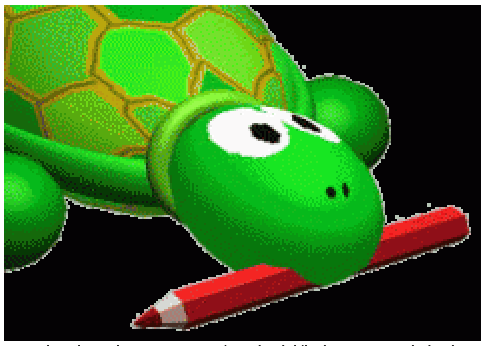
Turtle Graphics


Picture from: http://www.xda-developers.com/android/bring-nostalgia-back-with-turtle-graphics/

Objectives:

- Practice writing classes

Lab Overview:

(*Turtle Graphics*) The Logo language, which is particularly popular among personal computer users, made the concept of *turtle graphics* famous. Imagine a mechanical turtle that walks around the room under the control of a Python program. The turtle holds a pen in one of two positions, up or down. While the pen is down, the turtle traces out shapes as it moves; while the pen is up, the turtle moves about freely without writing anything. In this problem, you will simulate the operation of the turtle moving around a gridded surface by creating a Turtle class. The commands that are sent to the turtle are stored in a file. The file is read, the turtle executes each command, and then the program is done.

The Turtle Commands:

The set of turtle commands your program must process are as follows:

Command	**Meaning**
1	Pen up
2	Pen down
3	Turn right
4	Turn left
5, 10	Move forward 10 spaces (or a number other than 10)
6	Print the 2-dimensional array using a character of your choosing such as *
9	End of data (sentinel)

The Turtle class consists of the following instance data items:

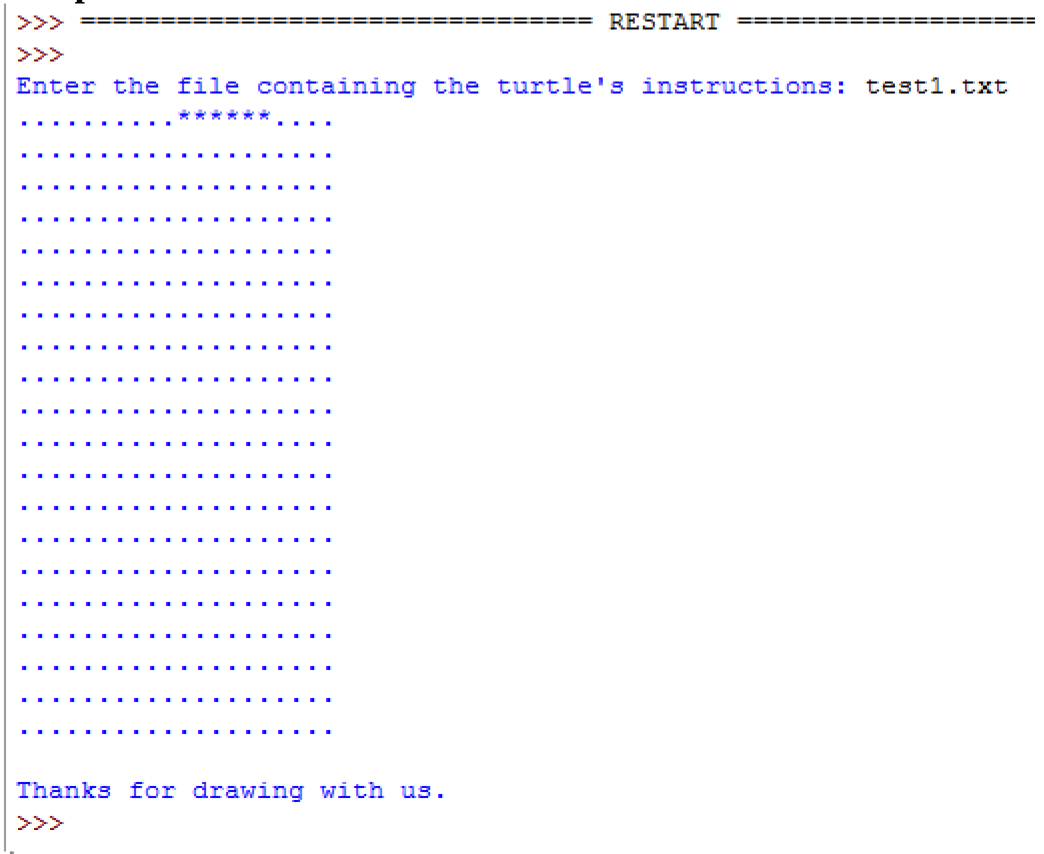
- Gridded surface that the turtle is walking on (2-dimensional list) where a period (.) represents an empty space and an asterisk (*) represents a space that has been "drawn" on.
- Pen position – whether up or down
- Location of turtle on grid. This is actually 2 data items, you'll need to keep track of the row and the column.
- Direction the turtle is facing. Note: It's easiest if 0 represents North, 1 represents East, 2 represents South, and 3 represents West.

The following are the behaviors:

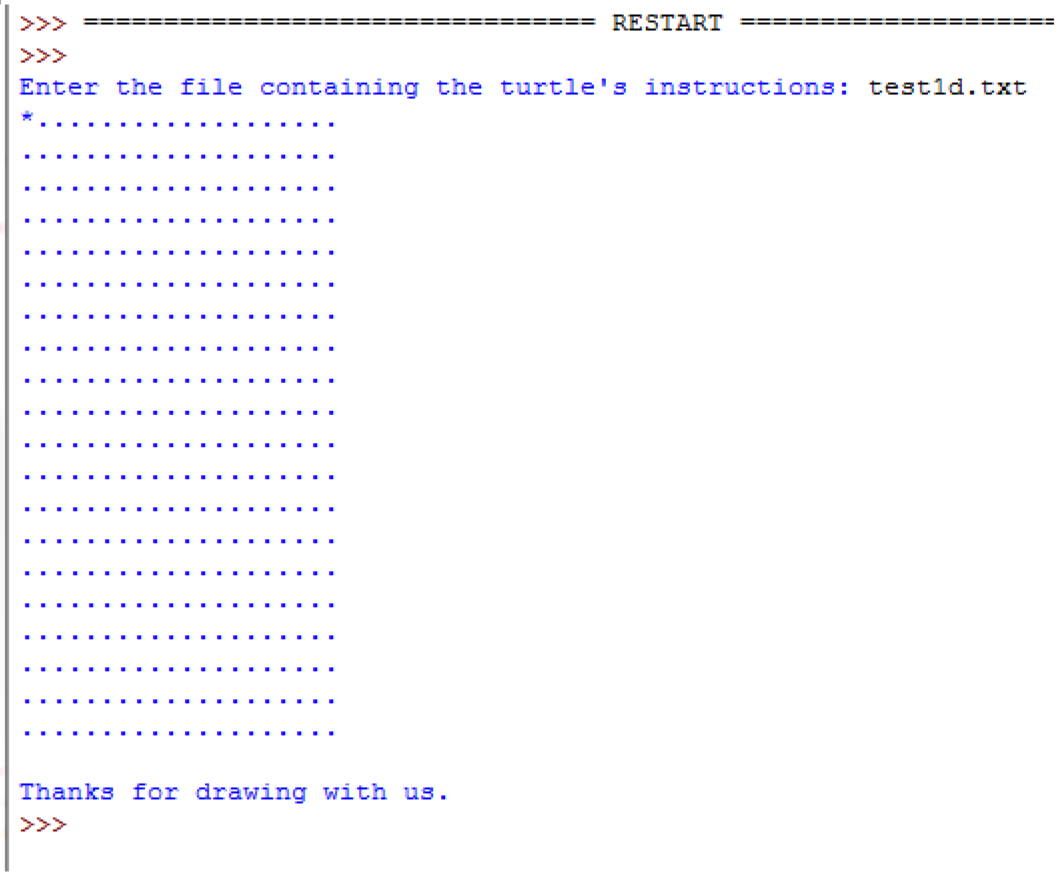
- change_pen_position. It requires an integer to be passed to it. If a 1 is sent, then the pen is up. If a 2 is sent, then the pen is down. If the pen is changed to the down position, it marks the square that the turtle is in.
- turn_right. Nothing is sent, if the method is called, the data item representing the direction the turtle is facing is changed accordingly.
- turn_left. Nothing is sent, if the method is called, the data item representing the direction the turtle is facing is changed accordingly.
- move_forward. This method is the longest. An integer is passed that is the number of steps that the turtle is supposed to take. If the pen is up, the turtle simply moves in the direction that it is facing until it has taken the appropriate number of steps or it gets to the edge of the gridded surface. You can picture the turtle as one of those mechanical puppies that walks until it gets to a wall. Once it hits the wall, it continues to move its legs, but it doesn't actually go anywhere. If the pen is down, it draws a line as it goes (stores an asterisk in each square as it passes through).
- print_grid. It doesn't require any information from the calling program. It prints the contents of the grid (remember a period represents no line has been drawn through a square, but an asterisk does) to the screen.

The program that creates the Turtle object, gets the file name from the user, reads the commands from the file, and calls the Turtle methods based on the commands has already been written. Your job is to write the class. Several different files are provided. Your class should be able to handle all of them. Start with test1 and proceed in order.

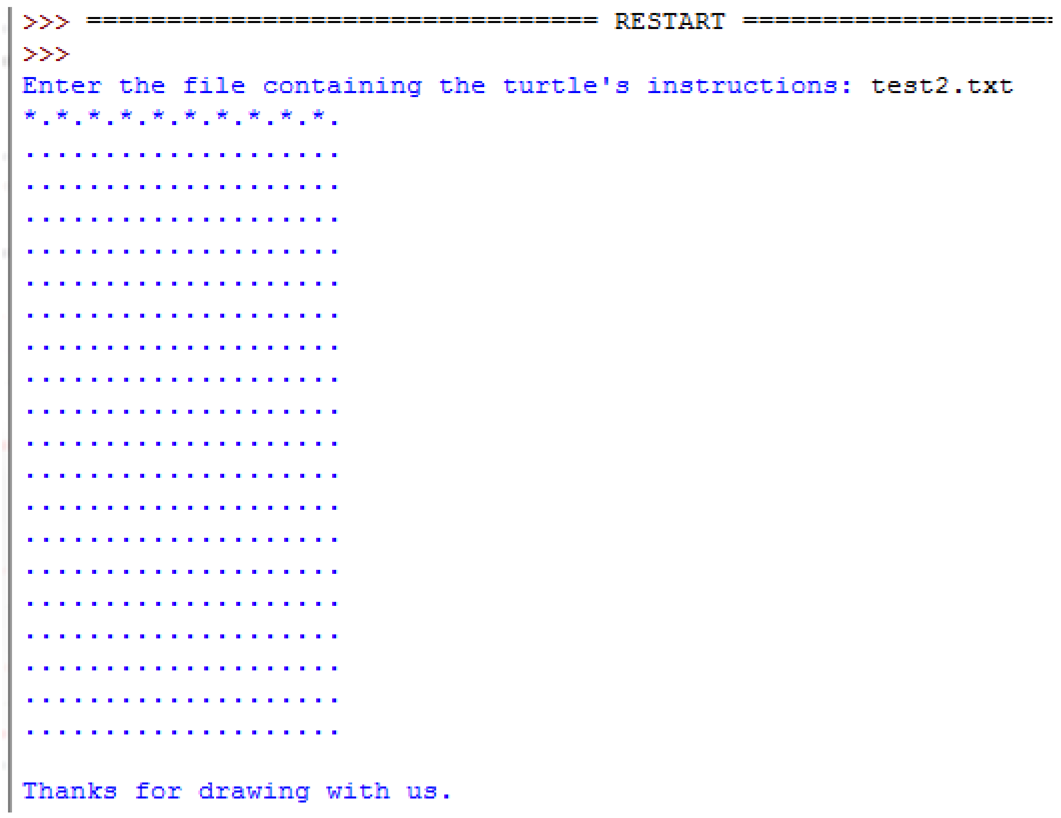
Sample Execution:

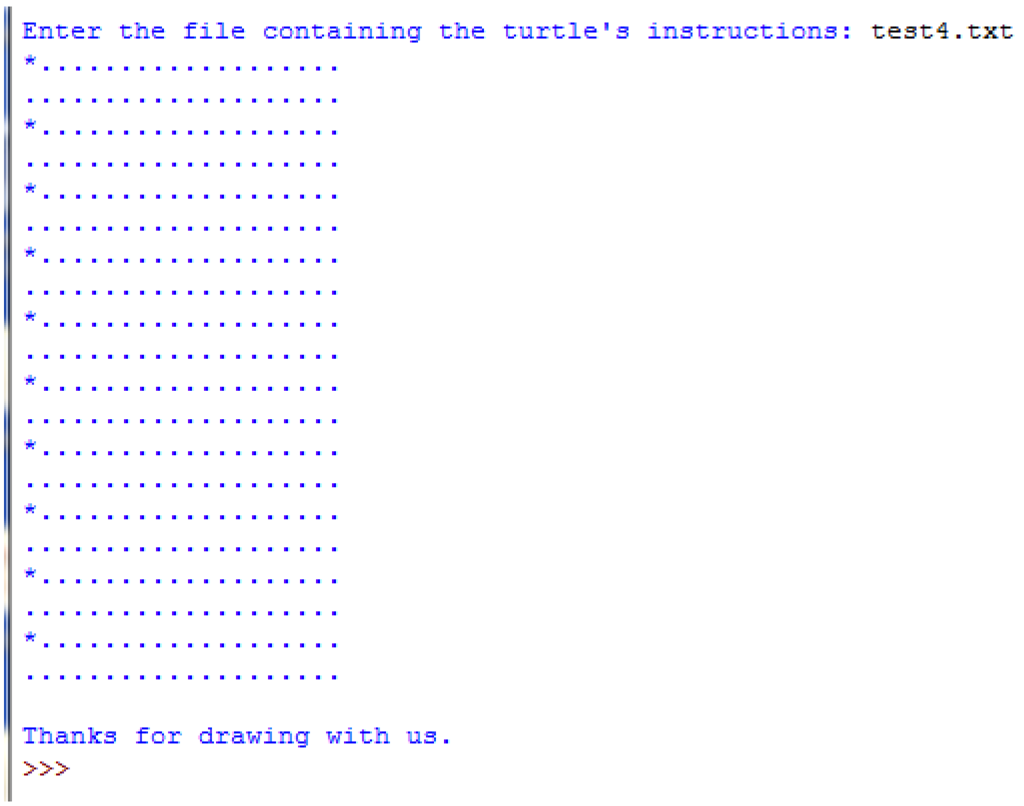


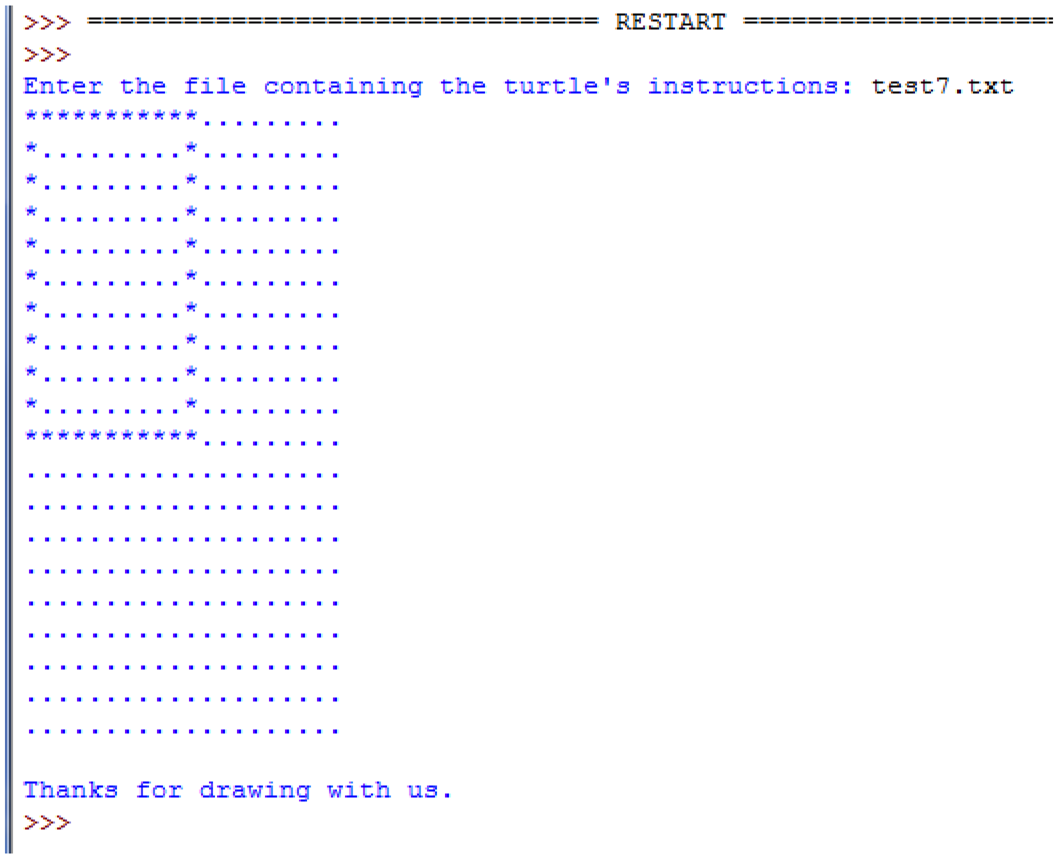
```
>>> ================================ RESTART ================================
>>>
Enter the file containing the turtle's instructions: test1.txt
..........******....
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................

Thanks for drawing with us.
>>>
```



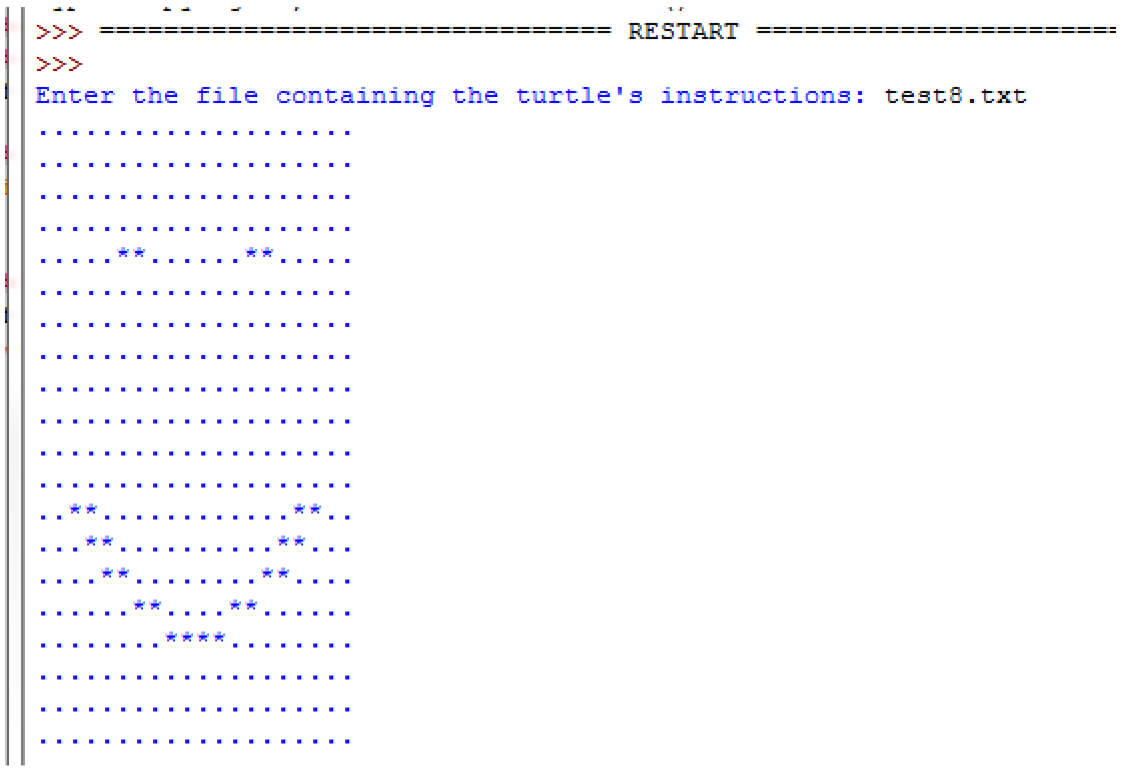
```
>>> ================================ RESTART ================================
>>>
Enter the file containing the turtle's instructions: test1d.txt
*...................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................

Thanks for drawing with us.
>>>
```


```
>>> ================================ RESTART ====================
>>>
Enter the file containing the turtle's instructions: test2.txt
*.*.*.*.*.*.*.*.*.*.
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................

Thanks for drawing with us.
```


```
Enter the file containing the turtle's instructions: test4.txt
*...................
....................
*...................
....................
*...................
....................
*...................
....................
*...................
....................
*...................
....................
*...................
....................
*...................
....................
*...................
....................
*...................
....................

Thanks for drawing with us.
>>>
```



```
>>> ================================ RESTART ====================
>>>
Enter the file containing the turtle's instructions: test7.txt
***********.........
*.........*.........
*.........*.........
*.........*.........
*.........*.........
*.........*.........
*.........*.........
*.........*.........
*.........*.........
*.........*.........
***********.........
....................
....................
....................
....................
....................
....................
....................
....................
....................

Thanks for drawing with us.
>>>
```



```
>>> ================================ RESTART ======================
>>>
Enter the file containing the turtle's instructions: test8.txt
....................
....................
....................
....................
.....**......**.....
....................
....................
....................
....................
....................
....................
....................
..**............**..
...**..........**...
....**........**....
......**....**......
........****........
....................
....................
....................
```

Program Requirements:

For the code:

- Your program should have the correct comment block at the top (see last assignment)

- Use appropriate comments throughout the code. Be sure to include comments before the class definition and before each method (member function) explaining the interface (what must be sent, what is returned) and the purpose of the function. Also, don't forget to make appropriate comments within the method definition.

- Make good use of whitespace. Be sure to include a blank line of space before each comment. Two lines of space between functions is nice, but one will suffice – whichever you choose, be consistent. Follow the Python programming style guide: https://www.python.org/dev/peps/pep-0008/#introduction

For the lab report, follow the Lab Report Format Guide and complete the following sections:

- Title Page
- Design – Create the UML diagram for the Turtle class.
- Analysis and Conclusions
- Extra Analysis Questions:
  - Did the test files help you to debug your class? If so, how?
  - What did you learn about writing classes from this lab?
  - What did you learn about creating tests from this lab?
- Appendix B - code (Be sure that you copy and paste the code into your word document, don't take a screen shot of it and paste that – it ends up being too hard to read.)

Deliverables:

Electronic submission in myCourses:

- Code (due by the end of the class period)
- Lab Report (due by the start of the next time that your lab meets) -- either a Microsoft Word document or a pdf

Paper submission in class:

- Lab Report (due at the start of lab the next time that it meets)

Grading:

Task	**Points**
Lab Report	30 points
Title Page	5 points
Design	5 points
Analysis and Conclusions	10 points
Extra Analysis Questions	10 points
Code	70 points
Pass all of the test1 files	10 points
Pass the test2 file	10 points
Pass all of the test3 files	5 points
Pass the test4 file	5 points
Pass the test5 file	10 points
Pass the test6 file	10 points
Pass the test7 file	10 points
Pass the test8 file	10 points