

Lesson 4: Using loops to create shapes

Repetition in shapes

Lesson 4: Using loops to create shapes

To modify a count-controlled loop to produce a given outcome

- I can identify the effect of changing the number of times a task is repeated
- I can predict the outcome of a program containing a count-controlled loop
- I can choose which values to change in a loop

How many bricks?

I am making a brick wall from yellow and red bricks.

I add 4 bricks of each colour, but my wall is only **half** as high as it needs to be. How many times would I need to repeat the yellow and red bricks to get **all the way** to the top?

4 × [yellow, red] = halfway

? × [yellow, red] = all the way



How many bricks?

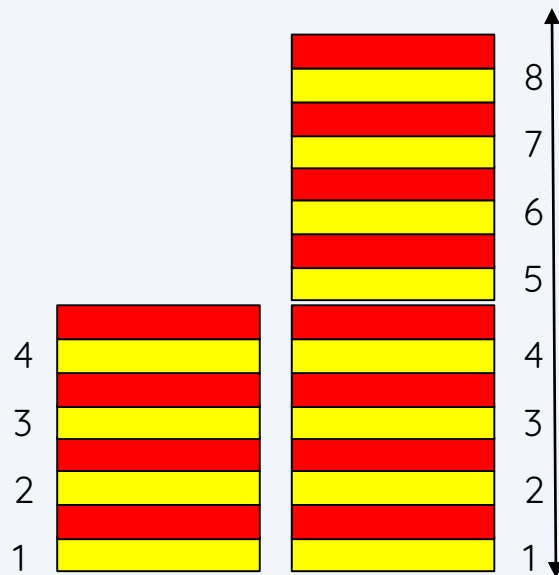
4 × [yellow, red] = halfway

8 × [yellow, red] = all the way

The number before the loop (in **bold**) tells us how many times to repeat the loop (the part in brackets).

It is the same in code, eg

```
repeat 4 [fd 100 rt 90]
```



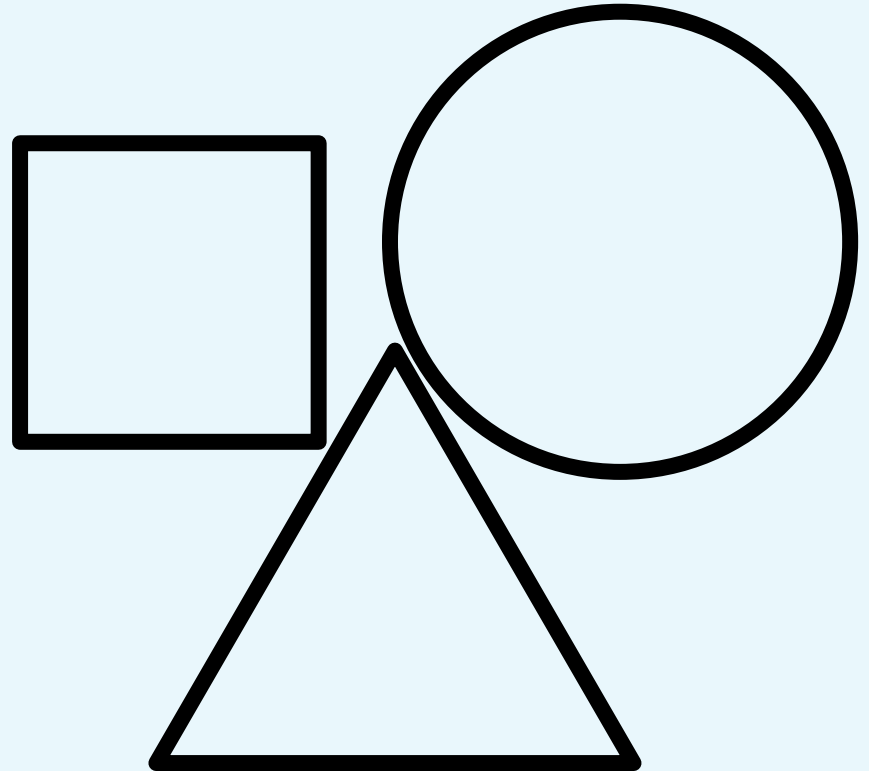
Predicting shapes - step 1

Draw the start point and direction of your turtle in the centre of your whiteboard.

Trace the following code by drawing it one step at a time.

```
FD 100 RT 90
```

What have you drawn?

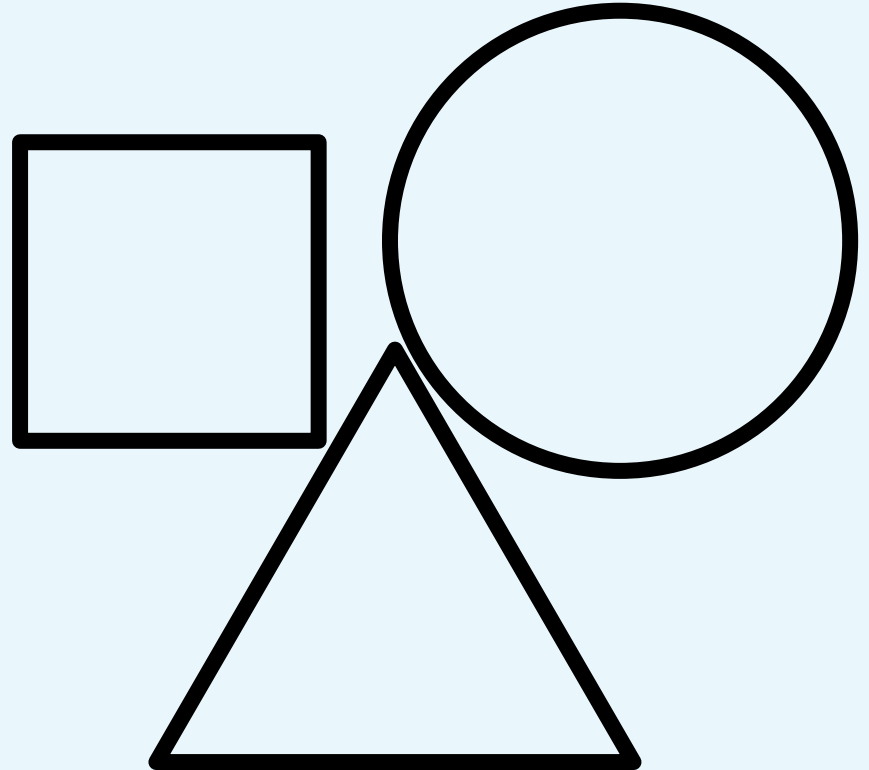


Predicting shapes - step 2

Add the next step:

FD 100 RT 90

What have you drawn?

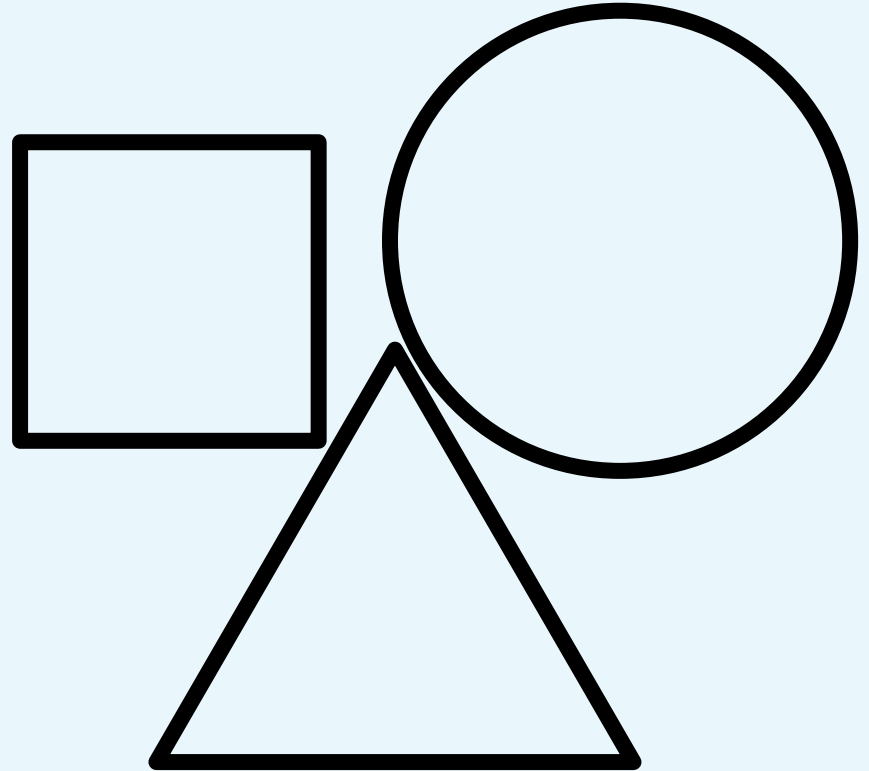


Predicting shapes - step 3

Add the next step:

FD 100 RT 90

What have you drawn?



Predicting shapes – step 4

Add the final step:

```
FD 100 RT 90
```

What have you drawn?

Using a **repeat** loop, our code would look like this. This code makes a square.

```
REPEAT 4 [FD 100 RT 90]
```



Predicting shapes

Here is a new code snippet:

```
REPEAT 3 [FD 100 RT 120]
```

Which values are different from the code for the square?

```
REPEAT 3 [FD 100 RT 120]
```

Predicting shapes

Read the code below out loud.

CS

```
REPEAT 3 [fd 100 rt 120]
```

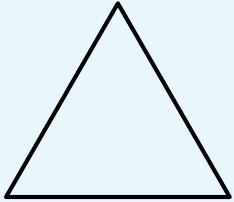
What shape do you think it will make?

Why do you think that?

Let's test the code in Logo.

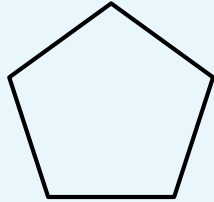
Turns in Logo

Triangle



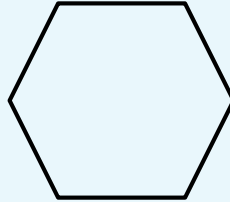
120°

Pentagon



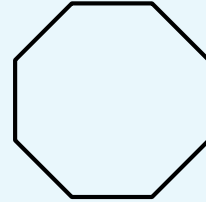
72°

Hexagon



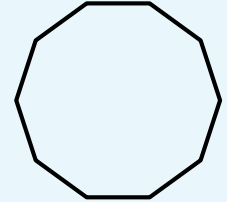
60°

Octagon



45°

Decagon








36°

The angle shown is the amount of turn needed for each corner of the shape.

Writing code for shapes

Use the activity sheet to work out the code snippets required for different shapes. Think about which values need to change in each example.

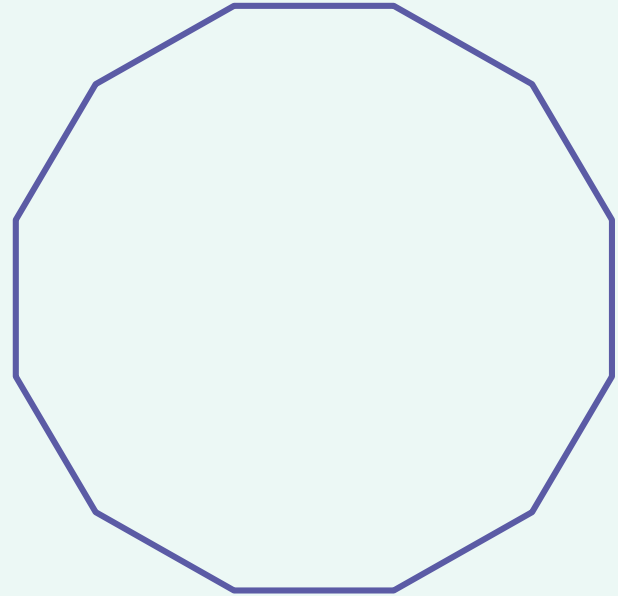
Shape	Exterior angle (360/number of sides)	Code snippet
 triangle	120°	REPEAT 3 [FD 100 RT 120]
 pentagon	72°	
 hexagon	60°	
 octagon	45°	
 decagon	36°	

Creating shapes

Using the code that you have just created, program the shapes in Logo.

Once you have programmed them, try these:

- Work out the repeat code for a rectangle
- Work out the code for any other regular shapes, eg a decagon
- Change the size of your shapes



Using loops to create shapes

- Look at these two code snippets:
 - `repeat 4 [fd 100 lt 90]`
 - `repeat 7 [fd 100 lt 90]`

- What will the turtle draw for each one?
- Where will the turtle be at the end?

Share ideas with a partner and explain your thinking.

Using loops to create shapes

```
repeat 4 [fd 100 lt 90]
```

This is the code snippet needed to create a square. When we use repetition in programming, it is called looping.

We can program a loop to stop after a specific number of times — here, it needs to loop 4 times.

How confident are you? (1-3)

- I can identify the effect of changing the number of times a task is repeated
- I can predict the outcome of a program containing a count-controlled loop
- I can choose which values to change in a loop

3 - Very confident



2 - Unsure



1 - Not confident



Next lesson

In this lesson, you...

Predicted what would happen from a code snippet

Changed values in loops to create shapes

Used count-controlled loops to create different shapes

Next lesson, you will...

Create and use procedures to make different shapes