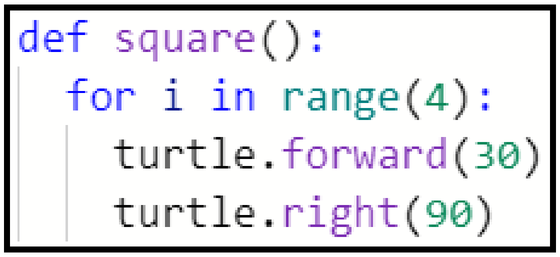
Lesson 3 - Subroutines/Functions

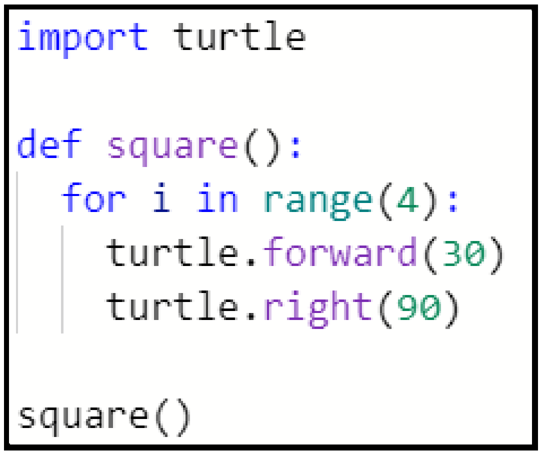
In this lesson we are going to learn about subroutines. Subroutines or functions allow us to break our code down into sections. These sections can then be reused elsewhere in our code and can save us time.

A subroutine starts with **def**, which stands for define. This tells the program that we are defining a subroutine. This is followed by the **name** of the subroutine, **2 brackets** and a **colon**.



```
def square():
  for i in range(4):
    turtle.forward(30)
    turtle.right(90)
```

The subroutine above will draw a square. To run the subroutine or “call it” we simply type the name of the subroutine in our program.

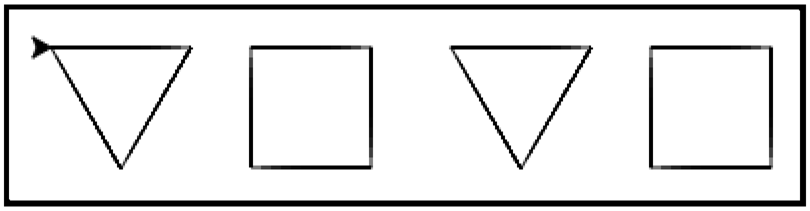


```
import turtle

def square():
  for i in range(4):
    turtle.forward(30)
    turtle.right(90)

square()
```

The full program.

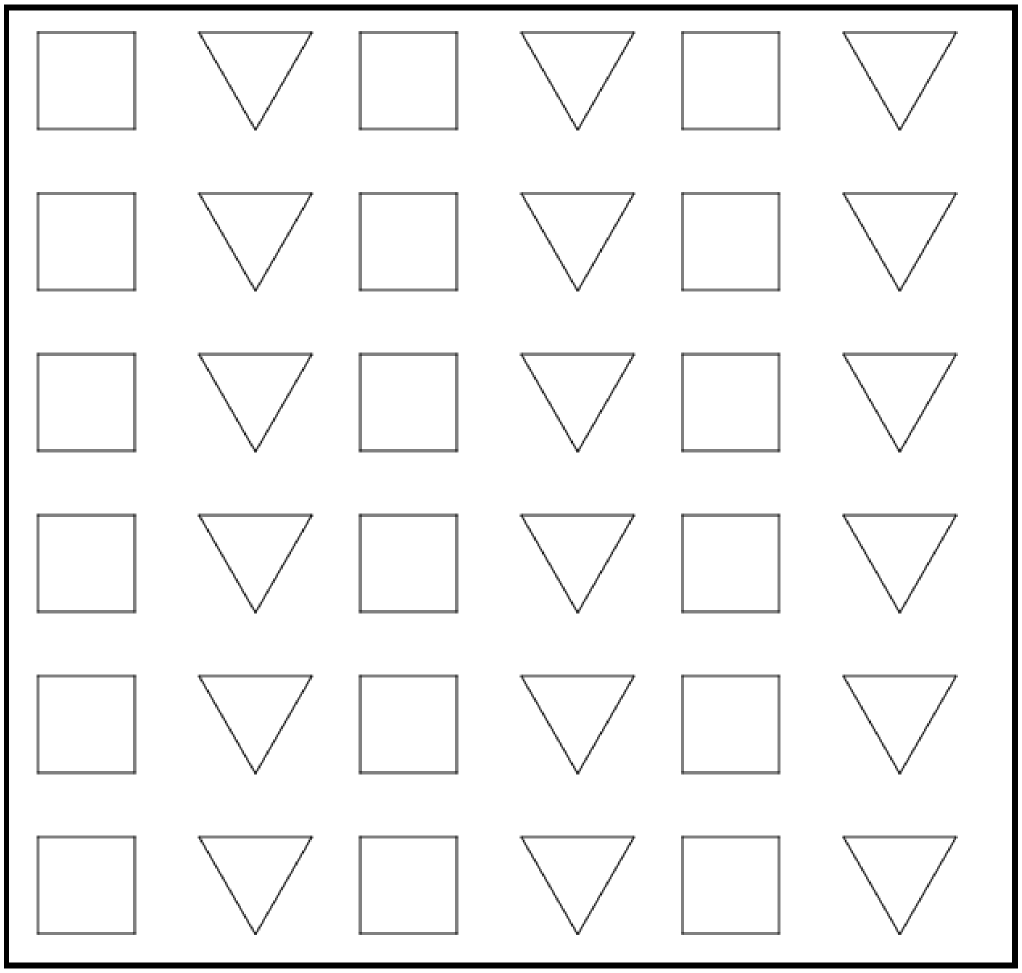
Challenge 1

- Run the program and ensure the subroutine works.
- Create a new subroutine to draw a triangle.
- Use both subroutines to draw two squares and two triangles in the pattern below.
- Use a for loop to draw this pattern



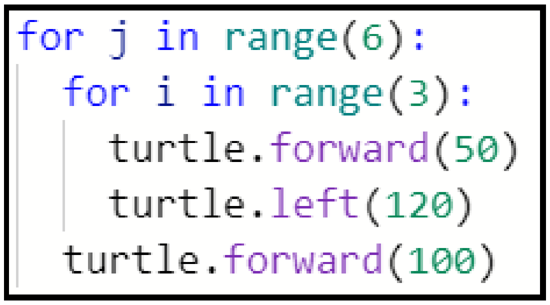
Challenge 2

Now that we have got a basic pattern done, we want to add to this pattern and create a pattern similar to the picture below.



What do we need?

- Create a subroutine to move the turtle to the top left corner of the screen.
- Create a subroutine to move to the position of the next shape in the pattern.
    - We will need to use **setheading(), penup(), pendown()** and **forward()**.
- Create a subroutine to move to the next row in the pattern.
    - We will move down the screen and all the way back to the left of the screen.
- Use a nested for loop to easily draw all the shapes. Below is a reminder of roughly what a nested for loop should look like.

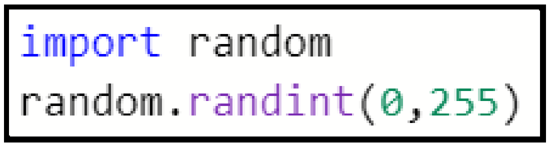


```
for j in range(6):
  for i in range(3):
    turtle.forward(50)
    turtle.left(120)
  turtle.forward(100)
```

Random Colours

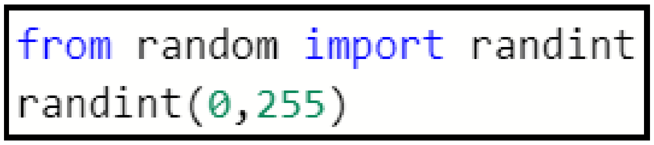
Now that we have all of our shapes, we are going to look at how to colour in each of these shapes a random colour.

To this we need to import the random library and use the randint function from it. There are two different methods of doing this.



```
import random
random.randint(0,255)
```

In this method we are importing the whole random library and then we specify that we want to use the function randint from the random library. Then we put the smallest number we want and the biggest number.

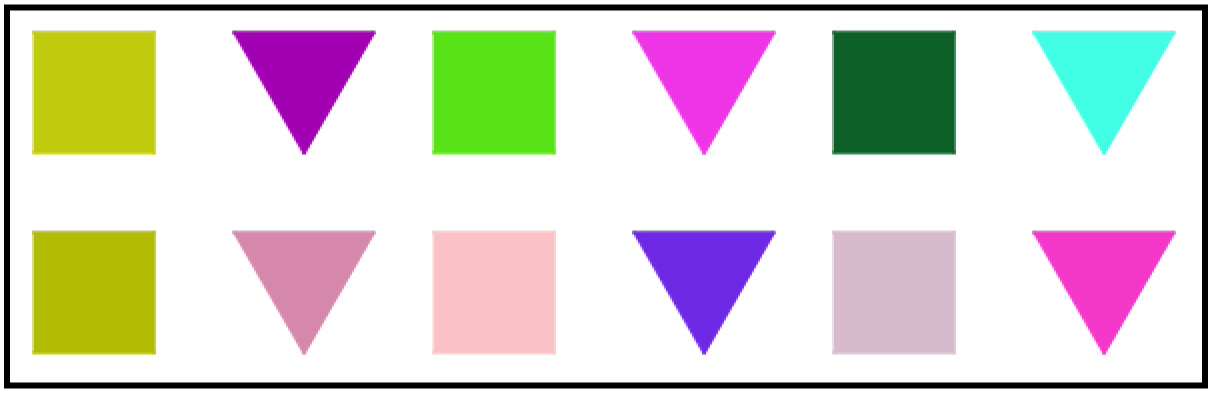


```
from random import randint
randint(0,255)
```

In this method we are only importing the randint function. Therefore our program only needs the name of the function and the smallest number we want and the biggest number we want.

Challenge 3

- Use randint(), color(), begin_fill() and end_fill() to color each of the shapes a different colour.

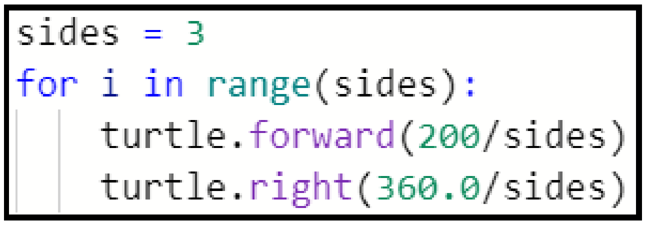


Passing Arguments

Although we now have subroutines to draw triangles and squares, we still don’t have any easy way of drawing other shapes, like pentagons and octagons.

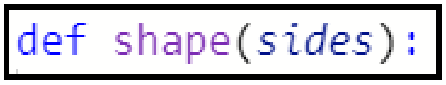
The subroutines that we wrote only do one thing and if we want to change it, we need to change the code completely.

Last week we learned about variables and how to use them to draw any regular shape we wanted. The code for this is below. All we needed to do to change the shape was to change the value of the sides variable.



```
sides = 3
for i in range(sides):
    turtle.forward(200/sides)
    turtle.right(360.0/sides)
```

We are now going to turn this into a subroutine and we will pass in the argument for the number of sides. The first line of the subroutine is below, the rest of the subroutine will look similar to the code above.



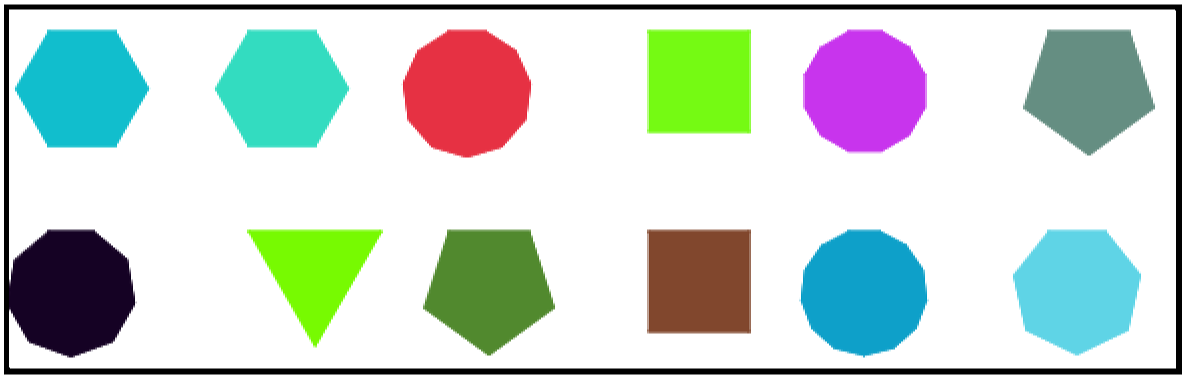
```
def shape(sides):
```

When we want to call the shape subroutine, we need to give it a number. It will look like this, shape(5).

Note: You can create subroutines that take more than a single number.

Challenge 4

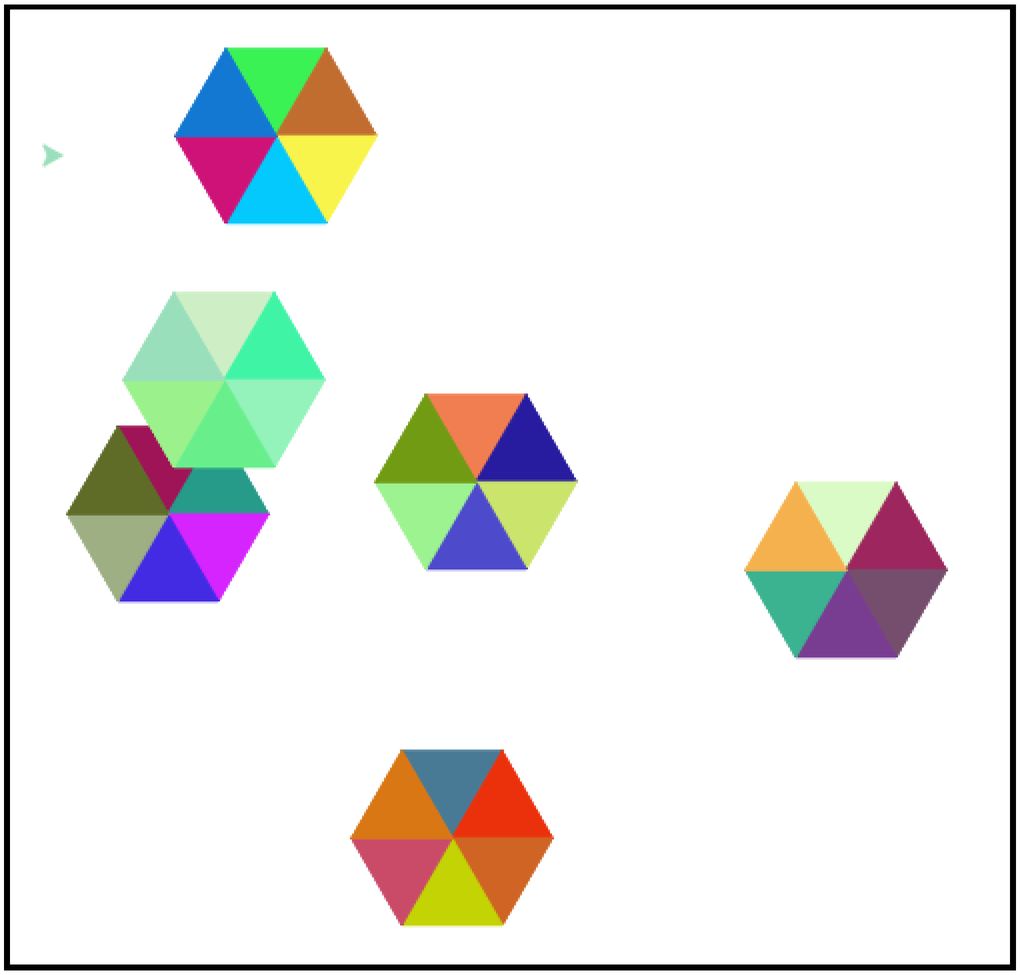
- Create a subroutine to draw regular shapes of any number of sides. Make sure that they are all roughly the same size.
- Using the randint function, change your code from the previous part to draw random shapes in a pattern like below.



Challenge 5
In a new program we are going to create the picture below. We are drawing the patterned hexagons 6 times and in random places.

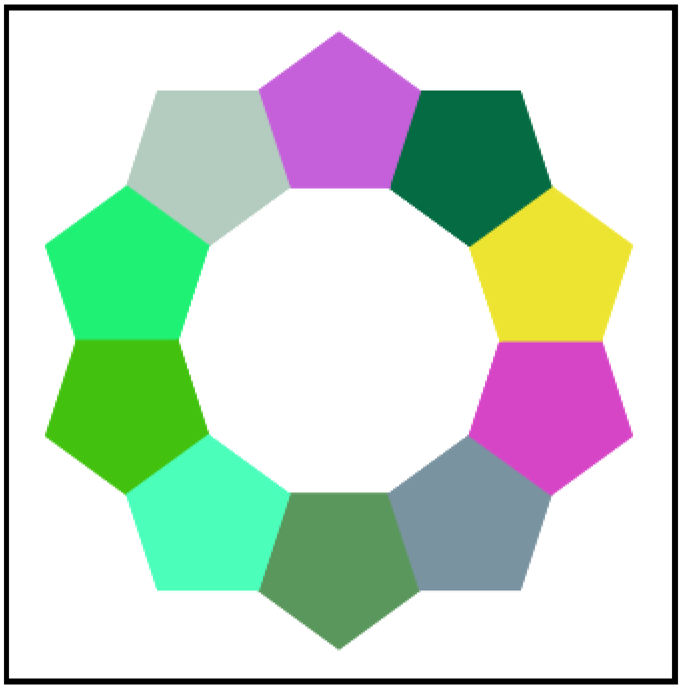
Tasks:

- Create a subroutine that will give us a random colour.
- Create a subroutine that will draw a triangle.
- Create a subroutine that will draw the hexagon pattern. You will need to use the triangle subroutine and it may be helpful to make the pattern without the subroutine or for loop first.
- Add code to move the turtle to random position after drawing each pattern.



Challenge 6

In a new program create the pattern below. This will be quite similar to the triangular pattern made in challenge 1. Reuse code from previous lessons where appropriate.



Challenge 7

In a new program create a chess board using what you have learned in this and previous lessons. Reuse code and create subroutines where applicable.