Australian **Computing** Academy

DT Challenge Blockly

Turtle

1. Introducing the Turtle
2. Angles with the Turtle
3. Looping with the Turtle
4. Add a Dash of Colour
5. Asking Questions
6. Making Decisions
7. Extension: Putting it all together!
8. Playground!

The Australian Digital Technologies Challenges is an initiative of, and funded by the Australian Government Department of Education and Training (https://www.education.gov.au/).

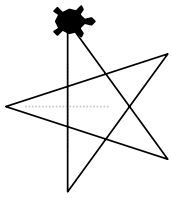
1

INTRODUCING THE TURTLE

1.1. Turtle

1.1.1. Introducing the turtle

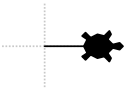
Meet the turtle (https://en.wikipedia.org/wiki/Turtle_graphics)!



In this course, you'll be using Blockly - a visual programming language - to drive the turtle. It's fun, and what you'll learn is the basis for all vector graphics (https://en.wikipedia.org/wiki/Vector_graphics) in computers.

1.1.2. Move forward

Let's make the turtle move! Click the ▶ button:



When you run the Blockly code, it makes the turtle move forward!

The number is the number of *turtle steps* to move. A bigger number will move the turtle further!

Try changing `30` to `100`, and running it again.

Where are the turtle blocks?

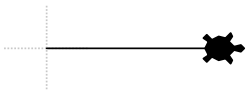
Turtle blocks only work on turtle questions. If you want to try them, skip to the next question, then come back to the notes!

1.1.3. Problem: Make a move!

Let's make a move!

Write a program to move the turtle forward **100 steps**.

The easiest way to do this is to use move forward 100 steps to move the turtle forward!



Hint

You will need to change the distance the turtle moves forward from 20 to 100.

Testing

- [ ] Testing the line.
- [ ] Testing there are no other lines.
- [ ] **Awesome! You've drawn your first picture!**

1.1.4. Turning corners

The turtle always starts facing to the right of the screen.

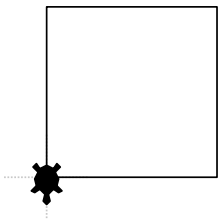
If you want to change which way the turtle is facing, you can use the turn left block. To make the block turn right, you need to click on left and select **right** from the dropdown list.

Click the ▶ button to see the turtle turning left:

Now the turtle is facing up! Change it to turn right by clicking the dropdown in the block and run it again.

1.1.5. Drawing a shape

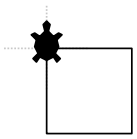
You can combine move forward and turn blocks to draw shapes:



1.1.6. Problem: Starting from Square One

Write a program to draw a square below the turtle.

The easiest way to do this is to use turn right rather than turn left. The turtle starts in the top left corner of the square:



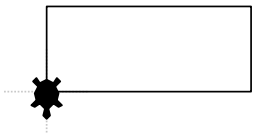
Each side should be **50 turtle steps** long.

Testing

- [ ] Testing the top of the square.
- [ ] Testing the right side of the square.
- [ ] Testing the bottom of the square.
- [ ] Testing the left side of the square.
- [ ] Testing the whole square.
- [ ] Testing for no extra lines.
- [ ] **Fantastic, you've drawn the *right* (pun intended) square!**

1.1.7. Problem: Get rect!

Write a Turtle program to draw a rectangle, with a width (top and bottom sides) of 120 turtle steps, and height (the left and right sides) of 50 turtle steps. The output of your program should look like this:



The *bottom left corner* of the rectangle is where the turtle starts.

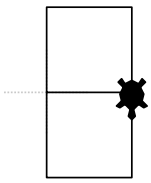
Testing

- [ ] Testing the top of the rectangle.
- [ ] Testing the left side of the rectangle.
- [ ] Testing the right side of the rectangle.
- [ ] Testing the bottom of the rectangle.
- [ ] Testing the whole rectangle.
- [ ] Testing for no extra lines.
- [ ] **Fantastic, you've solved this `turtle` problem!**

1.1.8. Problem: Domino

Write a program to draw a domino, made up of two squares on top of the other.

The turtle should start on the left in the middle of the domino:



Each side of each square should be **50 turtle steps** long.

Testing

- ☐ Testing the middle of the domino.
- ☐ Testing the bottom of the domino.
- ☐ Testing the top of the domino.
- ☐ Testing the whole domino.
- ☐ Testing for no extra lines.
- ☐ **Great work!**

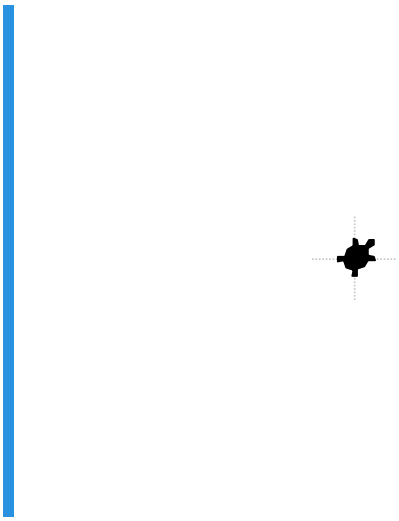
2

ANGLES WITH THE TURTLE

2.1. Angles

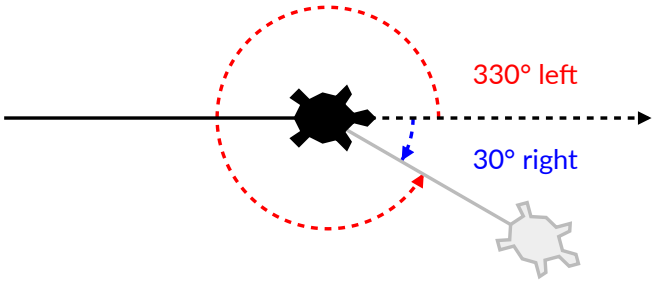
2.1.1. Turning different angles

So far our turn block has been *left* or *right* and this has been a 90 degree turn. We can turn the turtle different angles using a different turn block...



You can think of an *angle* as a *change of direction*. The blue block can have any number typed into it. Angles are measured in *degrees*. A 360° turn is a complete circle. Other turns are fractions of 360°.

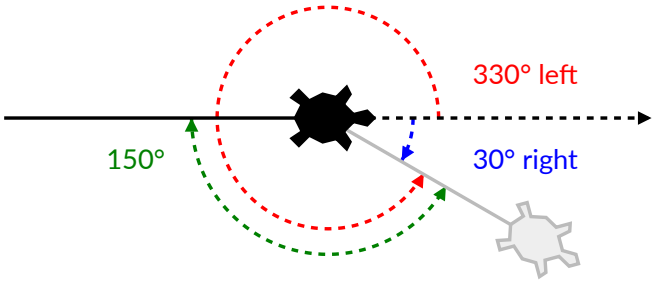
Try our interactive diagram! You can drag the grey turtle around.



You can see that the angle is the difference between the direction the black turtle is facing and the direction the grey turtle is facing. Most shapes can be drawn as a combination of lines and directions.

2.1.2. Use our turn calculator!

We've added a green *internal* angle between the two lines in our diagram. You can see the 180° – 150° = 30° turn you need:



If you get stuck with angle calculations, use the diagram!

2.1.3. Problem: Plate instead of bowl

Write a Turtle program to turn a very steep bowl shape into a plate.

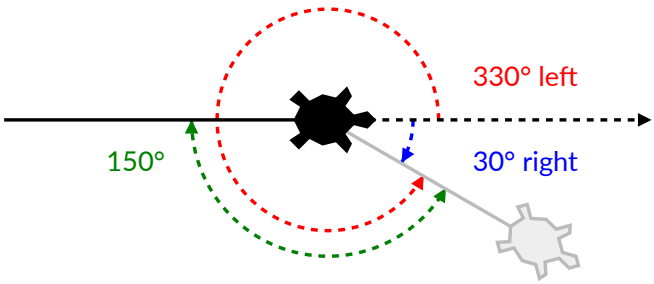
The lines should all remain the length they are (i.e. 50 turtle steps, 100 turtle steps and 50 turtle steps), but you need to change the angles.

Calculate the size of the turns you need to make. **The angle between the plate side and the plate base is 135˚.**

The turtle will start drawing from the top left corner:

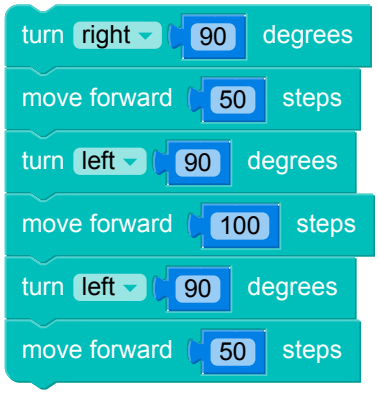
Hint

Try drawing the shape out on a piece of paper and calculating the angles you need before you start coding, and use the diagram below to help!



You'll need

program.blockly



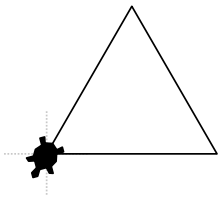
Testing

- [ ] Testing the left side of the plate.
- [ ] Testing the base line of the plate.
- [ ] Testing the right side of the plate.
- [ ] Testing the whole plate.
- [ ] Testing for no extra lines.

2.1.4. Problem: Equilateral triangle

Write a Turtle program to draw an equilateral triangle, with the sides being 100 turtle steps long. All angles in an equilateral triangle are 60°.

The output of your program should look like this:



The *bottom left corner* of the triangle is where the turtle starts.

Hint

You need to do an angle calculation to draw this shape!

Testing

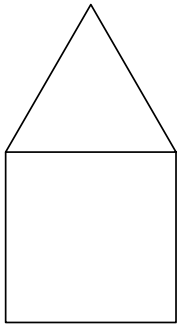
- [ ] Testing the bottom of the triangle.
- [ ] Testing the right side of the triangle.
- [ ] Testing the left side of the triangle.
- [ ] Testing the whole triangle.
- [ ] Testing for no extra lines.
- [ ] **Fantastic, you've drawn an equilateral triangle with `turtle`!**

2.1.5. Problem: Get your house in order

Use the turtle to draw a house!

The triangle at the top should have angles that are all 60° and all sides of the house should be 100 turtle steps long. The sides of the roof will also be 100 steps long.

The result should look like this:



The top left corner of the square is where the turtle starts.

Optional challenge!

Try drawing the house in one line, without drawing over the same line twice. Think about the *order* you need to draw the lines in.

Testing

- [ ] Testing the bottom of the roof (top of the square).
- [ ] Testing the right wall of the house (the right side of the square).
- [ ] Testing the left wall of the house (the left side of the square).
- [ ] Testing the floor of the house (the bottom of the square).
- [ ] **Testing the room of the house** (the square).
- [ ] Testing the left side of the roof.
- [ ] Testing the right side of the roof.
- [ ] **Testing the roof of the house** (the triangle).
- [ ] **Testing the whole house.**
- [ ] Testing for no extra lines.
- [ ] **Well done, your Turtle is now a master builder!**

3

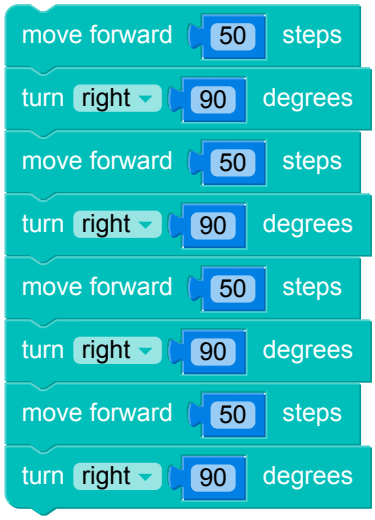
LOOPING WITH THE TURTLE

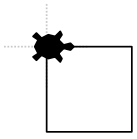
3.1. Loops with the turtle

3.1.1. Drawing shapes with loops

You probably noticed that you repeated yourself a lot in turtle programs. Using loops makes turtle much less repetitive!

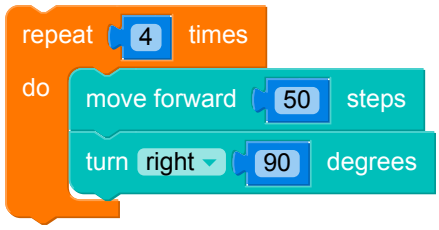
Drawing a square the long way:

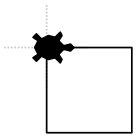




If we used a loop, then we wouldn't have to repeat same two blocks over and over again.

Here's a much shorter way of drawing a square, using loops:



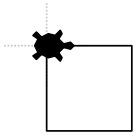


3.1.2. Problem: Back to Square One

Write a program to draw a square below the turtle.

Try using the repeat block to make life easier!

Each side should be **50 turtle steps** long.



Hint

Look for patterns that repeat themselves, and place those inside the repeat block!

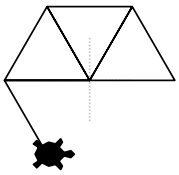
Testing

- [ ] Testing the top of the square.
- [ ] Testing the right side of the square.
- [ ] Testing the bottom of the square.
- [ ] Testing the left side of the square.
- [ ] Testing the whole square.
- [ ] Testing for no extra lines.
- [ ] **Fantastic, you've drawn the *right* (pun intended) square!**

3.2. Loops and Movement

3.2.1. Repeat Block

We can use loops like the repeat block to make some great patterns!

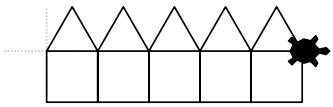


3.2.2. Problem: Row of terraces

We've already drawn one house - now let's draw a whole row of houses!

We've given you the code to draw one house, and angled the turtle to the right spot for drawing the next house. Your task is to change the program so it draws 5 houses in a row!

The result should look like this:

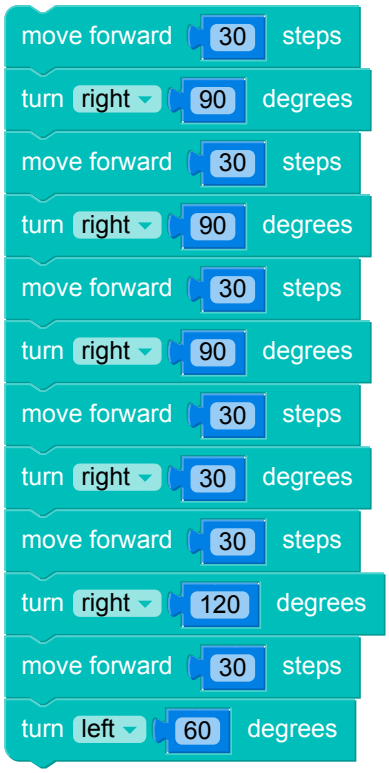


Hint

You only need to add one repeat block to solve this question!

You'll need

program.blockly



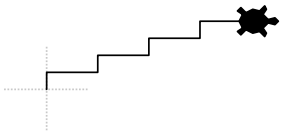
Testing

- [ ] **Testing the whole house.**
- [ ] Testing for no extra lines.
- [ ] **Well done! That's quite a row of houses!**

3.2.3. Problem: Staircase

Write a program that draws a staircase of **4 steps** that are **10 steps high** and **30 steps wide**.

The steps will go up and to the right of the screen, as shown in the example below.



Hint

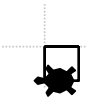
Think about what direction the turtle needs to face before you start drawing your steps.

Testing

- [ ] Testing the bottom step.
- [ ] Testing the whole staircase.
- [ ] Testing there are no extra lines.

3.2.4. Squares on squares on squares

When we use a loop block, it repeats everything inside of it. Let's say we wanted to draw 6 squares. We could try something like this:



Six squares are drawn, but **on top of each other**! To draw more squares, we'll need to add in an extra **move forward** step.

3.2.5. Moving between loops

Here, we've added an extra **move forward** step at the end of the instructions, so that each time the square is drawn in a new position, pointing the right way.

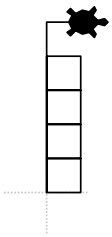
steps

steps

steps

steps

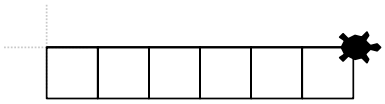
steps



3.2.6. Problem: Blocks in a Row

Let's draw a row of blocks!

The row should be 6 blocks long, and each square should have sides **30 turtle steps** long.



Testing

- [ ] Testing the right side of the first square.
- [ ] Testing the left side of the first square.
- [ ] Testing the whole square.
- [ ] Testing for no extra lines.
- [ ] **Great work!**

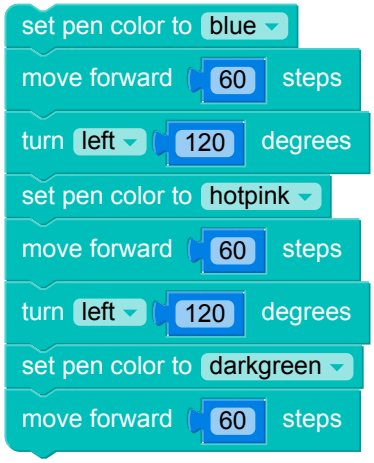
4

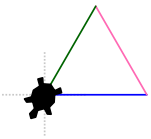
ADD A DASH OF COLOUR

4.1. Turtle colours

4.1.1. Now with colour!

Let's add some colour using the set pen color block:





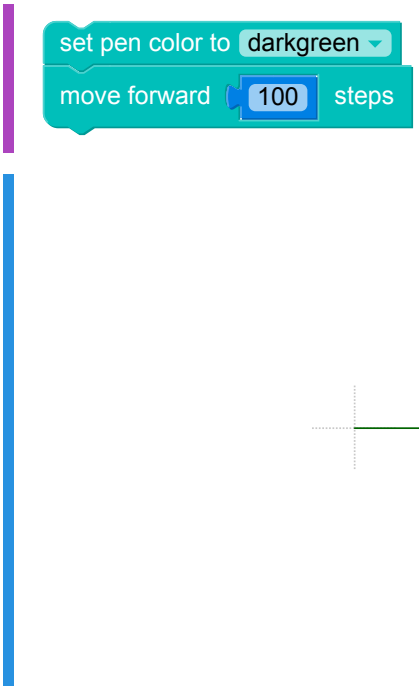
Most code uses *color* (American spelling)!

Most programs and modules (like `turtle`) will spell colour with the American spelling (c-o-l-o-r, no u) – so watch out!

4.1.2. Some of the colours of the rainbow!

There are lots of ways to set a colour. We're going to use a drop down list of some of the colours to start with. You can see the whole list of colour names that Turtle understands here (http://wiki.tcl.tk/37701).

We'll get to see a lot more colours very soon. You can try all of the ones we have in the list by changing the darkgreen below to a different colour.

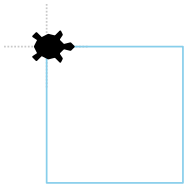


4.1.3. Problem: Skyblue Square

Let's add some colour to our square!

Write a program to draw a skyblue square using the turtle.

Each side should be **80 turtle steps** long.



Testing

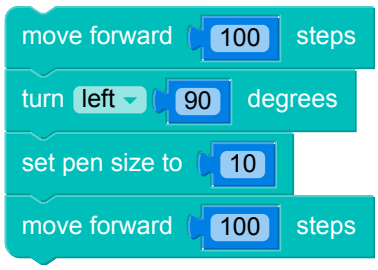
- [ ] Testing the top of the square.
- [ ] Testing the right side of the square.
- [ ] Testing the bottom of the square.
- [ ] Testing the left side of the square.
- [ ] Testing the whole square.
- [ ] Testing for no extra lines.
- [ ] **Great work! Blue skies (and boxes) from here!**

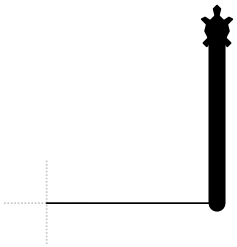
4.2. Turtle lines

4.2.1. Drawing thicker lines

Those lines are looking nice and colourful, but they're a bit thin! We can change how thick the pen is using the set pen size block.

The default pen width we have used so far is 1.





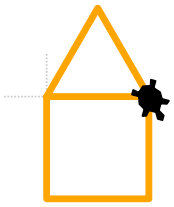
Try it out with different sizes!

4.2.2. Problem: Build a sturdy house

The house we drew earlier looks a bit flimsy. Let's draw one that looks more sturdy!

Draw an **orange** house with walls of **pen size 4**!

Each side should be **60 turtle steps** long.



Testing

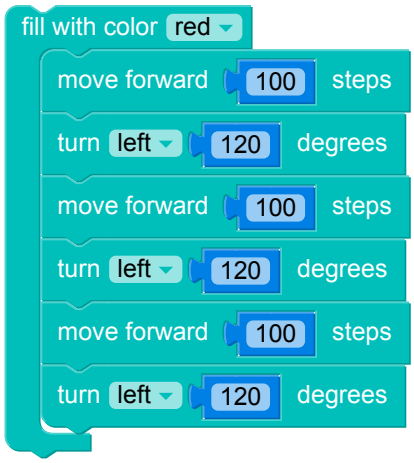
- [ ] Testing the top of the square of the house.
- [ ] Testing the right side of the house.
- [ ] Testing the bottom of the house.
- [ ] Testing the left side of the house.
- [ ] Testing the left side of the roof.
- [ ] Testing the right side of the roof.
- [ ] Testing the whole house.
- [ ] Testing for no extra lines.
- [ ] **Nice building work!**

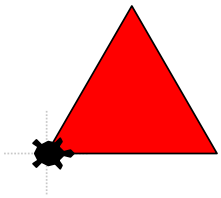
4.3. Filled shapes with the turtle

4.3.1. Filling with colour

The fill with color block can be used to fill in a shape with colour.

Any shapes which you draw inside of fill block will be filled with the colour you choose (at the end of drawing the shape).

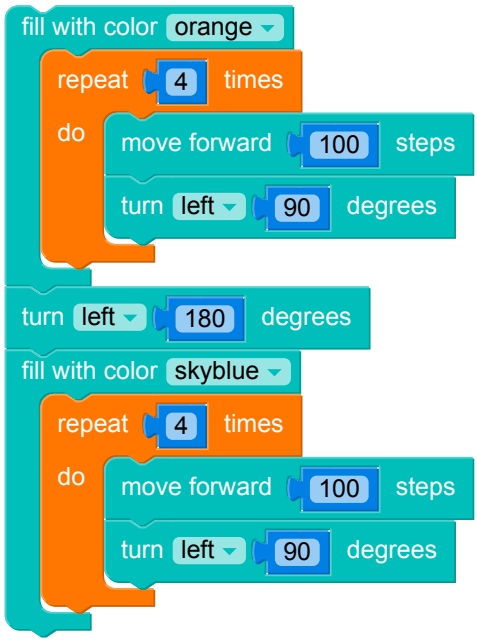


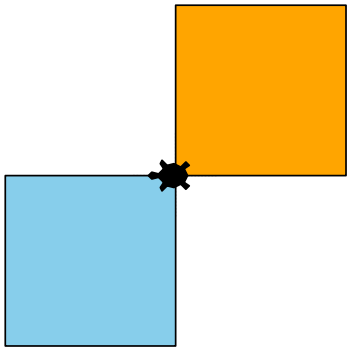


Try guessing what this program will draw before running the example! Then try changing the fill colour!

4.3.2. Fills and loops

We've already seen how drawing shapes is easier with loops, let's draw some shapes with both loops and fills!

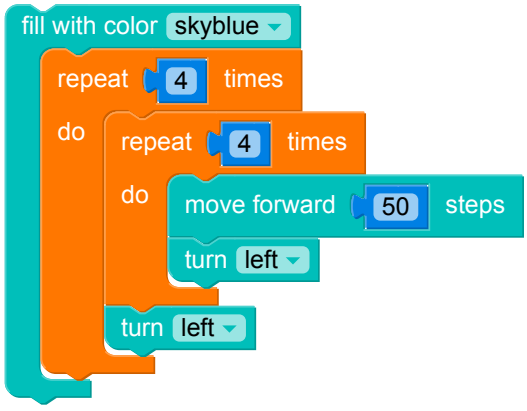


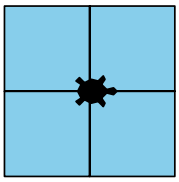


If we tried to put the fill *inside* the loop, then it would try to separately fill each side of the shape, which wouldn't work!

4.3.3. Choosing the fill colour

Just like the `pencolor` block, there's another `fill with color` block which uses the name of the colour.





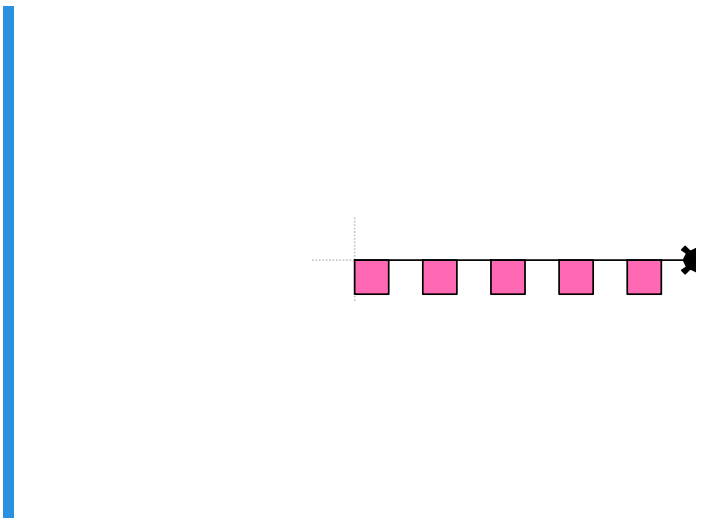
4.3.4. Problem: Square Pennant Flags

Papel Picado

In Mexico decorations for parties and festivals often include Papel picado (https://en.wikipedia.org/wiki/Papel_picado). These are very colourful squares of paper which have beautiful cut-out designs.

You're going to make the turtle draw a string of 5 simple colourful Papel picado.

The **five** flags should be squares that are **20 turtle steps long** on each side, there will need to be an **extra line** to join the flags together, this will also be 20 turtle steps long. To make it super colourful you need to get the turtle to fill with **hot pink**.



Testing

- [ ] Testing the right side of the first flag.
- [ ] Testing the bottom of the first flag.
- [ ] Testing the left side of the first flag.
- [ ] Testing the drawing of all the outsides of the flags.
- [ ] Testing the fill colour.
- [ ] Testing for no extra lines.
- [ ] **Time to party!**

4.3.5. All the colours of the rainbow!

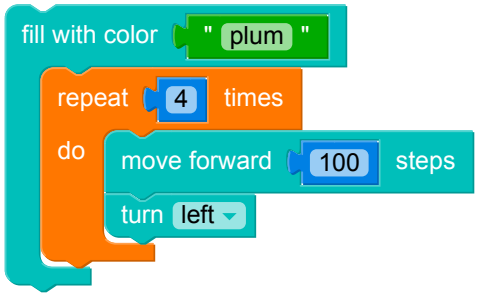
We don't just have to use our list of colours we can use **ALL** the colours. To do this we need to talk to the turtle with something called a **string**.

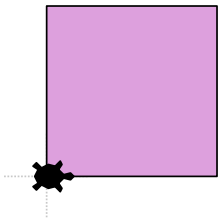
A **string** is just text. It is a group of letters or numbers put in a meaningful order, like a word, sentence or car number plate.

Computers don't understand **red** or **purple**, or any other human language. To a computer, they are just a string of letters.

The green string block can contain any letters, numbers, punctuation and spaces that you want to use in a message. We're going to use letters to tell the turtle which colour we want.

For example, the square below is filled with the colour **plum**. Try changing it to another colour by changing the name inside the green block.





The individual letters, digits, symbols and spaces are called characters and the word string is short for string of characters.

There are a huge number of colours that have names we can use. Here are some of our favourites:

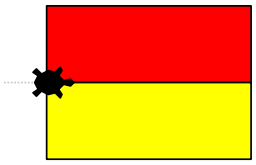
			gold	yellow
		springgreen	lawngreen	green
				skyblue
				plum
		pink	lightpink	mistyrose
			tan	wheat
			lightgray	white

4.3.6. Problem: Swim between the flags!

In Australia it's very important when we go to the beach to swim between the flags. The flags are a very specific colour pattern to make them easy to recognise. They have a red rectangle on the top and a golden yellow colour on the bottom.

Get the turtle to draw a lifesaving flag. It should have one rectangle on the top that is **red** and one rectangle on the bottom that is **yellow**. The rectangles should be **120 turtle steps** long and **45 turtle steps** high.

The turtle should start on the left in the middle of the flag:



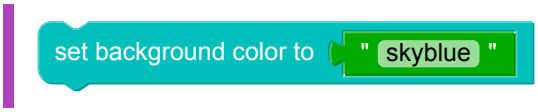
Testing

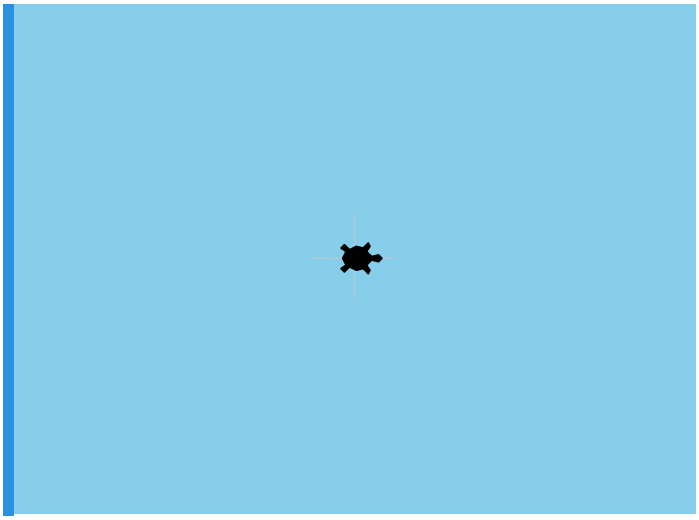
- [ ] Testing the middle of the flag.
- [ ] Testing the top of the flag.
- [ ] Testing the bottom of the flag.
- [ ] Testing the whole flag.
- [ ] Testing for no extra lines.
- [ ] **Great work!**

4.4. Advanced Turtle-fu!

4.4.1. Background colours

It's boring to always have a plain white background. That's why we have a background color block.





Also try some different colours.

The colour names that work are the same as the pen color and fill color blocks. (see the full list here (http://wiki.tcl.tk/37701)).

4.4.2. Problem: Fallout Shelter

The symbol for a fallout shelter is three **black** triangles on a **yellow** background.

Let's draw this symbol with the turtle!

Make the **pencolor dimgrey** so the edges stand out.

Each side of each triangle is **100 turtle steps** long and the triangles have all angles of **60 degrees**. The angle between each triangle is also **60 degrees**.

The output of your program should look like this:

The *centre* of the symbol is where the turtle starts.

> **Hint**
> You need to do an angle calculation to draw this shape!

Testing

- [ ] Testing the outside of the first triangle.
- [ ] Testing the outside of the second triangle.
- [ ] Testing the outside of the third triangle.
- [ ] Testing all three triangles.
- [ ] Testing the fill of the triangles.
- [ ] Testing for no extra lines.
- [ ] **Fantastic, but better stay away from radioactive materials anyway!**

5

ASKING QUESTIONS

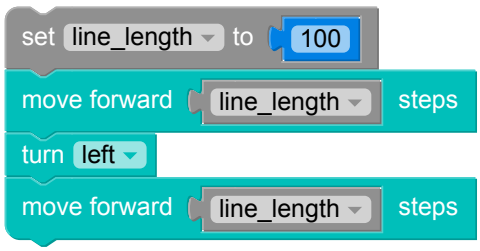
5.1. Variables and input

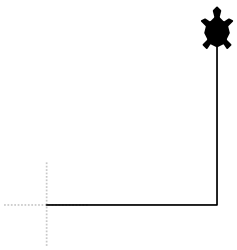
5.1.1. Remembering things

When drawing our square, we did the exact same thing 4 times: draw a line, and turn right. But if we made a typo and one side was longer than the others, it wouldn't make a square!

We need a **better way to remember and reuse** things like the side length.

We can use a *variable*! Each variable has a *name* which we use to set and get our value:





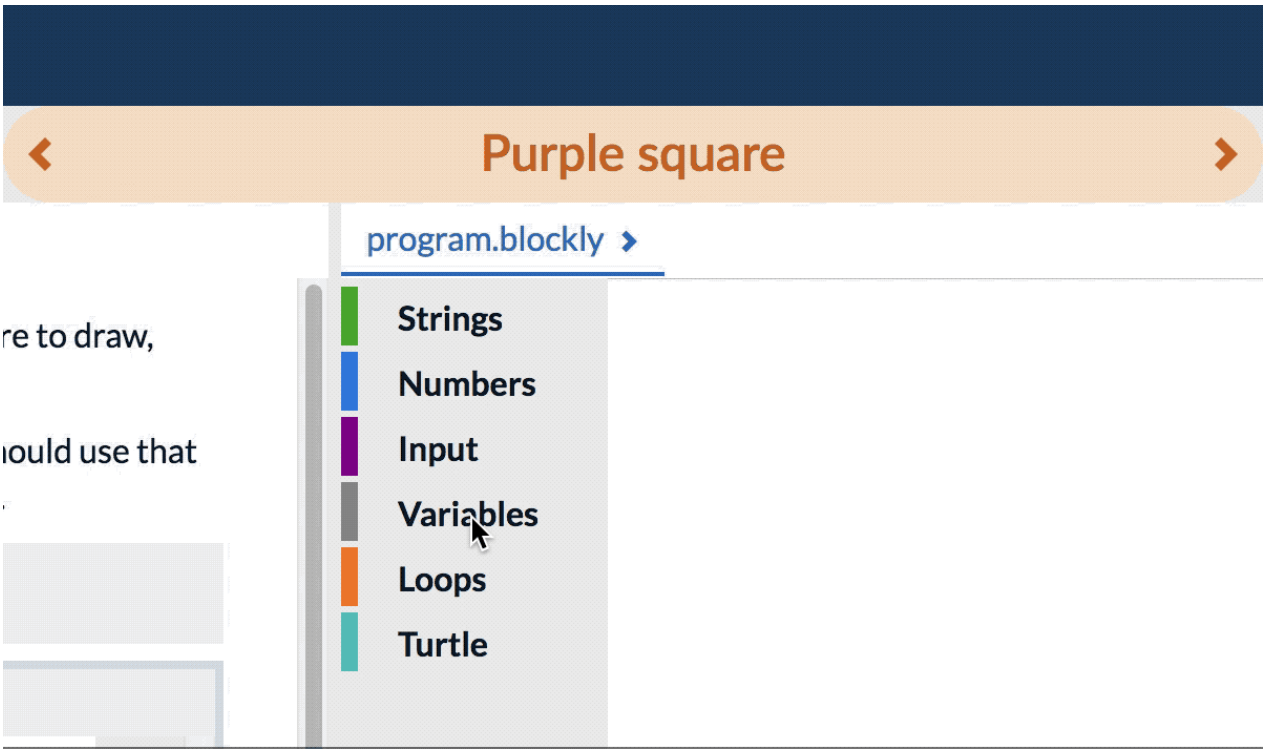
The set line_length block creates a new variable called `line_length`. It holds the value 100. We can then use the line_length block to use that number as often as we want.

Change the number in the set line_length block to different values, and you'll see that all of the lines in the drawing change to that value, and you only have to make one change to your code!

5.1.2. Using variables in Blockly

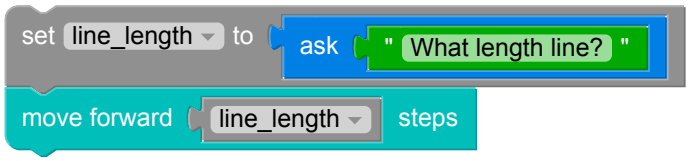
Creating a variable

To create a new variable, click the down arrow next to the variable name and select **New Variable...**



5.1.3. Asking the user for information

Variables save you from having to type the same values out a lot of times, but they also give us a way to store information that the user might give the program. Let's write a program that lets the user decide how long the line should be:



Run this program. Even if you haven't run any so far, run this one!

You will need to type a number and press `Enter`:

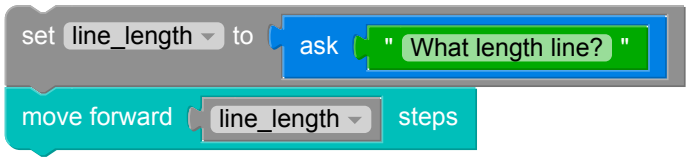
The ask block needs a question string to *ask the user*. It returns the user's answer to our program as a number. Our program stores the answer in the line_length variable so we can use it to draw later. You will find the ask block in the input menu.

Run it again with a different number. You should also try changing the wording of the question.

5.1.4. Different types of data

The computer uses words and numbers in different ways. This means we need a way to differentiate between them, and to declare whether we are using a string like " what length? " or a number like 10. We call these **data types**.

Blockly uses colour to show the data type: the green blocks are strings and the blue blocks are numbers.



We write a question in words, so we need to use a green string block for our question.

If we expect the user to type in a number, then we need to make sure that our ask block is blue, since we'll be using the user's answer as a number.

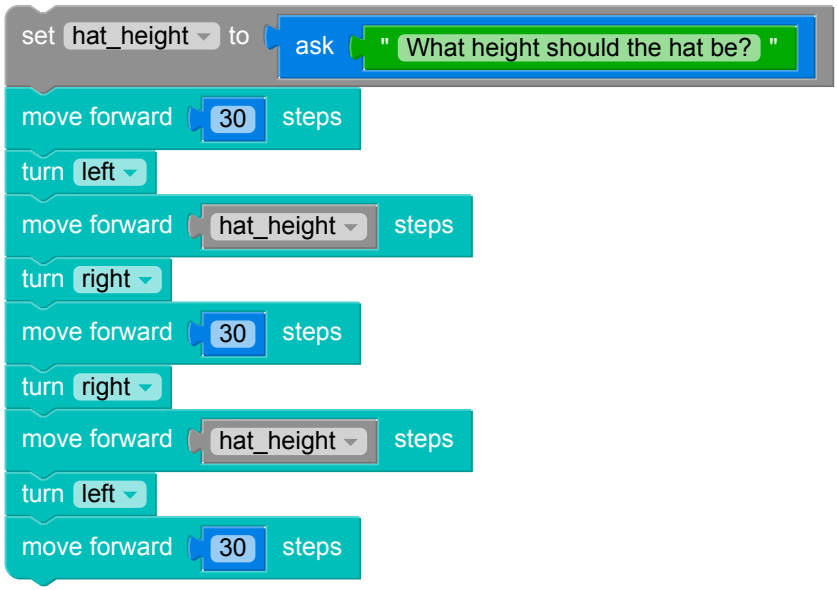
Since moving the turtle around means specifying distances, we need to use the blue blocks in our move forward blocks.

Variable blocks are special - they are grey because they take on the colour of the value last assigned to them - in this case, the blue of the ask block.

5.1.5. Custom shapes

Turtles would be boring if they drew the same shape every time.

Now that we can ask the user for input, let's ask them how high our top hat should be:



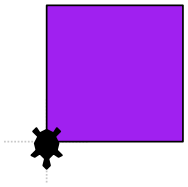
Try running this again and enter different heights!

5.1.6. Problem: Purple square

Write a program which asks the user what size square to draw, then draws it and fills it with purple.

The user will type a side length, and your program should use that side length input (in turtle steps) to draw the square.

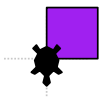
```
Side length: 80
```



The bottom left corner should be in the center of the page, where the turtle starts.

Here's an example where the user has chosen a much smaller side length.

```
Side length: 30
```

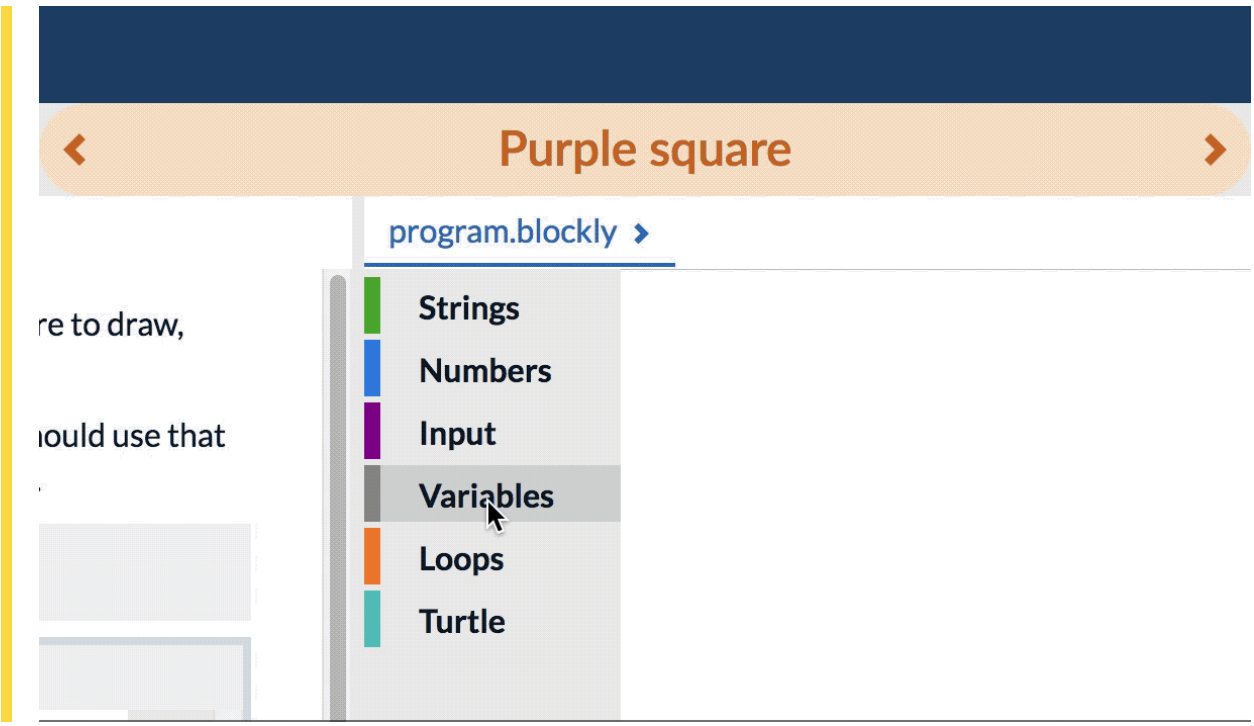


You should test your program with a whole range of different sizes! How big can the square be before it goes off the edge of the page?

Hint

The ask block is in the Input menu.

You'll need to use a variable!



Testing

- [ ] Testing the bottom line from the first example in the question.
- [ ] Testing the right hand side of the square in the first example.
- [ ] Testing the line at the top of the square in the first example.
- [ ] Testing the left hand side of the square in the first example.
- [ ] Testing the fill colour of the square.
- [ ] Testing there are no extra lines in the first example from the question.
- [ ] Testing the second example from the question.
- [ ] Testing there are no extra lines in the second example from the question.
- [ ] Testing a little square.
- [ ] Testing a big square.
- [ ] Testing a hidden test case.
- [ ] Testing another hidden test case.

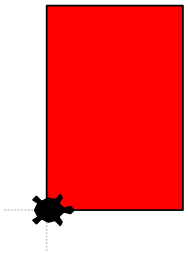
5.1.7. Problem: Coloured Cards

In many sports, coloured cards are used to let players know if a penalty has been awarded. For example, yellow cards are often a warning, and red cards send a player off.

Write a program which asks the user what colour card should be displayed, and draws it. The card should be **80** Turtle steps wide, and **120** Turtle steps tall.

We've given you a start!

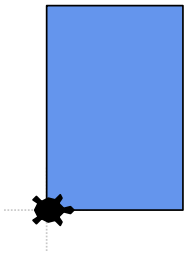
```
Colour: red
```



The bottom left corner should be in the center of the page, where the turtle starts.

Here's an example with an unusual colour.

```
Colour: cornflowerblue
```

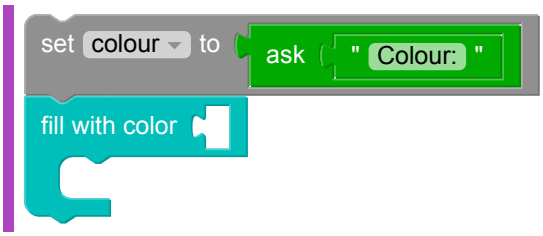


Hint

We will only test your code with colours that Turtle understands!

You'll need

program.blockly



Testing

- [ ] Testing the bottom line from the first example in the question.
- [ ] Testing the right hand side of the square in the first example.
- [ ] Testing the line at the top of the square in the first example.
- [ ] Testing the left hand side of the square in the first example.
- [ ] Testing the fill colour of the square.
- [ ] Testing there are no extra lines in the first example from the question.
- [ ] Testing the second example from the question.
- [ ] Testing there are no extra lines in the second example from the question.
- [ ] Testing a yellow card.
- [ ] Testing a hotpink card.
- [ ] Testing a hidden test case.

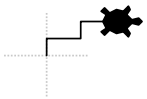
5.1.8. Problem: Step up

Write a program that draws steps. How many steps? That's the question!

Your program should ask the user how many steps to draw, and then draw them. Each step should be **10 Turtle steps tall**, and **20 Turtle steps wide**.

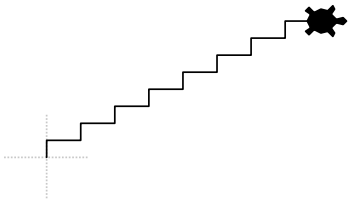
The steps will go up and to the right of the screen, as shown in the example below.

```
Steps: 2
```



Here's another example.

```
Steps: 8
```



Hint

How many steps you draw depends on the number you read in and save in a variable!

Testing

- [ ] Testing the bottom step from the first example in the question.
- [ ] Testing the whole first example from the question.
- [ ] Testing there are no extra lines in the first example from the question.

- [ ] Testing the second example from the question.
- [ ] Testing there are no extra lines in the second example from the question.
- [ ] Testing a small set of steps.
- [ ] Testing a big set of steps.
- [ ] Testing a hidden test case.
- [ ] Testing another hidden test case.

6

MAKING DECISIONS

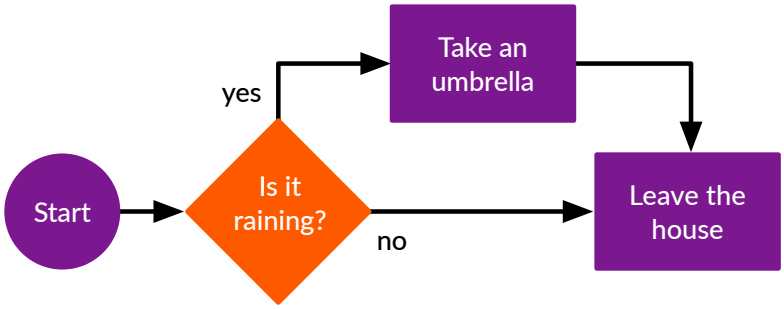
6.1. Making decisions

6.1.1. Why do we need decisions?

So far our programs have been just a sequence of steps that run from top to bottom. The programs *run the same way every time*.

In the real world, we **decide** to **take different steps** based on our situation. For example, if it's raining, we do an *extra step* of taking an umbrella before leaving the house.

This *flowchart* describes this process (or *algorithm*):



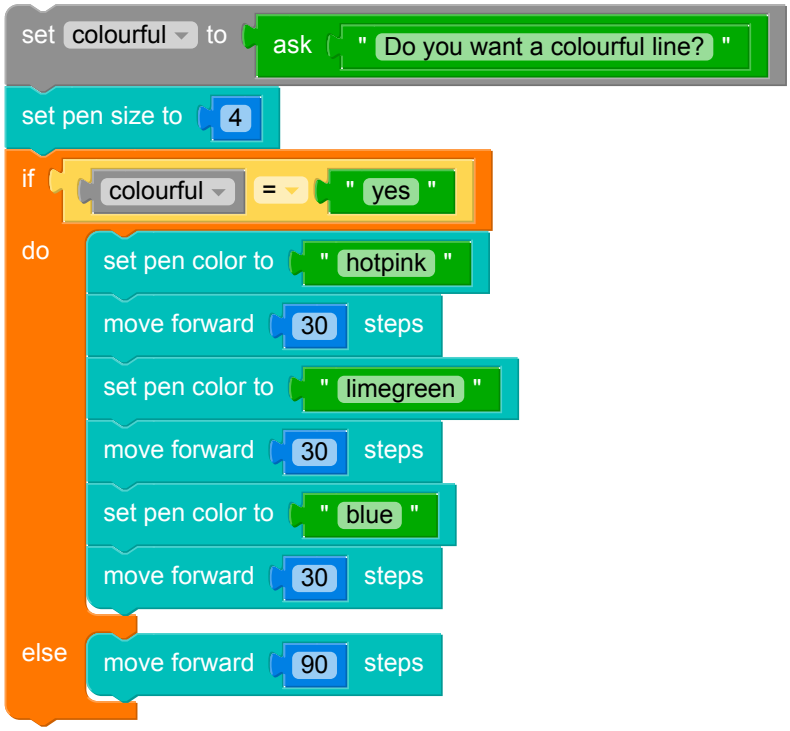
The diamond needs a `yes` or `no` decision. If the answer is `yes`, we do the extra step of taking an umbrella. If the answer is `no`, we skip it.

We can write this in Blockly using an if block.

6.1.2. Are two strings the same?

The computer decides what steps to run with an if block.

The computer might decide on what to draw based on what the user types in. Try answering `yes` and then `no` to this question:


set colourful to ask " Do you want a colourful line? "
set pen size to 4
if colourful = " yes "
do
set pen color to " hotpink "
move forward 30 steps
set pen color to " limegreen "
move forward 30 steps
set pen color to " blue "
move forward 30 steps
else
move forward 90 steps

6.1.3. Problem: Studybot - Highlighter or Pen

You're building a virtual robot to help you study. You've got it reading books, - the next step is highlighting and underlining notes!

Write a program that asks the user `Use a pen or highlighter?` and either highlights or underlines for **100 Turtle steps**.

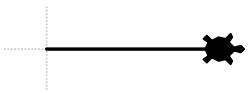
If the user types in `highlighter`, you should make the pen **yellow**, and pen thickness **15**.

```
Use a pen or highlighter? highlighter
```

Otherwise, you should draw a line to underline the notes. The line should be **black**, and pen thickness **2**.

Here's how the program should work if the user types in `pen`:

```
Use a pen or highlighter? pen
```



Hint

We'll only test `highlighter` or `pen` for this question!

Testing

- [ ] **Testing the highlighter.**
- [ ] Testing the highlighter has no extra lines.

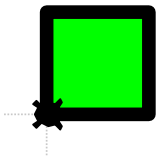
- [ ] **Testing the pen.**
- [ ] Testing the pen has no extra lines.
- [ ] **Nice work! Now get that robot studying!**

6.1.4. Problem: Traffic light

You're designing a traffic light system for robots! It's a square that is **60 Turtle steps** on each side.

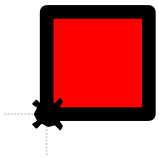
Write a program that reads in whether it is safe to go. If it is safe, you should draw a square with **green** fill and lines of thickness **8**:

```
Is it safe to go? yes
```



If it's not safe, you should draw a **red** square with lines of thickness **8**:

```
Is it safe to go? no
```



Hint

One way to solve this question is to use two variables! One for the user's answer, and one for the colour you should use to draw!

Testing

- [ ] Testing the bottom line from the first example in the question.
- [ ] Testing the right hand side of the square in the first example.
- [ ] Testing the line at the top of the square in the first example.
- [ ] Testing the left hand side of the square in the first example.

- [ ] Testing the fill colour of the square.
- [ ] Testing there are no extra lines in the first example from the question.
- [ ] Testing the second example from the question.
- [ ] Testing there are no extra lines in the second example from the question.
- [ ] **Celebrate! All signals are go!**

6.1.5. Problem: The Three Little Pigs

In the fairytale The Three Little Pigs (https://en.wikipedia.org/wiki/The_Three_Little_Pigs), the first pig builds a house out of straw, the second sticks, and the third bricks.

Write a program which asks the user whether they want to build using **`straw`**, **`sticks`**, or **`bricks`**, and then draws the house in the appropriate colour. A straw house should be **`orange`**, a stick house should be **`black`**, and a brick house should be **`slategray`**.

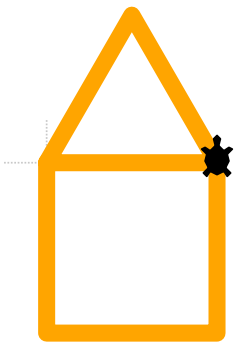
Material	Colour
`straw`	`orange`
`sticks`	`black`
`bricks`	`slategray`

The triangle at the top should have angles that are all 60°.

The triangle and square sides should all be 100 turtle steps long and 10 steps wide – using pensize.

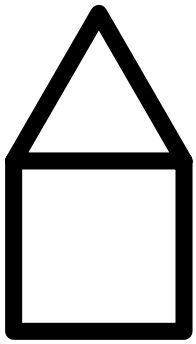
Here's a straw house:

```
Material: straw
```



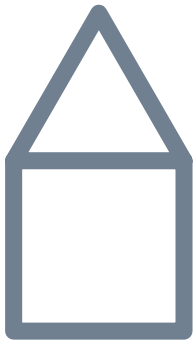
Here's a stick house:

```
Material: sticks
```



And here's a brick house:

```
Material: bricks
```



Testing

- [ ] Testing a house made of straw.
- [ ] Testing a house made of sticks.
- [ ] Testing a house made of bricks.

EXTENSION: PUTTING IT ALL TOGETHER!

7.1. Congratulations!

7.1.1. Congratulations!

Congratulations on making it through the course! You've learnt so much!

We've put together a few extension questions and some advanced Turtle tips and tricks to challenge you even more. Have a go, and also try out your skills in our Turtle Playground question at the end where you can draw whatever you like!

7.1.2. Problem: Heartbeat

If you've ever seen a movie where someone is in a hospital bed, you would have seen their heartbeat appear as a line on a heart monitor. When a heartbeat is even, it repeats the same pattern over and over again like this:

Notice how the turtle moves forward before it draws the first pulse.

Create a heartbeat pattern of 3 pulses that:

- Moves 20 steps forward before the start of the pulse;
- Turns 80° left to draw the start of the pulse;
- Moves 20 steps up to draw the start of the pulse;
- Turns 160° right at the top of the pulse;
- Moves 40 steps to draw the main part of the pulse;
- Turns 160° left at the bottom of the pulse;
- Moves another 20 steps to draw the last part of the pulse;
- Has a gap of 40 steps between each pulse (the hint will help with this).

Hint

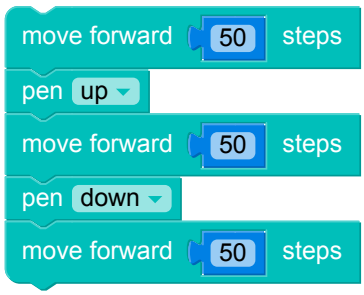
For this to work your turtle will need to face the same direction at the start and end of the loop, and the first and last thing it does is move forward 20 steps.

Testing

- [ ] Testing the shape of the heartbeat.
- [ ] Testing the gap between each beat.
- [ ] Testing the whole heartbeat.
- [ ] Testing there are no extra lines.

7.1.3. Pen up and down

Sometimes you'll need to move the turtle around into the right position without drawing anything. The pen up block does this:

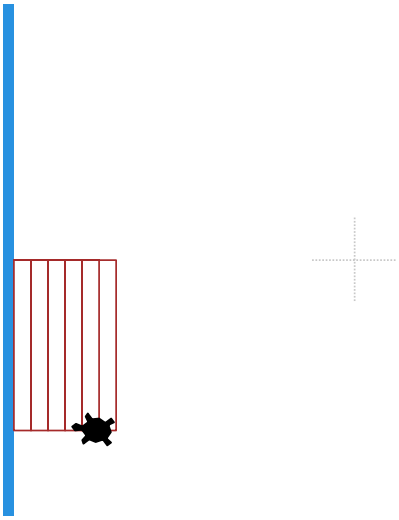


Imagine the turtle is holding a pen, after you've lifted up the pen with pen up the pen is off the paper and the turtle won't draw anything as it moves around. After you've put the pen back down with pen down, the turtle will draw as it moves again.

7.1.4. Problem: Wooden Fence

To build the whole fence it will take 40 planks of wood, but you don't have that many. Write a program to see how much fence you can build with the planks that you have:

```
How many planks? 11
```



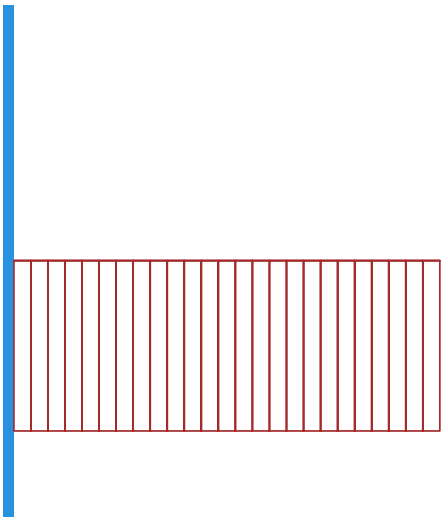
Your program should start off by moving 200 turtle steps to the left so that it starts off at the left-edge of the screen. Your program should work for any number up to 40 planks.

Important things to note:

- All lines should use pen colour " brown ";
- Each plank should be 100 steps high and 10 steps wide;
- The top of the fence should be the center of the space.

Here's another example:

```
How many planks? 25
```



Testing

- [ ] Testing the first example in the question.
- [ ] Testing the second example in the question.

- [ ] Testing with just two planks.
- [ ] Testing with three planks.
- [ ] Testing with 39 planks.
- [ ] Testing with all 40 planks.
- [ ] Testing a hidden case.
- [ ] Testing another hidden case.

7.1.5. Problem: Scaleable House

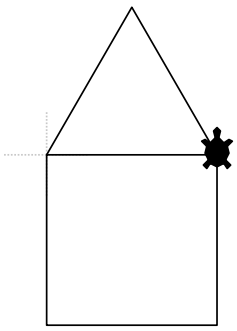
Let's draw a house for ants, or for giants!

You might have already learned about enlargement transformations - where we can take any shape and then scale it up by multiplying all of the sides by the same number. We'll use this transformation to draw a house of any size!

Write a program which asks the user what size the house floor is (in turtle steps), the scale they want the house to be transformed by, and then uses the turtle to draw the house.

The triangle at the top should have angles that are all 60° and all sides of the house should be the same number of turtle steps. Here's an example of a 50 turtle step house, scaled by 2 (so the sides all end up 50 × 2 = 100 steps long).

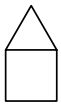
```
Size: 50
Scale: 2
```



The top left corner of the square is where the turtle starts.

Here's another example of a much smaller house:

```
Size: 10
Scale: 3
```



Copy Blocks

> Did you know you can copy blocks you've already made? Just right-click on the block in your answer and choose Duplicate!

Testing

- [ ] Testing the bottom of the roof (top of the square) in the first example from the question.
- [ ] Testing the right wall of the house in the first example from the question.
- [ ] Testing the left wall of the house in the first example from the question.
- [ ] Testing the floor of the house in the first example from the question.
- [ ] **Testing the room of the house** (the square) in the first example.
- [ ] **Testing the roof of the house** (the triangle) in the first example.
- [ ] **Testing the whole house** in the first example.
- [ ] Testing for no extra lines in the first example.
- [ ] Testing the second example house from the question (with 10 × 3 turtle steps).
- [ ] Testing a house where the size is 60 × 1 turtle steps.
- [ ] Testing a very very tiny house (3 × 5 turtle steps).
- [ ] Testing a hidden test case.
- [ ] Testing another hidden test case.

7.2. Turtle Playground

7.2.1. Problem: Turtle Playground!

Why not have a go at creating your very own drawing! You can write whatever code you like in this question. Consider it your personal playground.

We've even included a few extra blocks you might not have seen before. Try them out to see what they do! (You can even change how fast the Turtle moves!)

> **Save or submit your code!**
>
> There are no points to be earned for this question, so you can submit whatever code you like. Make sure you save programs that you want to keep!

Testing

- [ ] This question is a playground question! There's no right or wrong.

PLAYGROUND!

8.1. Turtle Playground

8.1.1. Playground Awaits!

This is the Playground Module. You can use these questions to draw whatever you'd like.

These questions aren't for points, just for playing, so have a go!

Make a work of art! Draw to your heart's content, and get those Turtles moving!

8.1.2. Problem: Turtle Playground!

Why not have a go at creating your very own drawing! You can write whatever code you like in this question. Consider it your personal playground.

We've even included a few extra blocks you might not have seen before. Try them out to see what they do! (You can even change how fast the Turtle moves!)

> **Save or submit your code!**
>
> There are no points to be earned for this question, so you can submit whatever code you like. Make sure you save programs that you want to keep!

Testing

- [ ] This question is a playground question! There's no right or wrong.

8.1.3. Problem: Turtle Playground!

Why not have a go at creating your very own drawing! You can write whatever code you like in this question. Consider it your personal playground.

We've even included a few extra blocks you might not have seen before. Try them out to see what they do! (You can even change how fast the Turtle moves!)

> **Save or submit your code!**
>
> There are no points to be earned for this question, so you can submit whatever code you like. Make sure you save programs that you want to keep!

Testing

- [ ] This question is a playground question! There's no right or wrong.

8.1.4. Problem: Turtle Playground!

Why not have a go at creating your very own drawing! You can write whatever code you like in this question. Consider it your personal playground.

We've even included a few extra blocks you might not have seen before. Try them out to see what they do! (You can even change how fast the Turtle moves!)

> **Save or submit your code!**
>
> There are no points to be earned for this question, so you can submit whatever code you like. Make sure you save programs that you want to keep!

Testing

- [ ] This question is a playground question! There's no right or wrong.