Unit 4: Driving Features | **Application: Custom Polygons**

In this lesson, you will earn your Rover driver's license by designing a 'regular polygon' maker.

Objectives:

- Use **input()** statements to enter data for the number of vertices and the length of each side of a regular polygon
- Display lights along the sides and corners.

Now that you have successfully driven a square route and a pentagonal route, you are ready to take your Rover driving test.

Create a program in which you enter the *number of vertices* and the *length of each side* of a regular polygon, drive that route, and light up the sky with a dazzling array of colors along the way. In addition to simply moving Rover, your program will seek input from the user, drive the proper distance for each side, and calculate the proper angle to turn at each vertex.

1. Make a copy of your pentagon program from the last lesson. We name it RVPOLY. Your code should be similar to the image to the right. You will make some additions and changes to this code.

2. Before the **for** loop, write two **input** statements to enter the number of vertices (**n**) and the length of each side (**s**).

   n = …. *(an input statement)*

   s = …. *(another input statement)*

   Remember to use **int**(**input**(…)).
   It is *good* to use more informative variable names like **vertices** and **length**. Just be careful not to use Python reserved words.

3. There are three values in the loop statements that need to change:

   range(?), **forward(?)**, and **right(?)**

After editing those three arguments, **<Run>** your program, enter values for the input statements, and test your program.

Watch Rover carefully or insert a marker in the marker holder and draw the polygon on paper.

If you used an expression for the brightness of the LED along each side, then you will also have to adjust those LED statements to account for the change in the number of vertices. It would be better to use a variety of colors, not just red and green. Consider random colors. To ensure that the color values remain in the range (0..255) and do not cause an error, you can add (…) **% 255** to your color values expressions.