Putting the Fun in Functions with Python: Functions and Rotations

In this project, you will create rotational pieces of artwork. First, you will use functions, loops and exterior angles, to create regular polygons. Next, you will use translations to move your polygon to various locations on the coordinate plane. Lastly, you will use rotations, to create rotational pieces of artwork.

*This activity requires a TI-84 Plus CE Python calculator, and Turtle module. To install the Turtle module, https://education.ti.com/en/product-resources/turtle-module/ti84ce-python

Objectives:

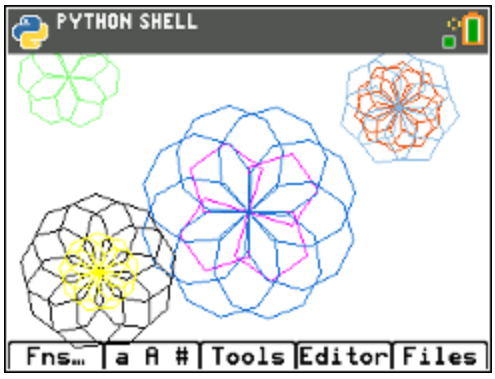
Programming Objectives:

- Define and use functions
- Use function notation to modularize code
- Use loops to repeat lines of code

- Use functions in a problem-solving situation
- Represent transformations in the plane
- Given a regular polygon, describe the rotations and reflections that carry it onto itself

Math Course Connections: Geometry- Translations and Rotations

For this project, you will write a program that uses functions, translations, and rotations to create works of art. You will write a program that lets you draw regular polygons anywhere on the screen. You will then use loops and rotations around a point to create symmetric art.



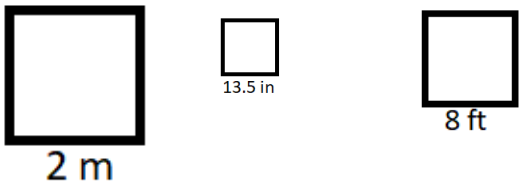
1. In math class you've used many functions. You may recall, for each input, a function has one and only one output.

   For example, the function to calculate and find the area of a square is:

   area(side) = side^2 or short hand a(s) = s^2

   The input side = 5 has exactly one output, 25.
   The input side = 7 has exactly one output, 49.

2. Use the function from step 1 to find the area for the following squares:
(not drawn to scale)



3. You might have used the equation f(c) = $\frac{9}{5}c + 32$ to convert degrees Celsius to degrees Fahrenheit.

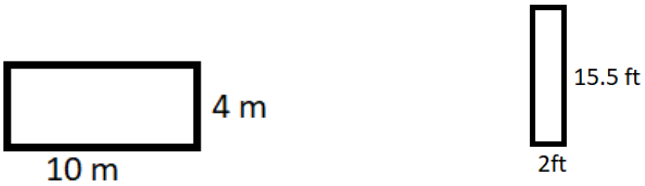
Use the function to complete the table below:

Celsius	Fahrenheit
0	
10	
20	
-5	

4. Some functions take more than one parameter.

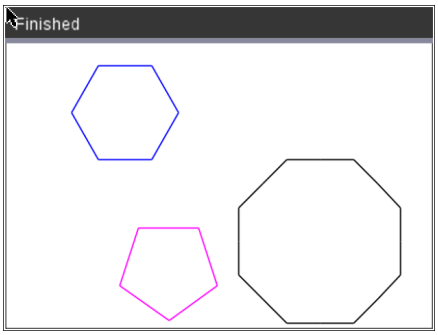
For example, the area of a rectangle is:
area(base, height) = base x height or a(b, h) = b x h

Use the function to find the area of the rectangles below:
(Not drawn to scale)



5. Computer programmers write functions to carry out repeated actions just like the math functions above. For this first activity, you will write a function to draw regular polygons with various dimensions.

   The picture on the right, has three different colored regular polygons, drawn using the same polygon function three different times.



6. There are a few key pieces of information you need to draw regular polygon. What do you think this information is?

   1.)

   2.)

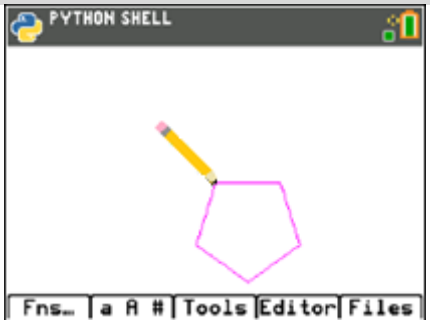
   3.)

   4.)

Teacher Tip: Possible answers might include: number of sides, length of sides, (x,y) location to start drawing, color. Student might also mention inter or exterior angle measurements.

7. How would you tell someone to draw a specific regular polygon?

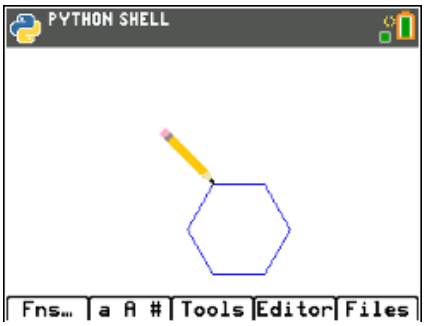
   If your pencil is at the origin (0,0), what would be the steps to draw the given pentagon? (The first step has been completed for you.)



   Steps:
   go forward 50 units
   ?
   ?
   ?

Teacher Tip: rotate 72 degrees, go forward 50 units, rotate 72 degrees, go forward 50 units, rotate 72 degrees, go forward 50 units, rotate 72 degrees, go forward 50 units

8. How would your steps listed in the previous pentagon example change if the graph was the octagon to the right?



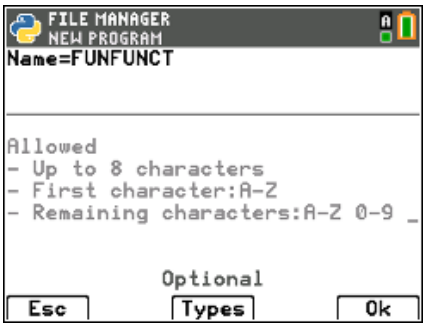
9. After completing the hexagon in part 8 and the pentagon in part 7, revisit your generic list in step 6.

   Add any additional items you deem necessary.

10. Now, let's write your program.

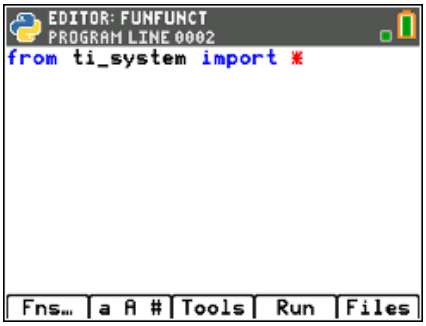
    Start a new program python project.
    [prgm] Python

    Name your program "FUNFUNCT".
    Select the type "Blank Program".

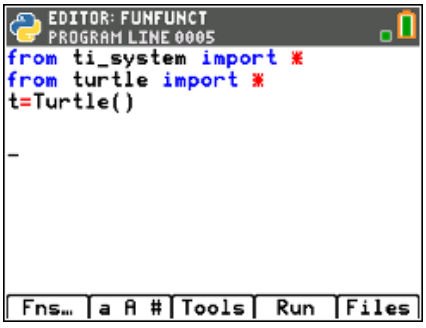


11. To draw in Python, you need the turtle library and they ti_sytem library.

    Fns > Modul > ti_system > from ti_system import *



    turtle library
    Fns > Modul > [Add – On] > from turtle import *



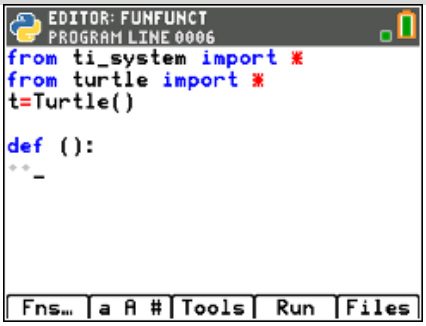
    *If turtle isn't an option, update your software.

Teacher Tip: Students should install the Turtle module prior to completing this activty.

12. In math class, functions usually start with f(x), g(x), area(s), area(b,h)…. You start with a name of a function and the argument (variables) it takes.

    To start a function in in python you say:
    def functionName(argument):

    To get a blank definition,
    Fns > Func > def

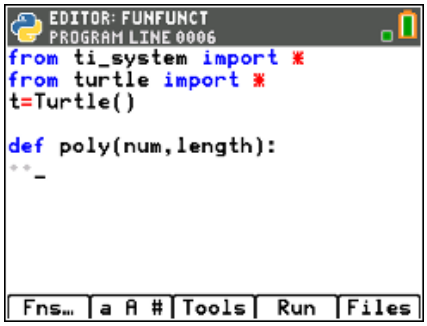


13. Look over your list from step 5.

    Does your list include number of sides and length of sides?

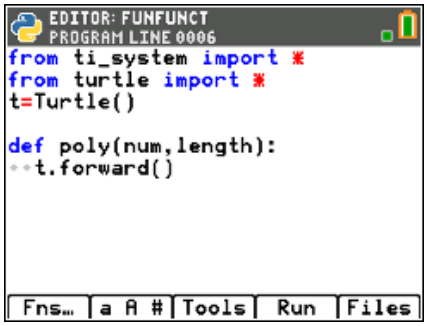
    Those will be the first two arguments for your function.

    Put num as the first argument and length as the second argument.
    def poly(num,length):

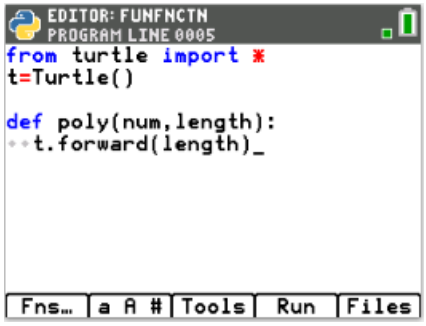


14. For all shapes, the turtle (pencil) needs to move forward. The command is **t.forward(distance).**

    Fns > Modl > turtle > Move > t.forward(distance)



15. Your function has an argument named length. That length will hold the distance the turtle should travel. Place length inside the t.forward() function parenthesis.



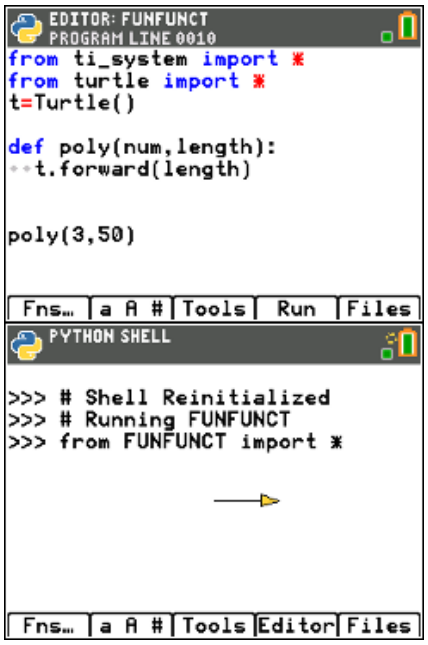
16. Now that your function has one command, lets use the function to draw.

    Go to the next line and remove the two diamonds. This will exit the function definition.

    Go down one more line and type poly(3,50).
    This will call (use) the poly function giving it num = 3 and length = 50.

    Execute your code **[Run]** by pressing [trace].

    Verify your turtle moved forward 50 units.

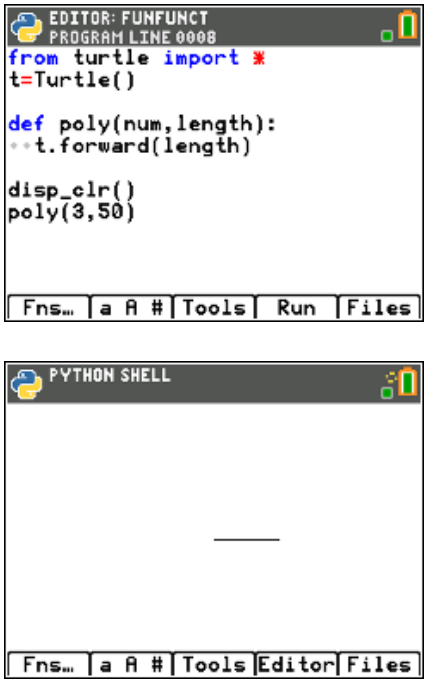


17. First, let's clear the three lines of code that are displayed in the shell.

    disp_clr()
    Fns > Modul > ti_system > disp_clr

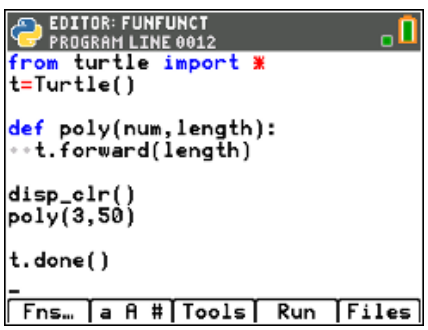
    Run the code. Verify the line segment display without the three lines of text.

    Your program still exits the turtle drawing screen after it draws.
    You will fix this in the next step.

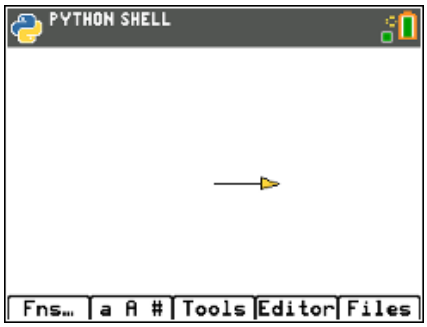


18. After your code poly(3,50) that draws the segment, add the command t.done(). This tells the computer it is done with the Turtle and adds a pause to your program. After adding this line, your drawing will stay on the screen until you press [clear] to exit.

    Fns > Modul > turtle > t.done()



Run your project. The line segment and turtle will stay on the screen until you press [clear].

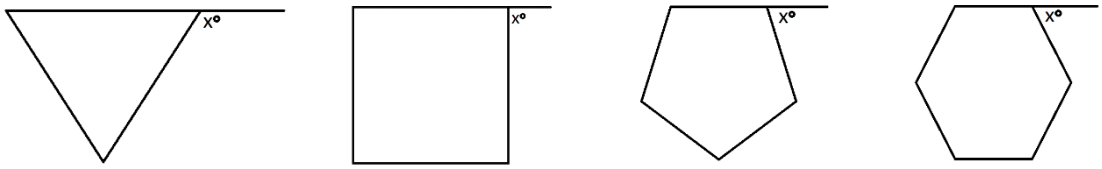


19. The length value 50 in t.forward(50) drew a straight line. Now you need to "turn" and draw the next side.

    How far should you "turn"?

20. You are correct if you said that depends on the shape. Each regular polygon has a different exterior angle.

    Find the exterior angle for each shape below.

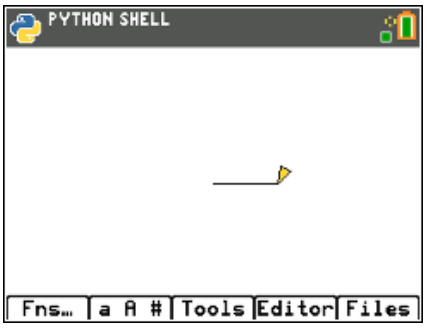


21. What is the generic formula for finding the exterior angle for a regular polygon?

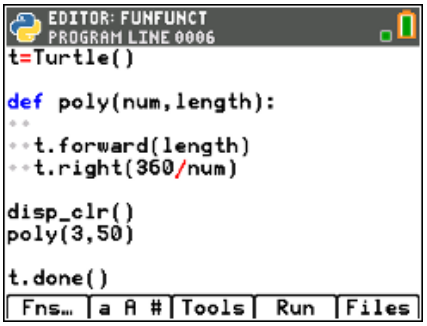
    Add a right turn to your code using your formula. Use the variable num in your formula since that is the argument in your function.

    Run your program.

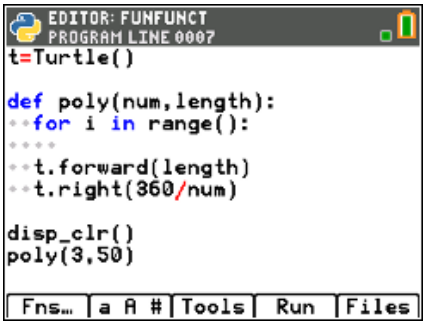
The line poly(3,50) stores 3 in the num variable and 50 in the length. Your turtle should have moved forward 50 then rotated 120°.



22. Now add a loop so this process will happen **num** times.

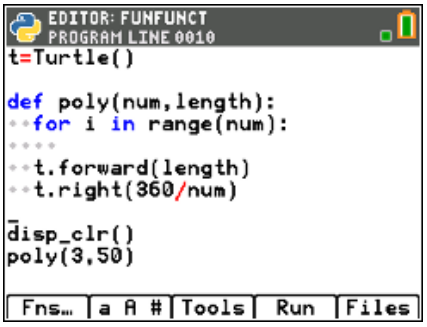


Add a **for loop**
Fns > Ctl > for index in range(size)

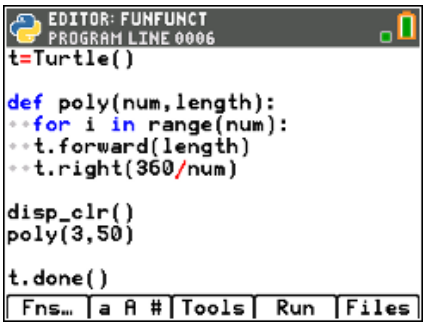


23. Place num inside the range() function.

This will ensure the loop happens num times.
If num is a 3 it will happen three times.
If num is a 4 it will happen four times.

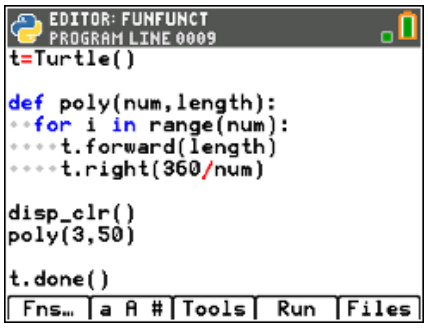


24. Remove the blank line.

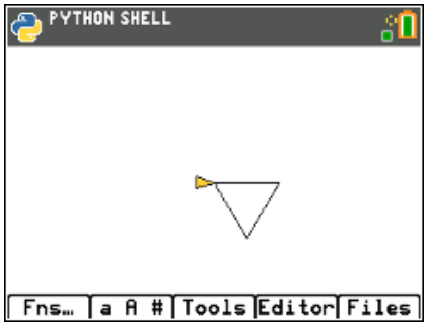


To indent the two lines, you can either add two spaces using the space key, [2nd][0].

Or, you can choose to put your cursor on each line and select
[Tools] indent

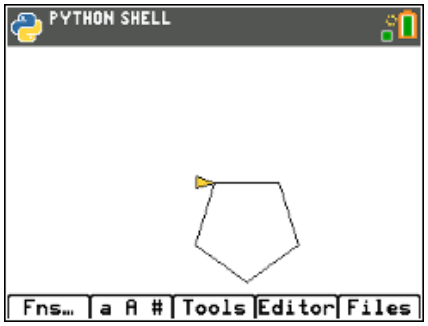


25. Execute your program



The line poly(3,50) passes the num 3 and length 50 to the code.
The **for loop** makes the code happen num (3) times.
The turtle moves forward length (50)
The turtle rotates right 360/num = 360/3 = 120°

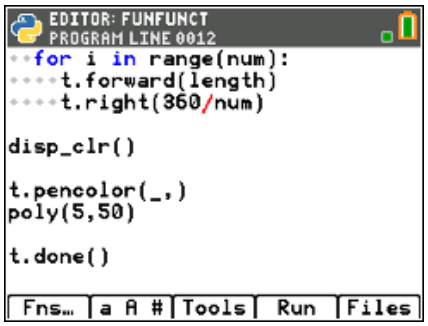
26. To draw a pentagon instead of a hexagon, change the poly(3,50) to a poly(5,50)



27. Now to add some color.

The command t.pencolor will set the pen color.

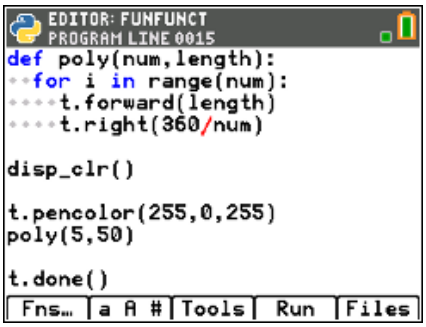
Fns > Modul > turtle > Pen > t.pencolor(r,g,b)



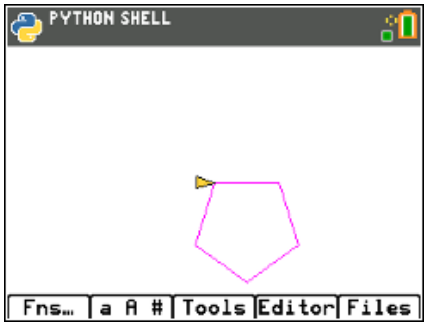
You have 256^3 or 16194277 color choices!
t.pencolor(r,g,b) allows you to enter integers from 0 – 255 for each parameters. r-red, g-green, b-blue

Try these two different values for red.
t.pencolor(255,0,0)- Red with the maximum value for red 255
t.pencolor(100,0,0)- Red set to 100.
What appears to be the difference?

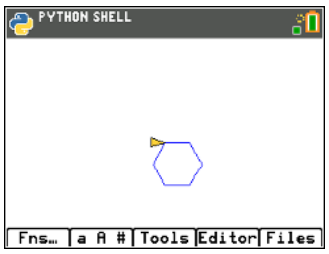
One way to get magenta is to use
t.pencolor(255,0,255)



Pick a pen color and draw your pentagon.
The example to the right uses magenta to draw the pentagon.

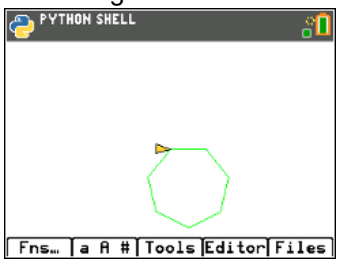


28. Can you modify your code to draw the following:



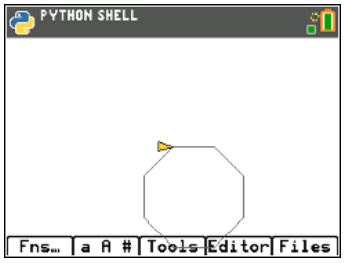
Hexagon Length 25
Color: blue

function call used:
t.pencolor(, ,)
poly(,)



Heptagon Length 35
Color: green

function call used:
t.pencolor(, ,)
poly(,)

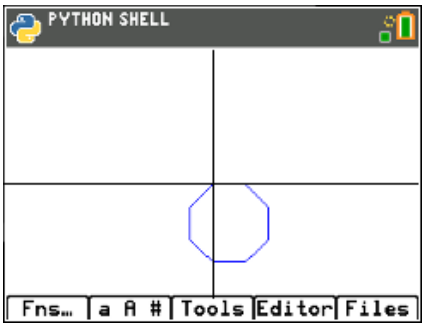


Octagon Length 40
Color: gray

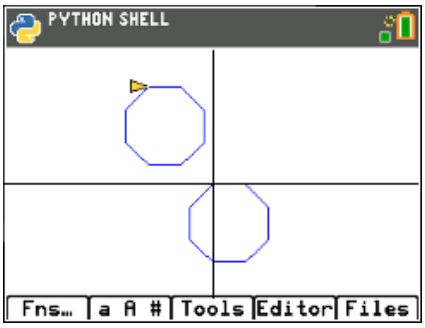
function call used:
t.pencolor(, ,)
poly(,)

29. Currently, your function draws all the regular polygons with the starting point at (0,0).

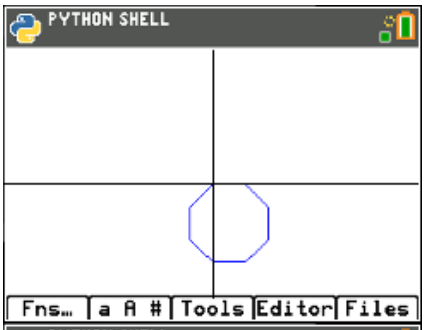
Imagine a coordinate grid overlayed on top of your picture.

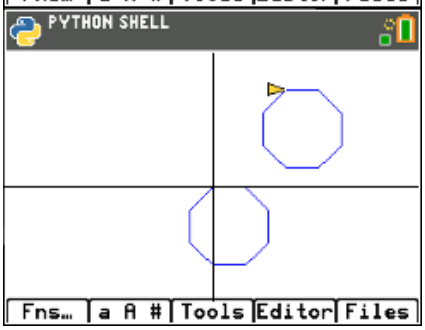


Mathematically, describe the transformation that would result in drawing the octagon in the new location.



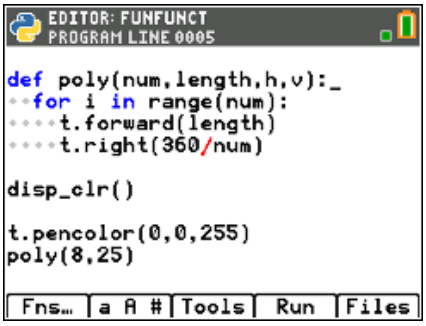
30. Mathematically, describe the transformation that would result in drawing the hexagon in the new location.





31. Adding a horizontal and vertical **translation** is simple in Python.

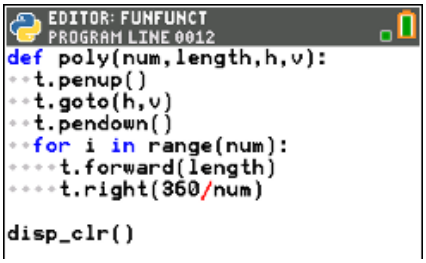
    To start, add a horizontal and vertical argument to your definition. For ease of typing, let's use h and v.



32. You will need to pick up the pencil so it doesn't draw.
    Preform a horizontal and vertical translation for the starting point.
    Put the pencil down.

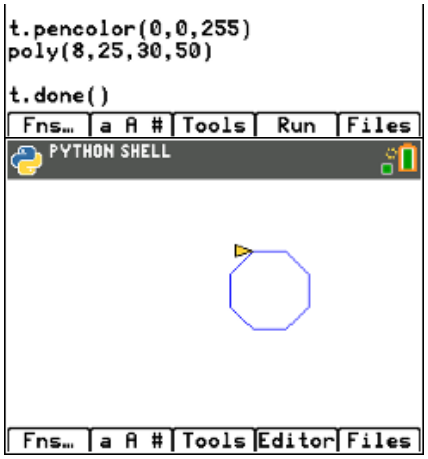
    Fns > Modul > turtle > Pen > t.penup()

    Fns > Modul > turtle > Move > t.goto(x,y)
    Replace the x,y templates with h and v



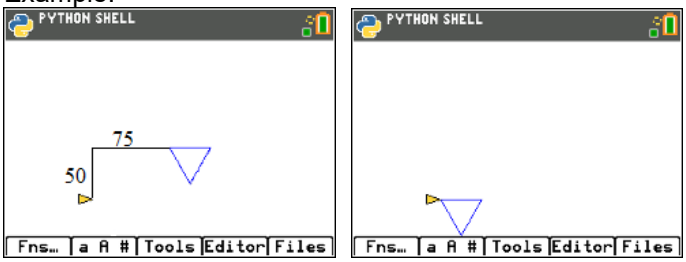
Fns > Modul > turtle > Pen > t.penup()

Lastly, in your function call give it a horizontal and vertical translation. The code on the right preformed a 30 unit horizontal translation and a 50 unit vertical translation.



33. Use your function to draw **translated** shape.

Example:



Math Description:

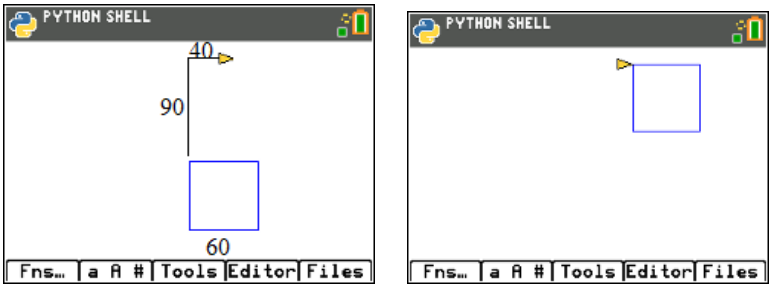
Horizontal Translation -75 units

Vertical Translation -50 units

Function call:

poly(3,70,-75,-50)

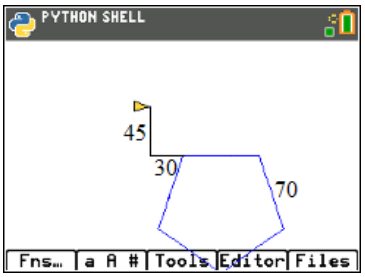
Practice #1

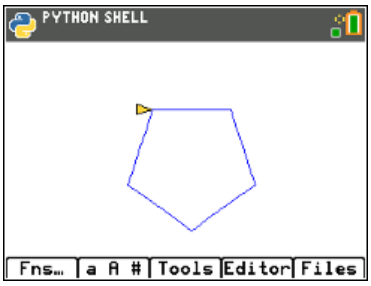


Math Description:

Function call:

Practice #2

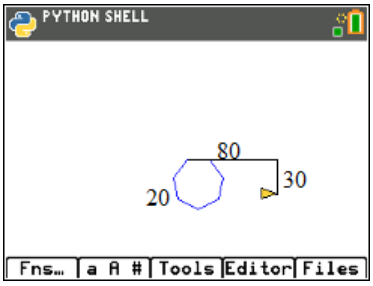


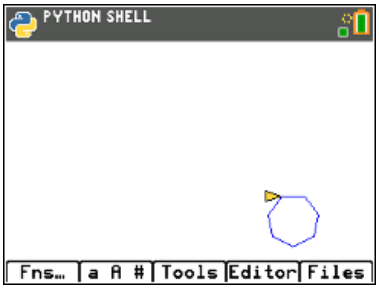


Math Description:

Function call:

Practice #3

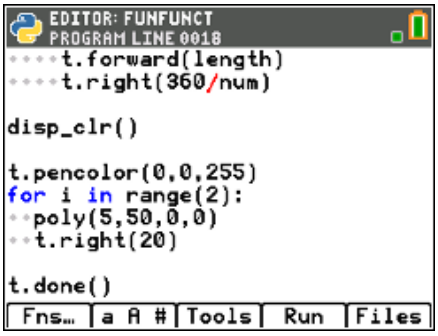




Math Description:

Function call:

34. Now let's add a rotation.



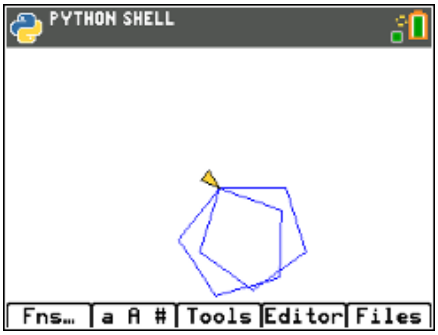
If you change your function call from one line, such as

```
poly(5,50,0,0)
```

to

```
for i in range(2):
    poly(5,50,0,0)
    t.right(20)
```

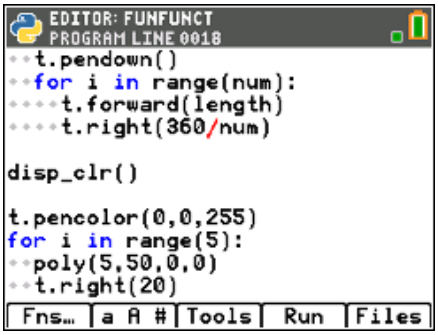
The code will draw the polygon, rotate right 20°, then repeat the code for a second polygon.

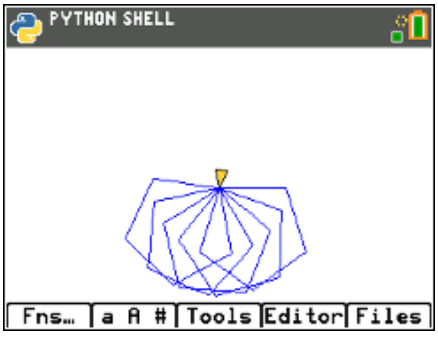


**for
Fns > Ctl > for index in range(size):

**Rotate right
Fns > Modul > turtle > Move > t.right(angle)

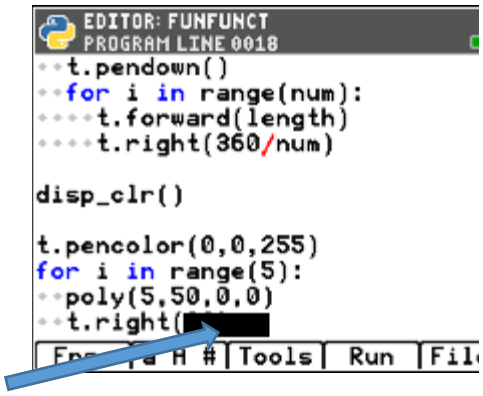
35. Modify your code to draw 5 pentagons each rotated 20 additional degrees from the previous one.





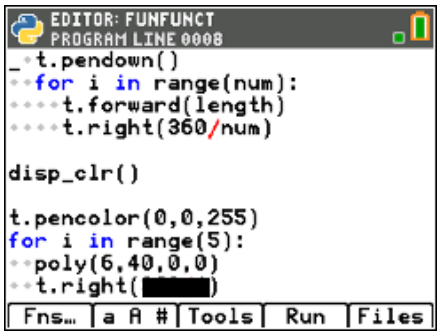
36. Modify the right rotation degree so the 4th copy of the pentagon (5th pentagon) "maps" or traces precisely over the original pentagon.

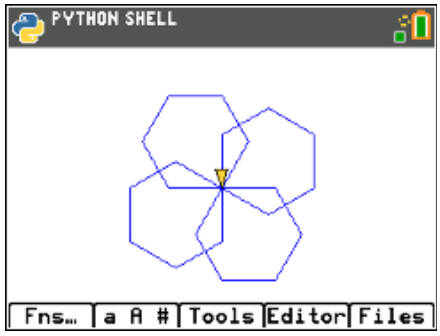
Original Copy 1 Copy 2 Copy 3 Copy 4



Record your answer here:_____

37. What angle of rotation allows a hexagon to rotate 4 times and "map" back onto itself?

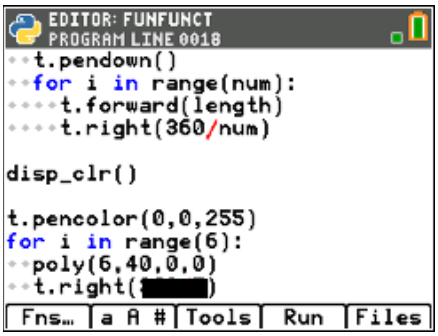


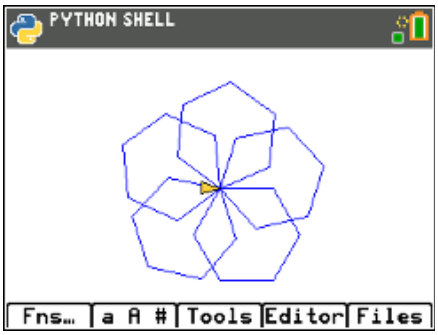


38. Do you think this same angle rotation would result in the 4th copy of an octagon "mapping" on itself? Explain your thinking.

39.

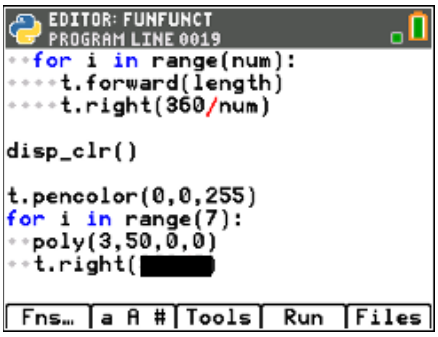
How many degrees should you rotate if you want to draw 5 hexagons before the 6th "maps" onto the original?

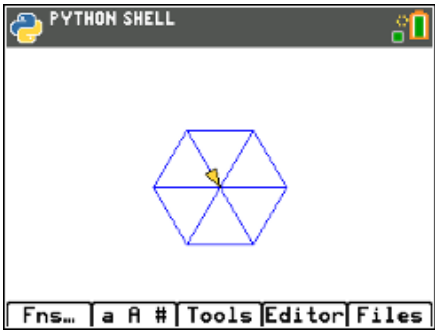




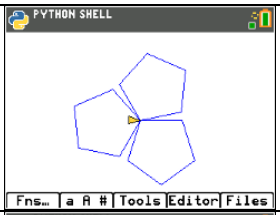
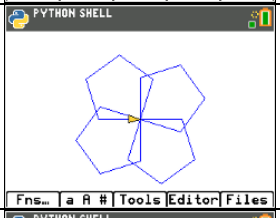
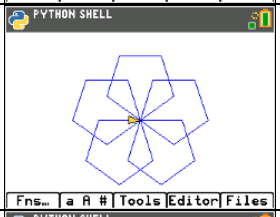
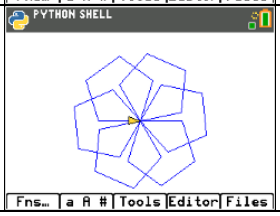
40.

How many degrees should your rotate if you want to draw 6 triangles before the 7th "maps" onto the original?





41. In general, the math to determine the number of degrees to rotate to “map” onto the original.

#of shapes drawn before a “map”		Math
3		
4		
5		
6		
N		

42. Now you can use your polygon function to make rotational art.

Rotated pentagon Rotated nonagon Geometric Art

For each new rotated polygon, you pick the

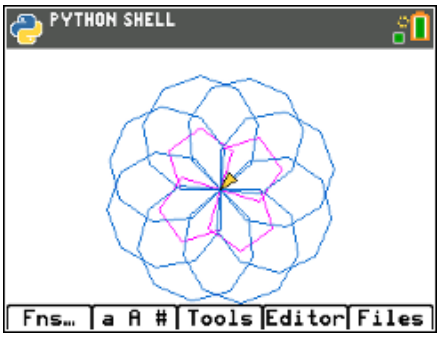
- color
- add a for loop
  - use the poly function
  - rotate

```
EDITOR: FUNFUNCT
PROGRAM LINE 0001
from ti_system import *
from turtle import *
t=Turtle()
t.speed(0)
def poly(num,length,h,v):
    t.penup()
    t.goto(h,v)
    t.pendown()
    for i in range(num):
        t.forward(length)
        t.right(360/num)
```

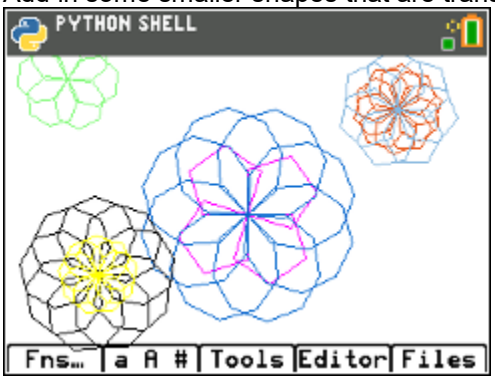
```
disp_clr()
t.pencolor(255,0,255)
for i in range(5):
    poly(5,30,0,0)
    t.right(360/4)

t.pencolor(0,100,200)
for i in range(9):
    poly(9,30,0,0)
    t.right(360/8)
t.done()
Fns… | a A # | Tools | Run | Files
```

Hint:
To speed up the drawing, add t.speed(0) before you draw.
Menu > Modul > turtle > Setting > t.speed

To hide the turtle
Menu > Modul > turtl > Setting > t.hideturtle



Add in some smaller shapes that are translated and rotated:



What kind of artwork can you make?

Can you make artwork that has at least three different rotational works of art centered at different points?

```
from ti_system import *
from turtle import *
t=Turtle()

t.speed(0)

def poly(num,length,h,v):
  t.penup()
  t.goto(h,v)
  t.pendown()
  for i in range(num):
    t.forward(length)
    t.right(360/num)

disp_clr()

t.pencolor(255,0,255)
for i in range(4):
  poly(5,30,0,0)
```

```
  t.right(360/4)

t.pencolor(0,100,200)
for i in range(8):
  poly(9,25,0,0)
  t.right(360/8)

t.pencolor(230,80,20)
for i in range(10):
  poly(6,15,100,70)
  t.right(360/10)

t.pencolor(130,250,120)
for i in range(6):
  poly(7,15,-120,90)
  t.right(360/6)

t.pencolor(130,180,220)
for i in range(7):
  poly(6,20,100,70)
  t.right(360/7)

t.pencolor(0,0,0)
for i in range(9):
  poly(8,20,-100,-40)
  t.right(360/9)

t.pencolor(255,255,0)
for i in range(9):
  poly(8,10,-100,-40)
  t.right(360/9)

t.hideturtle()

t.done()
```