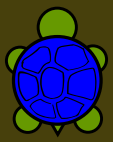


# Blue Shell Challenge Cards

**Teachers Notes**:

This set of Challenge cards is designed to be used with the Blockly Turtle resources. The full collection of resources for teachers is found in the **Turtle** area on bebras.uk.

This set of cards is for pupils who have achieved their Blue Shell Programmer award and are now working towards their Brown award. These cards introduce some new code blocks in a Function folder. Functions are like small programs that can be called again and again in a main program.

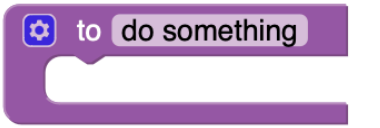
**Preparation**:

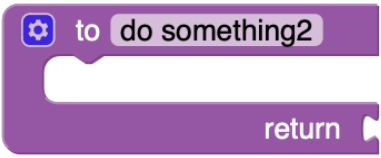
1. When the pupils login to their computers they should head to the ***Turtle Playground - Blue***. They should be directed to: bebras.uk -> Turtle -> click on the Blue turtle.
2. These cards should be printed out (size to:100% on A4 card, or "fill the paper" on A5 card) and laminated. Each pupil also needs their own Yellow Shell Record Card (which should not be laminated as they have to be written on). When a pupil completes a Challenge Card, its number can be written in their Record Card (in one of the clip boards).
3. In the first lesson, the teacher should show the students how to access ***Turtle Playground - Blue*** and the Functions video on Card 0. Note ***Card 0*** is for the teacher to use with the class. Pupils can start with ***Card 1***.
4. Students should complete a minimum of 8 cards so some choice is available.

# Blue Shell Challenge Cards

Code Blocks introduced in **Turtle Playground - Blue**:

A new ***Functions*** folder containing these blocks:



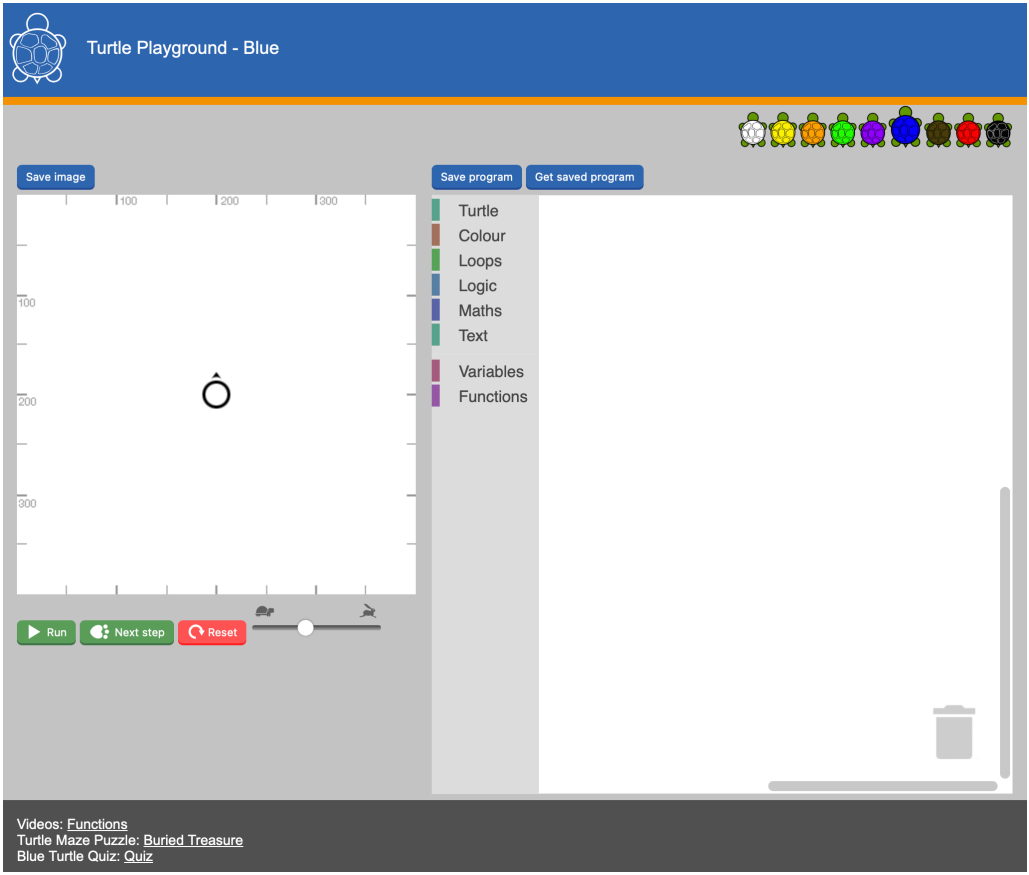



This Card is for teachers!

# Blue Shell Challenge Cards

0

1. Show your pupils how to go to ***Turtle Playground - Blue***




2. Show your class the Functions video.
3. Provide each pupil who has achieved their Blue Shell award with their new Record Card.
4. Distribute these Challenge Cards.

For the teacher of pupils working towards their Brown Shell Turtle Programmer award using Turtle Playground - Blue

# Blue Shell Challenge Cards

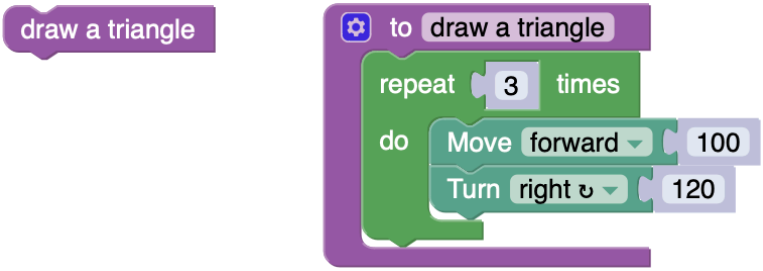
1

## Triangles

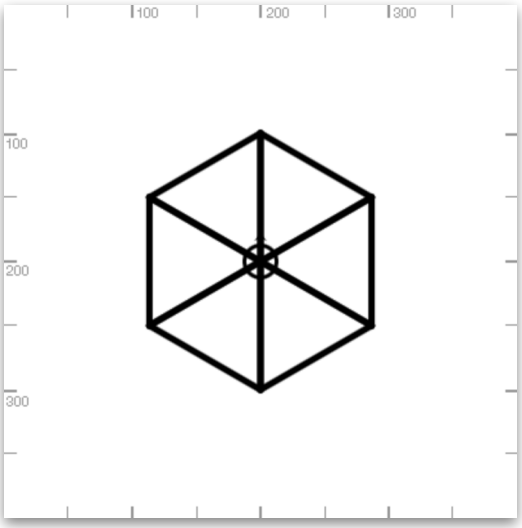
**Functions** are like small programs in another program. Making a function is also like designing your own code block.

### Challenge:

1. Make this program that draws a triangle:




2. Edit your program so that it uses this function to draw the shape shown:




### Hint:

- Try turning 60 degrees after drawing each triangle.

# Blue Shell Challenge Cards

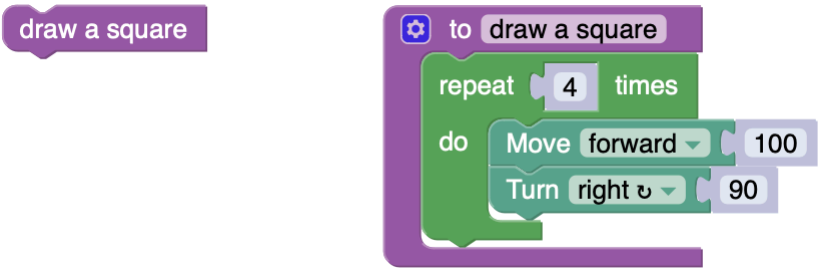
2

## Squares

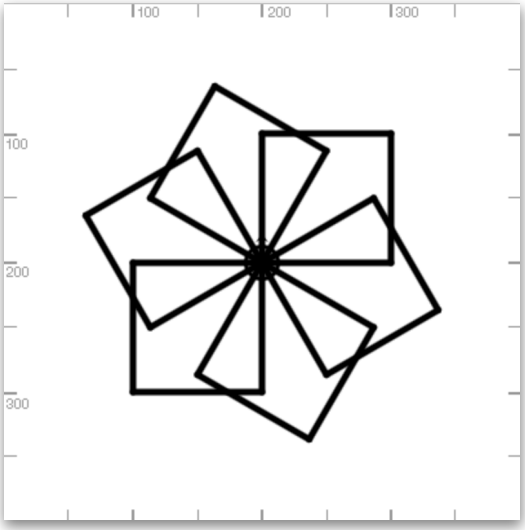
**Functions** are like small programs in another program. Making a function is also like designing your own code block.

### Challenge:

1. Make this program that draws a square:




2. Edit your program so that it uses this function to draw the shape shown:




### Hint:

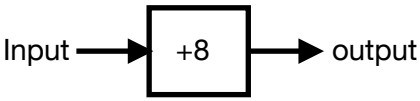
- Try turning 60 degrees after drawing each triangle.

# Blue Shell Challenge Cards

3

## Function machines

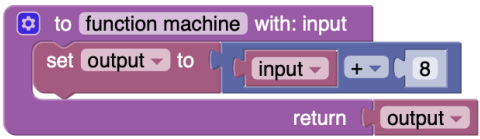
In maths lessons you may have met function machines like this one:




The square box is the function machine that adds 8 to any number sent to it and then **outputs** the answer.

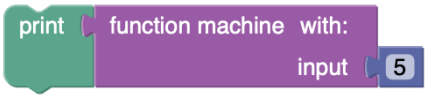
## Challenge:

1. Make this Blockly version of the function machine shown above:




   You will need to click on the **cog** to add an **input** (we have called it "input") and make a new **variable** (which we have called "output"). This is the box function machine shown above.

2. Make this short program. It **calls** the **function** and **outputs** the result - just like the function machines taught in maths lessons.



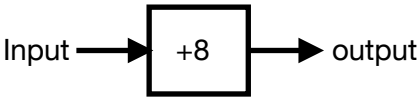

   a) Run this program and see how it works.
   b) Try changing the input.
   c) Try making a different function machine (e.g. one that multiplies by 5). Notice how you only need to edit the code blocks inside the function.
   d) Try making another function machine.

# Blue Shell Challenge Cards

4

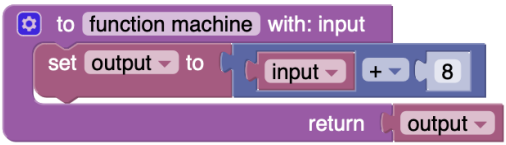
## Function machines

In Blue Shell Challenge Card 3 you made a function machine like the ones taught in maths lessons.

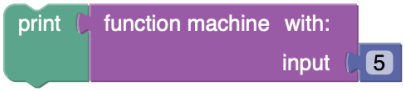



The square box is the function machine that adds 8 to any number sent to it and then **outputs** the answer.

Here is the **function**:




To **call** it we made this program:




## Challenge:

1. Make another version of this function machine that does not need any **input**:
   a) click on the **cog** in the function and remove the input.
   b) Use a *prompt for text with message* block to get the input instead.
   (You will need to change it to a *prompt for number with message* block.)
   Your new function should be called and printed with this small program:


2. Try making some other function machines.

# Blue Shell Challenge Cards

5

## Useful science equations

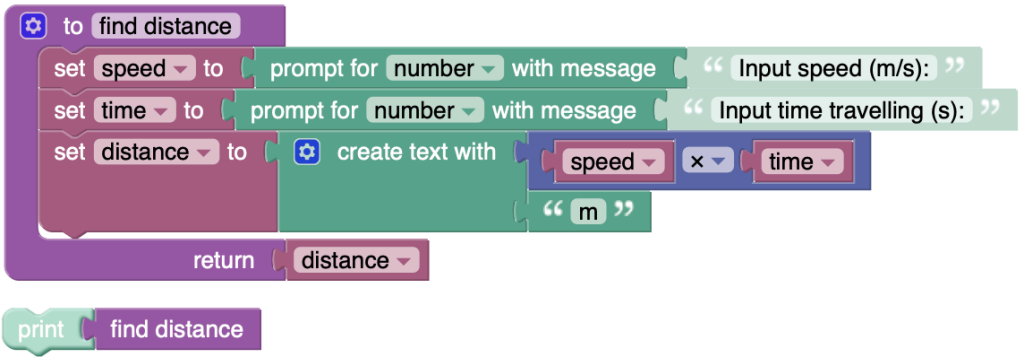
To find out how far (in metres) you can go, in a number of seconds, if you are travelling at a given speed (in metres per second), you can use this equation:

distance = speed x time

You will learn about a lot of these in secondary school science lessons!

## **Challenge**:

1. Make this program that includes a **function** to calculate distance:




2. Check the program works by running it and entering some numbers. Example: Travelling at a speed of 10 for 20 seconds should **output** 200.

3. Make a new program that calculates the volume of a box using this equation:

volume = width x length x height

## Draw a polygon

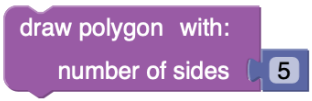
There is a simple formula to find out the angle the turtle needs to turn to draw regular polygons:

angle = 360 ÷ number of sides

(e.g. for a square the number of sides = 4, so angle = 360 ÷ 4 = 90)

## Challenge:

Write the **function** required by this program that draws a polygon of any number of sides:




This should draw a pentagon.
Check that, when the number of sides is 4, it draws a square.
Try different numbers to draw other regular polygons.

## Hints:

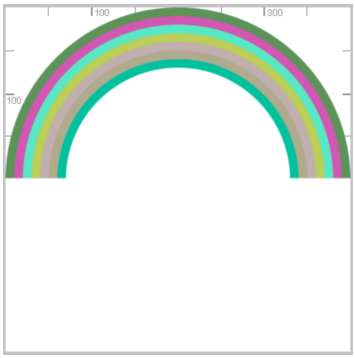
Don't read all of these hints before starting. Try to use the minimum number of hints that you can.

- You will need a **variable** called angle to store the angle.
- Your function will need to calculate the angle.
- Your function will need to move forward 100 and turn the calculated angle once for each side.
- Your function will need to contain a **loop** to draw each side and turn.
- Think about how many times you will need to loop to draw each polygon.

# Blue Shell Challenge Cards

7

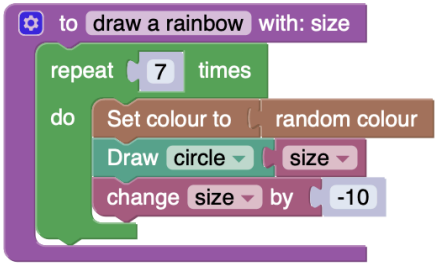
## Draw a rainbow

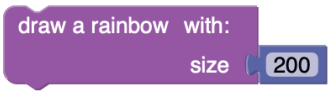


Your friend needs help.

They want to draw a rainbow that looks like this:

Here is your friend's program, but they don't know how to finish it.




## Challenge:

1. Copy out your friend's program and run it to see what it does.
2. Add some code to the **function** to complete their program so it does what they would like it to do.

## Extra:

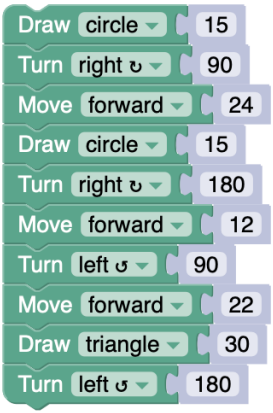
Amend the program so that it asks the **user** how big the rainbow should be.

# Blue Shell Challenge Cards

8

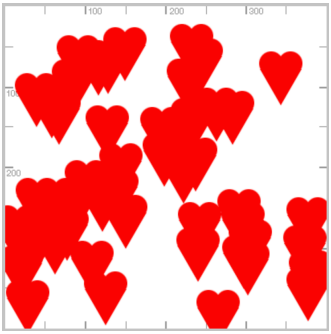
## Random hearts

The program on the right draws a heart shape:




## Challenge:

1. Copy out the heart code and put it in a **function** called “draw heart”.
2. Write a program that draws 50 random red hearts using your new function.



## Extra:

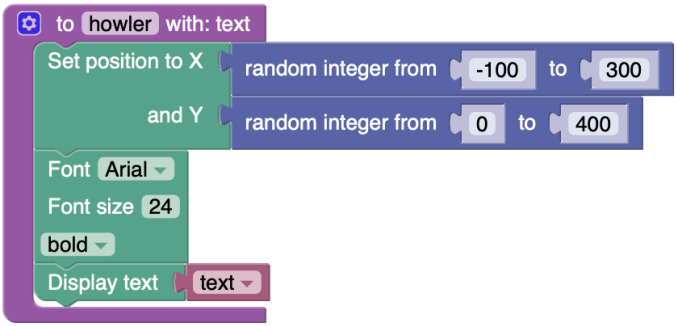
Edit your program so that it draws 50 randomly placed hearts, with random colours.

## Send a Howler

While at Hogwarts, Ron Weasley's mother sends him a magical letter called a howler. When opened, the letter flew into the air and shouted angrily at him.

## Challenge:

1. Make this **function** and then run it:




2. **Call** it like this:


Although this saves us having to remember various settings to **format** our text, it still needs some work!

3. Add some more **code blocks** to the howler function so that we hide the turtle and and set the writing to a random colour. Don't fix the fact that the writing is in the wrong direction yet.

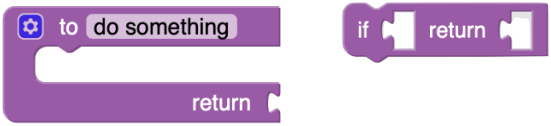
4. Call the howler 100 times in a **loop** and then work out how to make sure the numbers appear the right way up.

# Blue Shell Challenge Cards

10

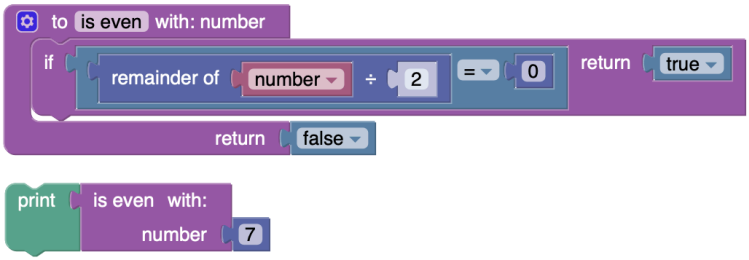
## Small, medium, large

Sometimes you may want a **function** to **return** different answers depending on the **input**. This can be done by combining these two blocks:




## Challenge:

1. Make this program and then run it:




   Try changing the input value from 7 to an even number.

2. Make a new function that takes a number as input and **returns**:
   "small" if number is below 10,
   "medium" if number is 10 or larger and less than 100,
   "large" if number is 100 or above.

3. Write a function called "favourite colour" that takes the name of a colour as input and returns "Yes, that is the best colour!" if the colour is "yellow" and "Oh, how horrid!" if it is any other colour.

# Password checker

## Challenge:

1. Write a **function** called “check password” that takes some text as **input** and then either **returns**:
   “Thank you for entering a password.” if the password has 4 or more characters.
   or:
   **calls** itself with the prompt: “Enter a longer password:” if the password is made of 3 or less characters.

   Your program should call the function and print out the result like this:


## Hints:

1. Your function will need to contain an *if … return* block and test to see how long the text is.


2. It may come as a surprise but functions can return a call to themselves! The code blocks above (excluding the print block) can be copied and attached to the final return space in your function. Try it and see how it works!

## Circles

A circle has a number of properties:

- The **radius** is the distance from the centre of a circle to its edge.
- The **diameter** is the distance from one edge to the other passing through the centre of the circle. Diameter = 2 x radius
- The **circumference** is the distance around the edge of a circle and can be calculated using the formula:
  circumference = 6.3 x radius
- The **area** of a circle can be calculated using the formula:
  area = 3.14 x radius x radius

## Challenge:

1. Write a **function** called “diameter” that takes a number called “radius” as **input** and then **returns** the value of the circle’s diameter.
2. Write a new function called “circumference” that takes a number called “radius” as input and then returns the value of the circle’s circumference.
3. Write a new function called “area” that takes a number called “radius” as input and then returns the value of the circle’s area.

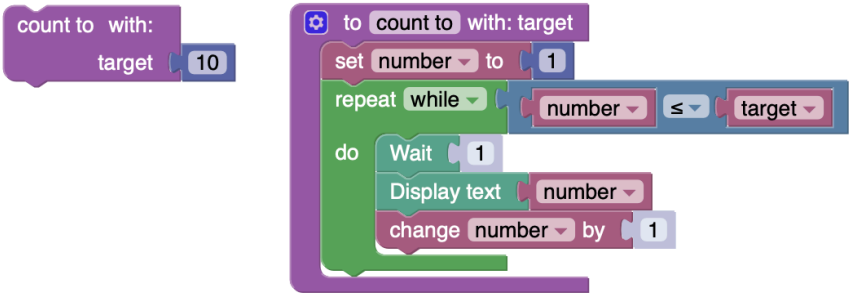
## Extra:

1. Write a function called “circle info” that takes the radius as input and returns some text giving the diameter, circumference and area of the circle. Write the code to **call** this function and **output** the text it returns.

## Counting

### **Challenge**:

1. Copy out this program:




The program is supposed to count, in this case to 10.
There are a few problems. Can you fix them?

2. Try and fix these problems:
   The text is too small.
   The turtle needs hiding.
   The text needs turning.
   The numbers print on top of each other.
   It would be best if the numbers appear in the middle of the drawing area.

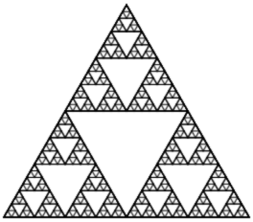
3. Save your program when you are happy it does what it is supposed to do.

4. Edit your program so that instead of counting up, it counts down to 0.

# Blue Shell Challenge Cards

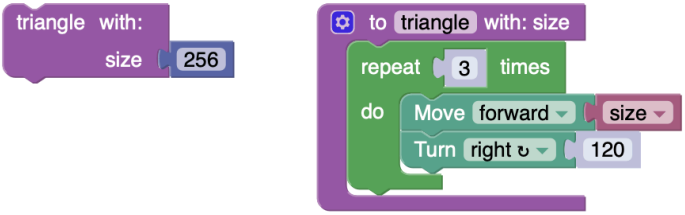
14

## Triangles in triangles



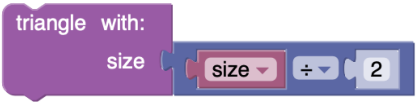
### Challenge:

1. Write this program that uses a **function** called “triangle”:



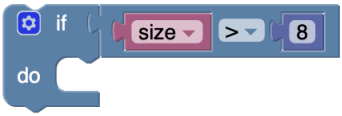

This is pretty simple to understand. It draws a side and turns right. It does this 3 times. We can change the size by changing the number 256. Don’t do this yet. (256 is a special number that can be repeatedly divided by 2 until it reaches 1 without ever producing a remainder.)

2. Inside the **loop**, just after the Turn **block**, add this group of code blocks:




Your **function** is **calling** itself at every corner of the triangle!
The new triangles also draw smaller triangles at all of their corners.
This will go on forever. You can run this but be ready to press Reset.

3. To make sure your program does not draw too many triangles, add this group of blocks around the *repeat 3 times* **code block**:




4. Try and centre your pattern and straighten it up. Then try changing the line thickness, and *size* in the if block. (Make sure to try: 4, 16, 32, 64 and 128.)