Turtley Amazing

In this resource you will take your first steps with the programming language Python

Python

Step 1 What you will make

In this resource you will take your first steps with the programming language Python to draw shapes, patterns, and spirals. You will use a module named Turtle. Along the way you will learn how to think in sequences, and use loops to repeat a sequence. This is a great stepping stone from a visual programming language like Scratch to the text-based environment of Python.

What you will learn

By making patterns with Python code you will learn:

- To take your first steps with the Python programming language
- How to draw lines with Python Turtle
- How to make turns
- How to change the pen colour
- To use loops to repeat some instructions and create shapes
- To use more loops to create spiral patterns

This resource covers elements from the following strands of the **Raspberry Pi Digital Making Curriculum** (https://www.raspberrypi.org/curriculum/):

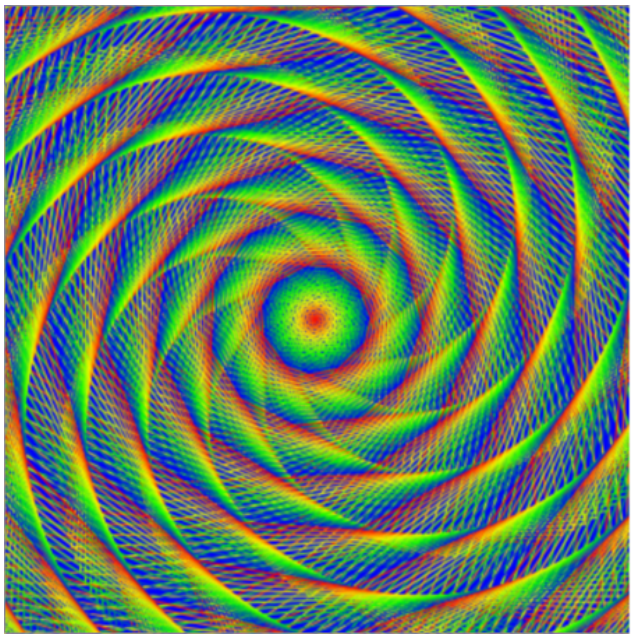
- **Combine programming constructs to solve a problem** (https://www.raspberrypi.org/curriculum/programming/builder)

Step 2 What you will need

- An internet connected computer

Step 3 Is it art, maths, or computer science?

Have a look at the image below. How would you describe it? Is is art, maths, or computer science?



It's a computer-generated image, but making it requires an understanding of art, maths, and computer science. Let's see how you too can make images just like this.

Step 4 Drawing a line

The image above is made up of lines and only lines! To get started, you need to know how to draw a line using a little bit of Python code. Below is an interactive Python environment called Trinket: you can write code in it, and then run it to see what happens.

- Click on **Run**

to see the code working.

- Now try changing the number in the line `turtle.forward(100)`, click on **Run**

again and see what happens.

Step 5 Turning

You've used code to draw a line. Good work! Now let's try making the turtle turn around. To do this you need to instruct the turtle not only to move forward, but also to turn right or left.

- What do you think will happen in the code above? Click on **Run** to see if you were right.

  `turtle.right(90)` turns the cursor 90 degrees right. You can also turn left with `turtle.left(90)`. To change the amount that the cursor turns, simply change the value of degrees.

- Complete the square shape you've started by adding the next lines of code and press **Run**. Keep trying until you get it right.

Challenge

Try to complete each of the challenges below.

- Draw a rectangle: two of the four sides need to be longer.
- Draw a triangle: how many degrees do you need to turn?
- Draw a cross: backwards and forwards work well together.
- Draw a circle: what happens if you turn lots?

Step 6 Changing colours

The default colour for the pen used by the turtle cursor is black, and the default background colour is white. You can change the colours to make your shapes look even better.

- Look at the code below. It contains three variables called `R`, `G`, and `B`.

Variables are a way of storing a value and giving it a name. For instance, there is a variable name `R` with a value of `255`.

The following line is commented out in the trinket above, but will be needed if you are using another editor. So if you are not working in Trinket, then remove the `#` symbol, to uncomment the line.

```
screen.colormode(255)
```

- Run the code and see what happens.
- Try changing the values of the three variables, and see what happens. (Note: the maximum value is 255, and after this there will be no effect.) What do you think R, G, and B represent?

Answer

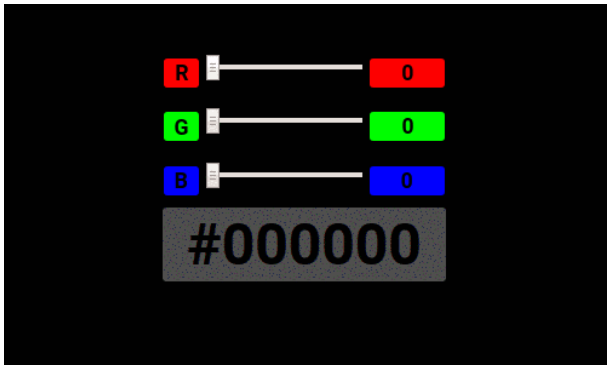
`R`, `G` and `B` represent how much red, green, and blue will be used in the colour. Each can have any value from `0` up to `255`.

So to make yellow, you could try the following:

```
R = 255
G = 255
B = 0
```

RGB colours

When we want to represent a colour in a computer program, we can do this by defining the amounts of red, blue, and green that make up that colour. These amounts are usually stored as a single byte and therefore as a number between 0 and 255.



Here's a table showing some colour values:

Red	Green	Blue	Colour
255	0	0	Red
0	255	0	Green
0	0	255	Blue
255	255	0	Yellow
255	0	255	Magenta
0	255	255	Cyan

You can find a nice **colour picker to play with at w3schools (http s://www.w3schools.com/colors/colors_rgb.asp)**.

You can change the value of your variables either by setting them to a new value, or by increasing and decreasing them.

- You can change the colour of the turtle as well. Run the code below to see what happens:

Challenge

Try to complete each of the challenges below.

- Complete the triangle above with a colour of your choice.
- Draw a square with sides which are four different shades of red.
- Draw a cross made of four different colours.

So you could alter colours by doing the following:

```
R = 255
G = 0
B = 0

turtle.color((R, G, B))
turtle.forward(100)
turtle.right(120)

R -= 20
G += 20
B += 5

turtle.color((R, G, B))
```

```
turtle.forward(100)
turtle.right(120)
```

Step 7 Repetition

Repeating lines of code is one of the fastest ways to get something done. Quite often in computer science, it makes more sense to repeat lines of code rather than write out another set of instructions. For example, the square you created earlier uses the same two instructions four times. Rather than writing them out four times, you could write them out once but add an instruction to repeat them.

In Python there are two types of loops that you are likely to use: a `while` loop and a `for` loop. If you want a section of code to repeat forever, or until a condition is set, then a `while` loop might be best. If you want to loop for a set number of times, then a `for` loop is preferable.

- Here, we have used a `while True` loop. This means that the code inside the loop (i.e. the code which is indented) will repeat forever. You can try it in Trinket to see what it does, but remember it will loop **forever**!

  ```
  from turtle import Turtle, Screen

  turtle = Turtle()

  while True:
      turtle.forward(1)
      turtle.right(1)
  ```

  This type of loop is not going to be very useful for drawing shapes with Turtle where you want to be more precise.

- In this example, a `for` loop has been used. Press **Run** to see what happens.

  A `for` loop repeats instructions a set number of times, in this case 8 times. A `for` loop has an associated variable (called `i` here). In this example, `i` starts from `0` and increases by `1` each time. Let's apply

this to the code to draw a square:

```
from turtle import Turtle, Screen

turtle = Turtle()

for i in range(4):
    turtle.forward(100)
    turtle.right(90)
```

- Copy and paste this code into the Trinket editor above and run it. The turtle has been asked to repeat two instructions four times to make a square.

- Once you have created one shape using a loop, you can repeat the shape again and again by putting it inside another loop. This is a great way to draw spirals. Adapt your code by making it look like this:

```
from turtle import Turtle, Screen

turtle = Turtle()

for i in range(30):
    for i in range(4):
        turtle.forward(100)
        turtle.right(90)
    turtle.right(25)
```

A spiral can be made by turning a small degree and then moving forward a small amount. The section of code for making a square is inside another `for` loop that repeats it 30 times, each time turning the cursor 25 degress to make a pleasing spiral shape.

Challenge

Try to complete each of the challenges below.

- Can you alter the `for` loop so that it draws a more interesting spiral using one of the shapes you made earlier, like a triangle or circle?

- Adding a few extra lines where you alter the variables `R`, `G`, and `B` would allow you to make a multicoloured spiral. Have a go at creating a rainbow spiral.

Try this to get started:

```
from turtle import Turtle, Screen

turtle = Turtle()
screen = Screen()

R = 255
G = 0
B = 124

for i in range(30):
    turtle.color((R, G, B))
    for i in range(4):
        turtle.forward(100)
        turtle.right(90)
        R -= 1
        G += 1
    turtle.right(25)
```

Step 8 Better spirals

- Have a go at reading over the code below and guessing what it does. Then run it to see if you were correct.
- The first thing you'll probably notice is that this is going to take ages to run. We can speed things up a bit, though. Add in the following line, before the `for` loop:

```
turtle.speed(0)
```

- Run the code again: it should be a little faster.
- Our code has made a multicoloured spiral by changing the `R`, `G`, and `B` variables. The colours are a little one-dimensional, though. Can you do a better job?

Step 9 Loopy colours

To get more interesting colours, you could write lots of colours in a long list, and then keep changing the colour of the turtle according to the colour of the list. You can create lists in Python, using square brackets `[ ]`.

Below is an example of a list of RGB colours:

```
colours = [(85, 211, 136), (197, 196, 126), (235, 233, 166), (25, 135, 222), (211, 64, 159), (159, 165, 106), (178, 160, 125), (36, 192, 70), (231, 184, 204), (63, 203, 219)]
```

- This next bit gets a bit complicated. Have a look at the code below, then run it to see what happens.

  The line `turtle.color(colours[i])` is telling the program to choose the "`i`th" item in the list. Remember that `i` starts from 0 and goes up to 9.

- What if you want a longer line? Try changing the number of loops in the `for` loop to `range(20)` and see what happens. Do you get an error?

Step 10 Modulo to the rescue

In the previous example, you need a way to keep looping over the list items, so when `i` gets to 9, it will go back around and get the "0th" item from the list again. This is where the modulo operator `%` can help you out.

- Look at the code below: run it and see if you can figure out what is going on. You shoud get `0` to begin with.

- Try changing the numbers in the `print` command. There are some examples to try below:

```
print(17 % 6)
print(12 % 6)
print(13 % 6)
print(6 % 6)
print(0 % 6)
print(1 % 6)
print(8 % 6)
print(11 % 6)
```

- Did you figure it out? The `%` operator prints out the remainder of a division. For example, 15 ÷ 6 is 2 with a remainder of 3. Therefore 15 % 6 would be 3. We can use this operator to help with the problem of running off the end of the list. If the `range` goes above the length of the list, you can just do a `%` of the length of the list.

- Have a look at the example below, and read through the code carefully to make sure you can see how the modulo operator is used.

Step 11 Enormous lists

You can now have a go at creating a list of colours that's a little longer than before. To do this you can use a `while` loop. Unlike a `for` loop, a `while` loop keeps running until a specific condition has been met.

- Look at the code below. The `while` loop is used to gradually increase the value of `G` until it reaches 255. Each time, the colours are added to the list.

- Can you add in two more `while` loops to add more colours? The next loop should gradually decrease `R` until it reaches 0. The final one should then increase `B` until it reaches 255. Have a go, but if you get stuck, all will be revealed in the last section.

Step 12 Putting it all together

You can now use the `while` loops along with the spiral code to make a really pretty spiral. Have a look at the code below, and make sure you understand what it is doing. Try playing with the value of the variables in the loops, to see what effect this has on the output.

Step 13 What next?

- Learn how to use `functions` to **draw snowflakes** (https://projects.raspberrypi.org/en/projects/turtle-snowflakes/) using Turtle.
- Create interactive stories using `lists` in Python with the **Storytime** (https://projects.raspberrypi.org/en/projects/storytime/) resource.
- Take your first steps **controlling physical objects** (https://projects.raspberrypi.org/en/projects/physical-computing) with Python and a Raspberry Pi.