



Unit 4: Driving Features

Skill Builder 3: Custom Turns and a Polygon

In this lesson, you will use custom turning angles and have Rover drive along a pentagonal path, lighting up in two different colors: one along the sides and another around the corners.

Objectives:

- Controlling the amount of turning
- Use `left()` and `right()` with arguments
- Use colors around the pentagon

Your project in this lesson will make Rover drive a **pentagonal** path. The square was easy because you knew to turn 90 degrees. But for a pentagon, *you* must tell Rover how many degrees to turn at each vertex.

From your Geometry experience, can you determine how many degrees Rover needs to turn at each vertex? See the hint in the picture to the right.

In addition to the driving, you will also have Rover put on a light show. Use the color LED on Rover to display *intensities* of one color along the sides of the pentagon and another color at the vertices.

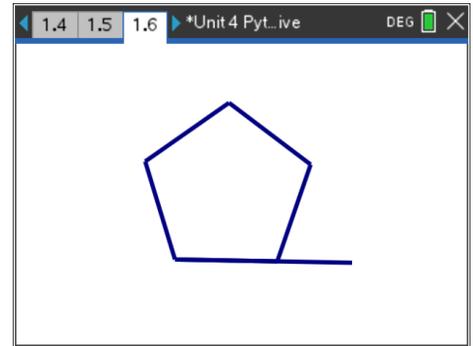
And, for extra fun, you can insert a marker in the marker holder and draw the pentagon – on paper but not on the table or the floor!

1. Begin by making a copy of the square-driving program from the last lesson.

2. Make *two* changes to convert from a square to a pentagon: there are 5 sides, not 4, and the angle to turn is 72 degrees:

```
for i in range(5)
```

```
    rv.left(72)
```



```

1.4 1.5 1.6 *Unit4 Pyt...ive DEG
=====
#square with red light on at the corners:
for i in range(4):
    rv.forward(1)
    rv.wait_until_done()
    rv.color_rgb(255,0,0)
    rv.left()
    rv.wait_until_done()
    rv.color_rgb(0,0,0)

```

```

1.7 1.8 1.9 *Unit4 Pyt...ive RAD
=====
# colorful pentagon:
for i in range(5):
    rv.color_rgb(0,255,0) #green sides
    rv.forward(1)
    rv.wait_until_done()
    rv.color_rgb(255,0,0) # red vertices
    rv.left(72)
    rv.wait_until_done()
    rv.color_rgb(0,0,0)

```



10 Minutes of Code - Python

TI-NSPIRE™ CX II WITH THE TI-INNOVATOR™ ROVER

UNIT 4: SKILL BUILDER 3

STUDENT ACTIVITY

3. Next, as Rover drives around the pentagon, the LED should start out with a dim color and gradually brighten. Use a *variable or expression* that makes the LED burn brighter at each side and each vertex. Your mathematics will come in handy.

Be careful to make sure that the color values stay between 0 and 255. You can use the variable `i` which varies from 0 to 4 to calculate a color value.

One possible expression: $50 + 50 * i$

Note that the last statement, `rv.color_rgb(0,0,0)`, is not indented (dedented).

This takes the statement out of the loop so that the LED is not turned off at each vertex, but only once at the end of the program. To turn the LED off, use *either* `rv.color_rgb(0,0,0)` or `rv.color_off()`. They do the same thing.



```
1.7 1.8 1.9 *Unit 4 Pyt_ive RAD 19/19
#=====
# colorful pentagon:
for i in range(5):
    rv.color_rgb(0, 50 + 50 * i, 0) # green sides
    rv.forward(1)
    rv.wait_until_done()
    rv.color_rgb(50 + 50 * i, 0, 0) # red vertices
    rv.left(72)
    rv.wait_until_done()
rv.color_rgb(0,0,0)
rv.color_off()
```