Lesson 13: Functions with Turtles

In the previous lesson we learned how to create our own Python functions. This lesson shows how Python functions can be useful in creating drawings using the turtle module. It will also give you good practice with functions and hopefully deepen your understanding.

A Triangle Drawing Function

How do you draw a triangle using the turtle module? Using our knowledge from "Lesson 6: Turtle Graphics", we would do it like this (don't bother typing this, we'll do it a better way a little later):

```
>>> import cpif.turtle as t
>>> t.forward(100)
>>> t.left(120)
>>> t.forward(100)
>>> t.left(120)
>>> t.forward(100)
>>> t.left(120)
```

Using the greater knowledge and wisdom we gained in "Lesson 8: Looping with Turtles", we would never repeat the repetitious code like we did above. Instead, we'd draw a triangle using a `for` loop, like this:

```
>>> import cpif.turtle as t # only if you haven't typed this yet
>>> for i in range(3):
          t.forward(100)
          t.left(120)
```

But now I'd like to show you an even more powerful way to draw shapes, by writing a function and then calling it later. Effectively, you can draw any shape with just one short command, once you write a function for it.

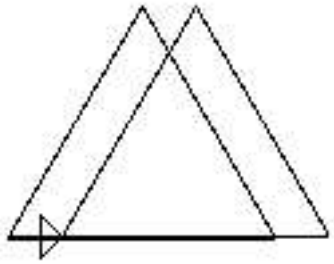
Try defining a triangle-drawing function:

```
>>> import cpif.turtle as t # only if you haven't typed this yet
>>> def tri():
          for i in range(3):
                    t.forward(100)
                    t.left(120)
```

Now it is as if you had added a new command to the Python language called `tri()`. You can draw a triangle any time you like by calling that function. Let's clear the window, draw a triangle, move the turtle a little, and draw another triangle:

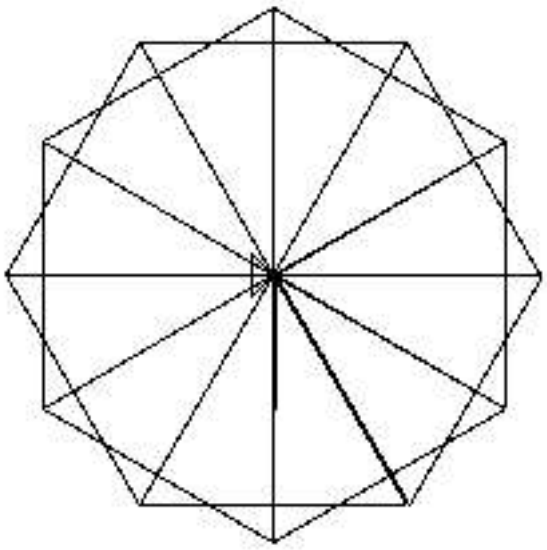
```
>>> t.reset()
```

```
>>> tri()
>>> t.forward(20)
>>> tri()
```



Just to show you how powerful this can be, let's try calling our triangle function inside a `for` loop, rotating it a little each time:

```
>>> t.reset()
>>> for i in range(12):
          t.right(360.0 / 12)
          tri()
```



(In case you can't tell, I really like this stuff!)

Bigger and More Colorful: Functions with Parameters

Our triangle drawing function is fun, but it always draws triangles that are the same size and color. Wouldn't it be nice if we could tell it what size and color of triangle we want? That's what parameters are for. As described in the previous lesson, parameters are additional pieces of information that tell a function how to do it's work. We'll create a version of the `tri()` function that takes color and size parameters.

From now on in this lesson we'll do the examples in the Editor window instead of the Python Shell window, so we can save our work and come back to it later.

In the Python Shell window, select the File menu, then “New Window”. A new Editor window will appear. In this new editor window, select the File menu, then “Save”. Save the (empty) file as **`tri.py`** in your **`MyPrograms`** directory. This file is where you will put your commands for this part of the lesson.

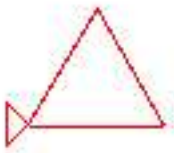
In your **`tri.py`** window type the following commands:

```
import cpif.turtle as t


def tri(color, size):
    t.color(color)
    for i in range(3):
        t.forward(size)
        t.left(120)


tri('red', 50)
```

Press Ctrl-S to save your program, and F5 to run it. You should see this:

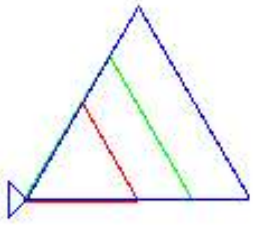


This little program defined the `tri()` function with color and size parameters. Then the last line in the program called the `tri()` function and gave it specific values for its parameters. The first parameter value `'red'` went to the first parameter, `color`, and the second parameter value 50 went to the second parameter, `size`. That's what told your `tri()` function what color to use for the triangle and how long its sides should be.

You can see the benefit of function parameters by creating several triangles of different shapes and sizes. Add the following two lines to the end of your **`tri.py`** program:

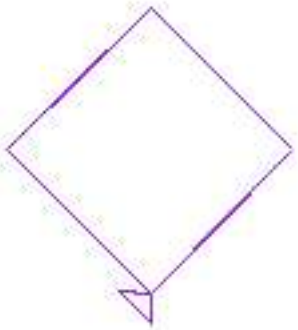
```
tri('green', 75)
tri('blue', 100)
```

Press Ctrl-S to save the program, and F5 to run it. Here is what it draws:



Quiz 13

1. Create a program called **`squ.py`**. In your **`squ.py`** program, define a function called `squ()` that takes three parameters: color, size, and angle, in that order. The `squ()` function should draw a square. It should use the parameters to tell it what color and size to make the square, and what angle it should be turned to. (Hint: use `t.setheading(angle)` to set the starting turtle angle.) The last line of your **`squ.py`** program should call your `squ()` function like this: `squ('purple', 75, 45)`. The result should be a picture that looks like this:



2. Now add three more lines to the end of your **`squ.py`** program that draw three more squares, each with a different color, size, and angle.