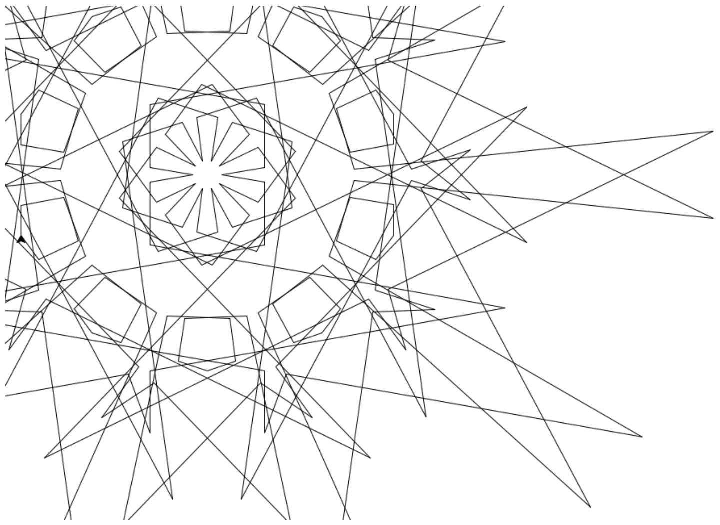
Computational Problem Solving CSCI-603
Sun Star Lab 3

6/3/2021



In your programming assignment you will be developing a graphical application that draws patterns in the turtle window using recursive techniques.

1 Problem Solving Session

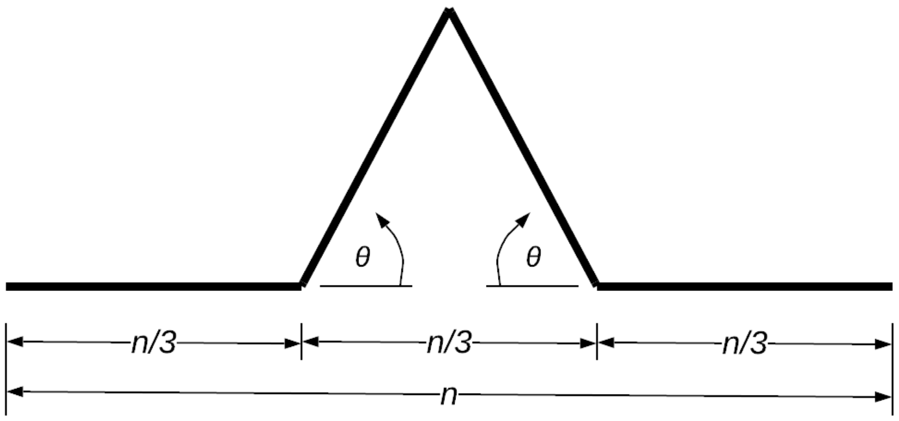


Figure 1. Specification for one side of a Koch Snowflake drawing at level 2

1. Write a simple function `draw_side1(n: int)` that draws a straight line. `n` is the length of the line. As preconditions, the turtle will be facing east with the pen down, and the post conditions are that the turtle is left at the end of the drawing just done, pen down.
2. Write a function `draw_side2(n: int)` that draws the lines shown in Figure 1. `n` is the horizontal length of the entire drawing. Assume that θ is a fixed angle of $60°(\pi/3$ radians). To draw the second segment, you must replace it by calling twice `draw_side1` with the correct length.
3. Now create your recursive function `draw_side` that takes a length and level parameters. `level` is always positive (not even zero). The behavior should be:
   - If the level is 1, the image from `draw_side1` should be drawn.
   - If the level is 2, the image from `draw_side2` should be drawn.
   - If the level is 3, Figure 2 below should be drawn.
   - If the level is 5, Figure 3 below should be drawn.

   Note: You are not allowed to call `draw_side1` nor `draw_side2`.

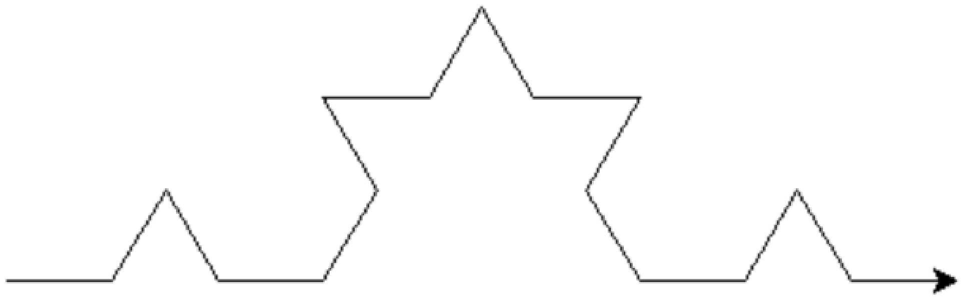


Figure 2. `level` = 3

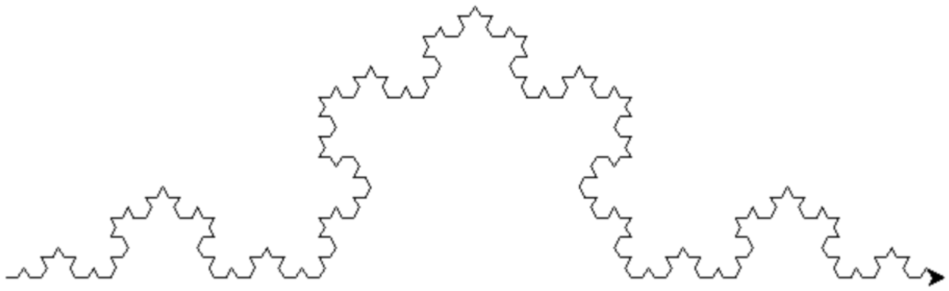


Figure 3. `level` = 5

4. Finally, change `draw_side(n,level)` so that it returns the complete (cumulative) length of all the lines drawn. The total length must be computed recursively rather than with a closed formula.

At the end of problem-solving, put all group members' names on the sheet, number each item and hand in your work, one copy per team.

2 Implementation

For the implementation you will develop a program called `sunstar.py` to draw a sun star. The sun star is similar to an equilateral, equiangular polygon in that each side of the polygon is identical. The difference is that the pattern for each polygon's side will be a slightly variation of the Koch curve from the problem-solving session.

2.1 Polygon Side Pattern

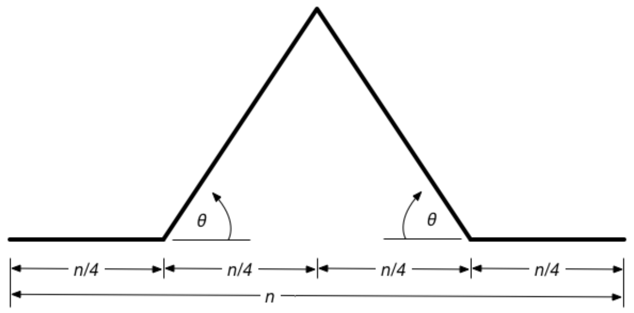


Figure 4. Specification for one side of a Sun Star drawing at level 2

Modify the `draw_side` function from the problem-solving session to generate the following fractal curve. This function must receive as parameters the side's length `n`, level `l`, and the deviation angle θ. It should draw one side of the sun star and returns its total length.

- If the level is 1, the function should draw a straight line of length `n`.
- If the level is greater than 1, the second and third segments are drawn using a recursive call to `draw_side` with the level one less.

Figure 5 shows such a drawing with level initially equal to 3. Figure 6 shows such a drawing with level initially equal to 7. Note that if the level is 2, the drawing would be as Figure 4.

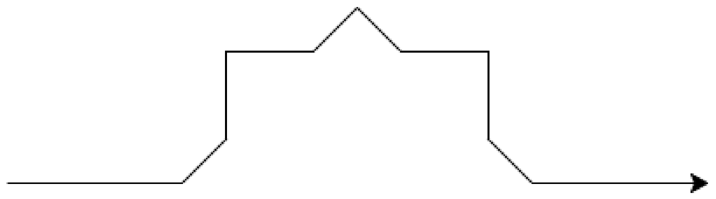


Figure 5. `level` = 3

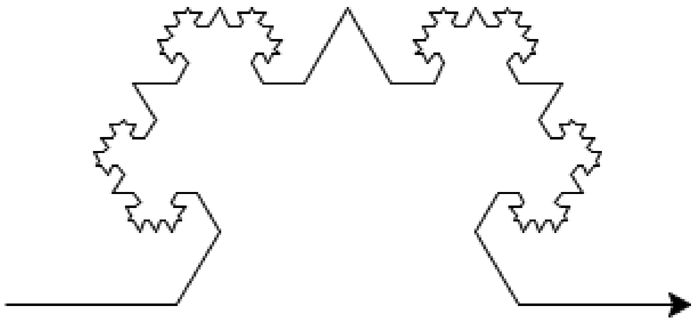


Figure 6. `level` = 7

2.2 Specification

Your program will start up and immediately prompt the user, via standard input, for three or four values.

1. The number of sides the shape will have (positive integer)
2. The length of one side in pixels (technically, its level-1 equivalent length) (positive integer)
3. The number of levels (positive integer)
4. The deviation angle θ in degrees (integer)
   (*This parameter is only requested if the number of levels is greater than 1.*)

All user input must be error-checked, with error messages of this form displayed when needed.

`Value must be a` *type*. `You entered` '*string-value*'.

The user is given an unlimited number of chances to enter a legal value. The program only need check for correct type, not correct range of value, e.g. non-negative.

Hint. Look at the `fullmatch` function in the `re` package. Also study the form of *regular expression patterns*. For example, `r"[0-9]+"` is a pattern matched by all strings only containing decimal digits. (As an alternative, and if you have experience with exceptions, you can handle the exception thrown when numeric conversion fails.)

Once the parameters are all entered, the sun star should be drawn on the turtle's canvas. Note that, when your program draws the figure, it may be off center. This is OK.

In addition, your program must print, on standard output, the total length of the lines drawn.

(Note that if a deviation angle of 0 is given, the number of levels is irrelevant; a simple polygon gets drawn, albeit more inefficiently if the number of levels is large.)

What follows is a sample run of your program for a specific set of parameter values. If you actually run your completed program, the following text should appear in the console. The only user-entered information is after the colons. The turtle will draw the figure shown in Figure 7 in a separate canvas window.

```
Number of sides: 10
Length of initial side: 150
Number of levels: 4
Deflection angle: 45
Total length is  2185.6601717798208
```

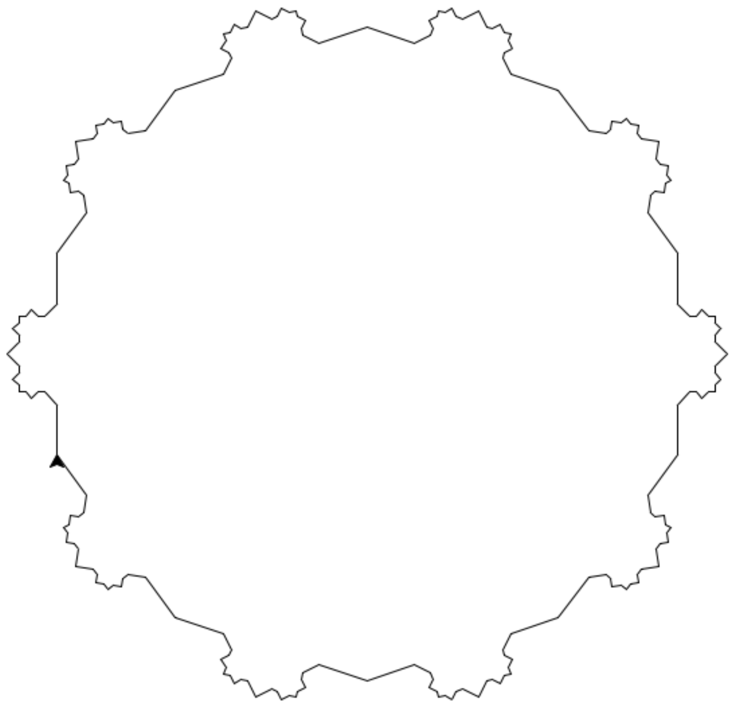


Figure 7. A Sample Drawing

Find below more sample drawings for your reference:

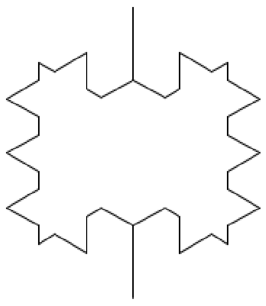


Figure 8. A sun star with 2 sides, side's length 200, level 4, and deviation angle 60.

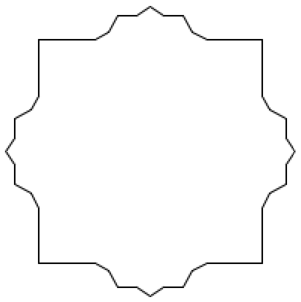


Figure 9. A sun star with 4 sides, side's length 150, level 3, and deviation angle 30.

3 Grading

The assignment grade is based on these factors:

- 20%: Attendance at problem-solving and results of problem-solving teamwork
- 10%: Proper user input (including order)
- 10%: Error checking and re-prompting
- 15%: Accurate drawing of one side of the polygon
- 15%: Accurate drawing of overall polygon (angles, lengths)
- 10%: Length calculation
- 10%: Good design practices
- 10%: The code follows the style guidelines on the course web site.

4 Submission

Submit your `sunstar.py` file to the MyCourses assignment before the due date.