Computer Science Courses

Project 5: The dragon hits again

- Due date: December 6th, 23:59PM
- Teamwork reflection due date: December 6th, 23:59PM

- This is a team project.

- The project is worth 100 points.

- All the team members will get an equal grade.

- **ONLY the team leader must turn-in the project code**.
- **Each** team member is also required to answer some questions about the teamwork experience and to submit them **individually**. Please see section `TODO 8: Teamwork reflection` of this assignment for the description of the questions and how the answers to them must be turned-in. **You will loose 5 points if you do not submit the teamwork reflection answers**.

UPDATE: Test case 3 updated.

VERY IMPORTANT!!!

You **MUST** specify the following information at the beginning of the project4 file:

```
# Team leader: Last name, First name, Purdue account
# Team member 1: Last name, First name, Purdue account
# Team member 2: Last name, First name, Purdue account
```

Before submitting the file project5.py, check carefully that the header above is correctly completed:

- If the file project5.py does not contain the header above, **only the student submitting the file will receive the grade**. This will apply also in the case that the team was registered.
- **Only the team members listed in the header of project5.py file will receive the grade.**

Libraries:

- This project will use the Python libraries, PIL Image.
- You might need the Math package for computations.

Project description

In this project you will use your knowledge of recursive functions to generate a string encoding a Dragon curve of order *n*, where the order of the curve will be provided by the user.

Then you will translate the L and R string into a NSWE string, where N,S,W,E mean respectively:

- go forward in the north direction
- go forward in the south direction
- go forward in the west direction

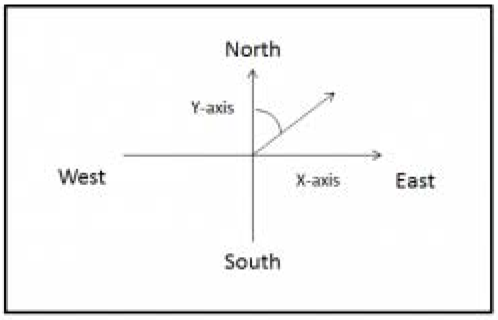
- go forward in the east direction

Then you will use the NSWE encoded string as input and draw the corresponding curve in a 2D window. The dimensions (width and height) of this window will be given to you in a file (`Parameters.txt`).

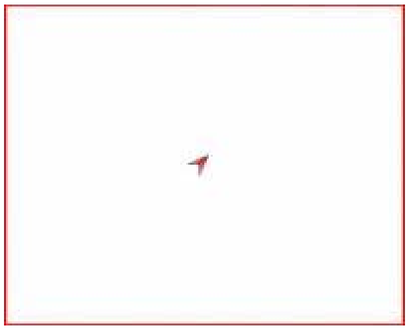
The **initial position** of the `turtle`, that is its X and Y coordinates,as well as the **length** of the movement will be given to you in the same file containing the window dimensions. Each movement will have the same length.

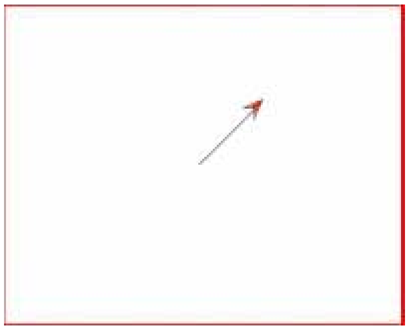
The initial orientation of the `turtle` will be given by the user as an angle (expressed in degrees) between the line passing through the turtle and the positive Y axis:

- 0 degrees means that the initial orientation will be parallel to Y axis
- 45 degrees means that the initial orientation is at an angle of 45 degrees to the positive y-axis (see Figure 1 below)

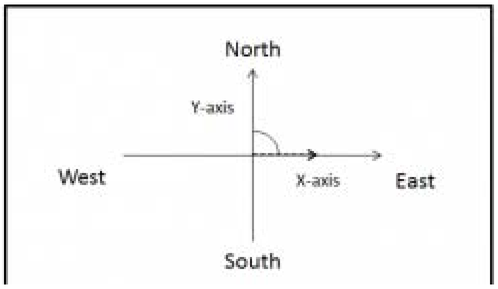


Case 1 - initial position at (0,0); initial orientation angle = 45

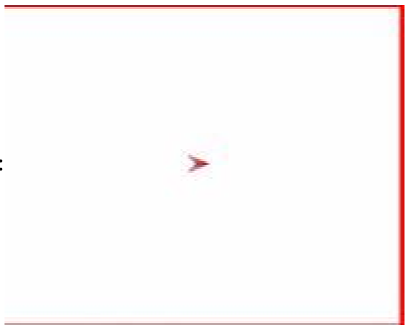
Initial Orientation of turtle : 

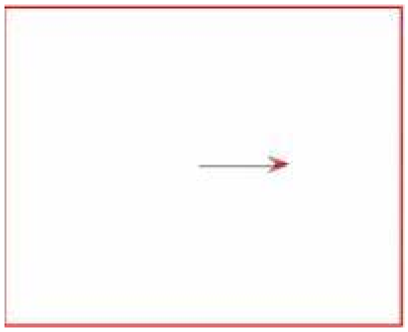
Position of turtle after movement in North Direction: 

- 90 degrees means that the initial orientation will be parallel to the X axis (see Figure 2 below)



Case 2 - initial position at (0,0); initial orientation angle = 90

Initial Orientation : 

Position of turtle after movement in North Direction: 

SETUP

1. Create a folder **project5** under your folder CS177.
2. Download the code below (to that just click the **filename project5-sk.py** link below:

filename project5-sk.py

```
# CS177 Fall 2012 - Project5
#
# IMPORTANT: The following information MUST be filled!!!
#
# Team leader: Last name, First name, Purdue account
#
# Team member 1: Last name, First name, Purdue account
#
# Team member 2: Last name, First name, Purdue account
#

from PIL import Image
from PIL import ImageDraw
import math

#TODO1 write function getParameters



#TODO2 write function flipAxis



#TODO3 write function rotate
```

```
#TODO4 write function drawLineNSEW



#TODO5 write the function dragon



#TODO6 write the function cvtLR2NSEW



#TODO7 write the function main
```

- Under project5 folder, download the file parameters.7z. You need to extract the file Parameters.txt from the .7z archive. This file contains the parameters you will use:
- the first parameter is the windows' width
- the second parameter is the windows' height
- the third parameter is the length of the segment line you must draw
- the fourth parameter is the initial position of the `turtle` in the format (x, y)

This is the content of the file Parameters.txt provided to you:

```
100 100 10 (25,75)
100 100 25 (25,50)
100 100 20 (20,40)
150 150 20 (60,40)
150 150 15 (100,40)
300 300 10 (130,200)
300 300 10 (130,200)
300 300 10 (130,200)
500 500 10 (160,300)
500 500 10 (110,300)
550 550 8 (150,190)
650 650 6 (250,180)
800 800 3 (550,540)
```

Movement direction

- when drawing, you must use a coordinate system with the origin (0,0) in the bottom left corner of the image (remember that PIL has the origin in the top left corner)
- The direction of the line segment corresponding to the movement to be drawn must be determined by taking into account the angle provided by the user and the character you are seeing (N, S, W, E)

Project details

TODO 1:

Write the function `getParameters`

The function will read the file Parameters.txt given to you and will create a list containing the corresponding parameters.

Note that the file Parameters.txt contains the following lines:

```
100 100 10 (25,75)
100 100 25 (25,50)
100 100 20 (20,40)
150 150 20 (60,40)
150 150 15 (100,40)
300 300 10 (130,200)
300 300 10 (130,200)
300 300 10 (130,200)
500 500 10 (160,300)
500 500 10 (110,300)
550 550 8 (150,190)
650 650 6 (250,180)
800 800 3 (550,540)
```

but you have to generate the following list:

```
[
['100', '100', '10', '(25,75)']
['100', '100', '25', '(25,50)']
['100', '100', '20', '(20,40)']
['150', '150', '20', '(60,40)']
['150', '150', '15', '(100,40)']
['300', '300', '10', '(130,200)']
['300', '300', '10', '(130,200)']
['300', '300', '10', '(130,200)']
['500', '500', '10', '(160,300)']
['500', '500', '10', '(110,300)']
['550', '550', '8', '(150,190)']
['650', '650', '6', '(250,180)']
['800', '800', '3', '(550,540)']
]
```

The function must return the list above.

NOTE:

- Each element of the list shown above is a list of strings .

TODO 2:

Write the function `flipAxis`.

- This function will have two arguments, *position* and *bounds*:
  - `position` is the 2D coordinate of the current position, that is, a **tuple** of integers x and y, where x is the x-axis position, and y is the y-axis position
  - `bounds` is a **tuple** of (maxX,maxY), which is the width and height of the current image.

This function will translate the `position` from that of a standard coordinate system - that is, one where the origin (0,0) is in the bottom left corner -, to one which PIL understands. Since we want the origin (0,0) to be in the bottom left corner of the image and PIL images have the origin in the top left corner, we need to convert between the two.

The function will return the tuple containing the new x and y coordinates.

TODO 3:

Write the function `rotate`.

This function will rotate a point (x,y) by an angle specified by the user about the origin in clockwise direction.

- This function will take in three input arguments, *x*, *y*, and *angle*:
  - `x` is the current position on the x-axis
  - `y` is the current position on the y-axis
  - `angle` is the angle given by the user
- This function will return the new point `(x',y')` which is the current point `(x,y)` rotated by `angle` about the origin.

TODO 4:

Write the function `drawLineNSEW`.

This function will draw a line of `length` pixels in the direction specified by the `direction` input argument. All lines must be black.

- This function will take in six input arguments, *direction*, *myImage*, *position*, *angle*, *color*, and *length*:

- `direction` is a single character ('N', 'S', 'E', or 'W') that determines the direction of the line
- `myImage` is the PIL image you are editing
- `position` is the 2D coordinate of the current position, that is, a tuple of integers x and y, where x is the x-axis position, and y is the y-axis position
- `angle` is the angle (in degrees) with which the line will be rotated.
- `color` is a triple (r,g,b) which will be the desired color of the line
- `length` is the number of pixels to draw each line segment
- This function will return the x,y coordinate of the pixel corresponding to the ending point of the drawn line segment

HINTS

1. Convert the angle to radians.
2. Get the end position of the line based on the direction you are moving.
3. Rotate the end position by the angle.
4. Draw the line from the start position to the end position using the line() method of ImageDraw.

TODO 5:

Write the function `dragon`.

This function will generate the string of R's and L's based on the number of folds.

- This function will take in two input arguments, *gen*, *root*:
  - `gen` is the number of folds the curve is going to be.
  - `root` is the letter of the current node
- This function will return the the string of R's and L's

HINTS

1. It is a recursive function.
2. You will have the same number of R's on the left side of the root as the number of L's on the right side of the root.

TODO 6:

Write the function `cvtLR2NSEW`.

This function will convert the string of R's and L's to NSEW depending on the current orientation.

- This function will take in two input arguments, *str, dir*:

  - `str` is the string of R's and L's generated by the dragon function
  - `dir` is the initial orientation of the turtle
- This function will return a string consisting of NSEW characters

HINTS

1. If you are facing N and you see an L, you will move W.
2. If you are facing N and you see an R, you will move E.

TODO 7:

Write the function `main`. This function has no arguments.

- The function must do the following things:
  1. Read the file Parameters.txt and build the list containing the parameters (see **SETUP** and **TODO1**)
  2. Ask the user what he wants to do: draw and dragon curve or exit.
  3. If the user don't want to exit
     a. Ask the user for the number of folds, and store it in a variable. NOTE: the number of folds MUST be an integer between 0 and 14.
     b. use the number of folds as the index into the list (of lists) that you built in TODO1. You must get the windows width, the window height and the line segment length from this list.
     c. Ask the user for the initial angle, and store it in a variable.
     d. Call dragon and get the string of R's and L's.
     e. Convert the string to NSEW
     f. Print the string of NSEW to the screen.
     g. Ask the user for the initial direction of the turtle (N/S/E/W)
     h. Draw the curve
     i. Save the image when you are finished. The saved file must have the name "dragon_image_#_a_angle.jpeg" where # is the number of folds and angle is the user's input for angle. For example, if the # of folds is 1 and the input angle is 45 degrees, the file is named "dragon_image_1_a_45.jpeg"
  4. Otherwise exit.

The prompts should look like the following ones:

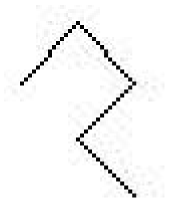
```
type command --
   1 - dragon
   2 - exit
:: 1
number of folds: 2
initial angle: 45
initial direction (N/S/E/W): N
NESE
```

Test Cases

1. **TestCase1**

```
type command --
   1 - dragon
   2 - exit
:: "1"
number of folds: "2"
angle: "45"
initial direction (N/S/E/W): "N"
NESE
```

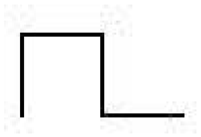
```
image saved
```



---

1. **TestCase2**

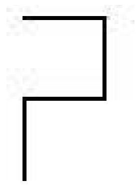
```
type command --
   1 - dragon
   2 - exit
:: "1"
number of folds: "2"
angle: "0"
initial direction (N/S/E/W): "N"
NESE
image saved
```



---

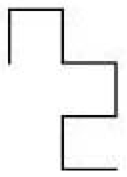
1. **TestCase3**

```
type command --
   1 - dragon
   2 - exit
:: "1"
number of folds: "2"
angle: "0"
initial direction (N/S/E/W): "E"
ESWS
image saved
```



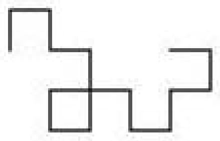
- **TestCase4**

```
type command --
   1 - dragon
   2 - exit
:: "1"
number of folds: "3"
angle: "0"
initial direction (N/S/E/W): "N"
NESESWSE
image saved
```



- **TestCase5**

```
type command --
   1 - dragon
   2 - exit
:: "1"
number of folds: "4"
angle: "0"
initial direction (N/S/E/W): "E"
ESWSWNWSWNENWNWS
image saved
```



TODO 8: Teamwork reflection

Each of the team members is required to answer the following questions and submit the answers individually.

1. How much faster do you think you completed the project by working in a team, compared to working on it alone?

2. Did you gain skills in conflict resolution while working on the team? Were you able to use these skills effectively to enhance your team's performance? Illustrate with an example.

3. This is the second time you are working in the same team. Describe whether and how the experience of the first team project helped you improve your teaming experience for this second team project.

Write the answer to the questions above in a file named `<your Purdue account>-teamreflect.txt. For example: lmartino-teamreflect.txt`.

Each team member MUST submit this file.

GRADING RUBRIC

TODO	Points
TODO1	10
TODO2	5
TODO3	10
TODO4	20
TODO5	10
TODO6	20
TODO7	20
PAPER	5
TOTAL	100

TURNIN INSTRUCTIONS

Login into your UNIX CS account. Then go to your project5 folder using the following commands:

```
$cd CS177
$cd project5
```

Then launch the following command AS IS:

```
$turnin -v -c cs177=COMMON -p project5 *.py *.txt
```

cs17700/projects/project5.txt · Last modified: 2012/12/06 17:06 by vgandiko