Lesson 9: Drawing a Flower

In this lesson, we'll learn how to use loops and functions to copy shapes.

By making multiple copies of a petal shape, we'll be able to draw a flower!

PART 1: Setting up the Turtle

As with the other turtle programs, we first need to set up a turtle before we can use it.

Type the code below into a new program:

```
import turtle
myT = turtle.Turtle()
____________    #set turtle speed to fastest
turtle.done()
```

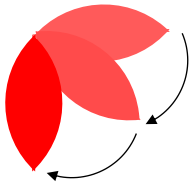
This time we'll call our turtle `myT`. The name of a variable doesn't matter so long as we refer to the correct one when using its functions.

PART 2: Defining a Petal function

Next, we'll begin defining functions that we can use to draw the shapes for our flower.

We'll start by adding a petal function, which we can call multiple times to draw petals.

Each time we call the petal function, it will draw one petal, and afterwards we'll rotate the turtle a little bit and call the function again.



Try adding this code above `turtle.done()`:

```
myT.speed(0)
def petal(myT):
    myT.fillcolor("red")
    startdirection = myT.heading()
    ___________   #set pendown to draw visible lines
turtle.done()
```

We define a function to create a petal and then call the `fillcolor()` function on turtle myT to set the color of the petal to red.

We create a variable called start direction and use the `heading()` which states the direction the turtle is facing, in degrees.

We've still got more to do with our `petal(myT)` function in order to draw the petal.

Add the following code to the `petal(myT)` function:

```
def petal(mT):
    .  .  .
    myT.pendown()
    myT.begin_fill()
    myT.lt(32)
    for x in range(5):    #Draw one edge of the petal
        myT.fd(20)
        myT.rt(16)
    myT.rt(100) # Position turtle to begin drawing other edge of petal
    _____________: #Draw the other edge of the petal
        __________
        __________
    ___________    #Bring penup
    myT.end_fill()
    myT.setheading(startdirection)
petal(myT)          #This calls the petal function
turtle.done()
```

Here, we're using my`T.begin_fill()`, which tells our turtle to start filling a shape. Our turtle will only fill in the shape when we call my`T.end_fill()` after moving it.

After this we use a sequence of loops, and turn/ forward commands to make a petal.

Next, we lift the pen, and call my`T.end_fill()` to fill in the shape we drew.
This way, we can reposition the turtle after drawing the petal, and not worry about our turtle drawing extra lines or shapes then.

We'll use the `setheading()` function to change the direction our turtle is facing. This way, we can rotate the turtle back to the direction it was facing before it started drawing this petal (which is stored in the `startdirection` variable).

Last, we use the `petal(myT)` function to draw a single petal. **Try testing your code!**

PART 3: Creating a Circle Function

Here, we'll use a loop to create a circle, like we did in the previous lesson.
This circle will go in the center of our flower, after we draw the petals.

Before we create the circle, we'll want to move the circle to a specific position.

Try typing this code <u>above</u> `turtle.done()`:

```
. . .
def petal(myT):
    . . .
    myT.setheading(startdirection)
def circle(myT):
    ____________________     #sets the fill color to yellow
    ____________________          #lifts the pen
    myT.setheading(0)             #faces the turtle to the right
    myT.goto(-1.75, 20)           #moves turtle to this point
    _________                   #puts pen down
    _________             #start filling shape
    for x in range(36):     #loop will draw the circle
        myT.forward(3.5)
        myT.right(__) #rotate turtle amount to right before next draw
    myT.end_fill()      #completes filling shape
    myT.penup()
petal(myT)
circle(myT)    #Be sure to add this line which will run the circle function
turtle.done()
```

Now, if you test your program, it should draw a single petal, and then the center of the flower on top of the petal.

PART 4: Using the Drawing Functions

Add the following code outside of the functions, and <u>above</u> `turtle.done()`:

```
import turtle
myT = turtle.Turtle()
myT.speed(0)
def petal(myT):
    .  .  .
    myT.setheading(startdirection)
def circle(myT):
    .  .  .
    myT.penup()
myT.home()
myT.rt(22.5)   #Position turtle to begin making a petal
for p in range(_):  #Make 8 petals
    petal(myT)     # Indent this line from what you previously typed
    myT.__(__)   #Rotate turtle 45 degrees to right after each petal
circle(myT)        # You have already typed this – don't re-type
myT.hideturtle()
turtle.done()
```

First, we call `myT.home()` to set the turtle to the default position on the screen, and rotate it slightly.

After that, we call `petal(myT)` 8 times, using a `for` loop and turning the turtle each loop. Since the `petal()` function we defined asks for a turtle object, we pass `myT` when we call the function. This lets `petal()` know what turtle to use when it tries to draw.

After this, we call the `circle(myT)` function, also passing the same turtle object. We call the circle function afterwards so that it draws on top of all the petals.

Using `Turtle`, any shapes drawn later in the code will appear on top.

Lastly, we hide the turtle so that it's not on top of our flower drawing.

Now, if you test your program, you should see your turtle draw the whole flower!

PART 5: BONUS: Adding a stem

Now that we've made the rest of the flower, you can try to add a stem as well.

There are a few ways to do this, so we'll let you decide how to make the stem function. You could draw a stem as a rectangle, or as a curved line, for instance

Here's a list of movement functions which you can use in your stem function:

`myT.forward(p)`	`myT.fd(p)`	Moves the turtle forwards **p** pixels
`myT.backward(p)`	`myT.bk(p)`	Moves the turtle backwards **p** pixels
`myT.left(d)`	`myT.lt(d)`	Turns the turtle left by **d** degrees.
`myT.right(d)`	`myT.rt(d)`	Turns the turtle right by **d** degrees.
`myT.goto(x,y)`	-	Moves the turtle to position **(x, y)**
`myT.setheading(h)`	-	Turns the turtle towards direction **h** degrees
`myT.home()`	-	Places the turtle at its original position.

Try using some of these functions to draw a stem for your flower!