu/content/labs/lab07/images/decrease5.png |

## Task 5: Nested squares

Write a recursive method called nestedSquares(number,size) that draws number nested squares. The largest square has size size and each subsequent square is half the size of the previous one. The squares are anchored in the lower left corner.

**This function must be recursive and maintain a position and heading invariant.**

|  |
| --- |
| nestedSquares(3,90) http://cs111.wellesley.edu/content/labs/lab07/images/nest3.png |
| nestedSquares(5,130) http://cs111.wellesley.edu/content/labs/lab07/images/nest5.png |

## Task 6: Diagonal squares

Write a recursive method called diagonal(number,size) that draws number decreasing diagonal squares. The largest square has size size and each subsequent square is half the size of the previous one.

**This function must be recursive and maintain a position and heading invariant.**

|  |
| --- |
| diagonal(1,100) http://cs111.wellesley.edu/content/labs/lab07/images/d1.png |
| diagonal(2,100) http://cs111.wellesley.edu/content/labs/lab07/images/d2.png |
| diagonal(3,100) http://cs111.wellesley.edu/content/labs/lab07/images/d3.png |
|  |

## Task 7: Super diagonal squares

Write a recursive method called superDiagonal(number,size) that draws number decreasing diagonal squares. The largest square has size size and each subsequent square is half the size of the previous one. In superDiagonal, squares are placed on the diagonal at both the upper right corner and the lower left corner.

**This function must be recursive and maintain a position and heading invariant.**

|  |  |
| --- | --- |
| superDiagonal(1,125) http://cs111.wellesley.edu/content/labs/lab07/images/justbox.png | superDiagonal(2,125) http://cs111.wellesley.edu/content/labs/lab07/images/super2.png |
| superDiagonal(3,125) http://cs111.wellesley.edu/content/labs/lab07/images/super3.png | superDiagonal(4,125) http://cs111.wellesley.edu/content/labs/lab07/images/super4.png |

## Task 8: Prosperity Turtle Challenge

The Chinese New Year festivity is round the corner again. It is time to gear up and maximize the red packet collection trade. You reckon that building good rapport and striking an impression is ultimately important in a big red packet. After all, you only legitimately get red packets once a year from adults (other than your parents)!

You decided that the best way to make an impression is to showcase your newly learnt programming tool, Python’s turtle graphic module, by ‘designing and drawing’ an original greeting card for your prospective giver. In this time and age where snail mailing of greeting cards are considered old-fashioned, and generating free e-cards is considered too common and may be dangerously viewed as being insincere, you reckon that by putting in effort in designing a unique card on your own will be much more appreciated by the receiver.

In this problem set, you will be using the turtle graphic module to create an original greeting card.

**Requirements:**

* The time taken to ‘draw’ the greeting card should not be too lengthy! Keep it under 15 seconds.
* Utilize functions when creating generic shapes – so that your code will not be unnecessarily long and error prone.

**Assessment Criteria:**

Your greeting card will be assessed based on the following criteria:

* Overall aesthetic appeal
* Creativity and Originality
* Appropriateness for use as a greeting card
* Coding style (you are highly encouraged to use concepts like functions, loops to make your program more readable)