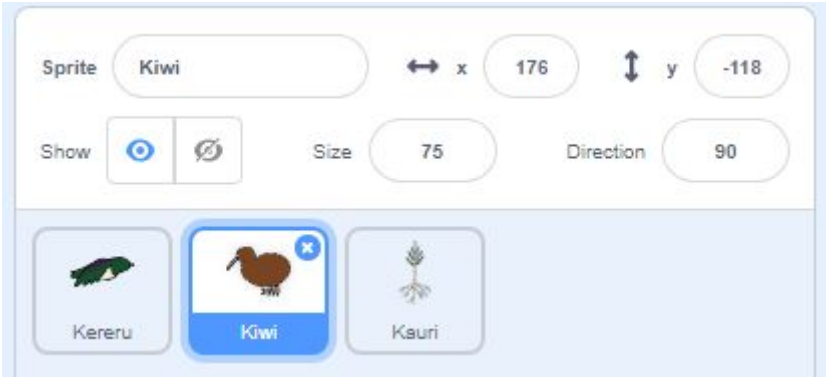
# The Kauri & the Kereru

## SESSION 1 – STARTING OUT

Head to https://scratch.mit.edu/projects/244879006/ and click 'See Inside.'

---




Make sure the kiwi sprite is highlighted.

We need to use a block to signify when our program will start. These can be found in **Events**. From **Events**, find the 'when this sprite clicked' block. Drag and drop the block into the grey programming area.


We next need to find a sound. Click on **Sounds** in the scripts tab. Most sprites have an associated sound, but not all. The kiwi sprite has a preloaded sound that has been uploaded from the DOC website.

In **Sounds**, find the block called 'play sound ____'. Drag and drop this block into the programming area. Drag the 'play sound ____' block under the 'when this sprite clicked' blocked. A grey shadow will appear under the 'when this sprite clicked' block when the 'play sound ____' block is close enough to snap together.


Congratulations! You have now written your first program! To test it, click on the kiwi to listen to its call!

Not working? If you didn't hear a sound, you will need to look for the problem. This is called debugging. Have the two blocks snapped together like two lego blocks? If not, try moving the sound block closer to the event block so that they snap together. Are your speakers turned on with the volume up?

Wrong sound? Some sprites have more than one option for sound. Click the drop down arrow in the 'play sound ____' block and make sure you have the sound you expect.

## Sequencing

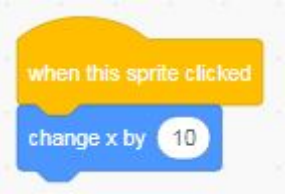
Make sure the kereru sprite is selected.


We can also make our sprites move. In **Motion**, there are two ways we can make a sprite move across the screen, 'move __ steps' and 'change x by __'.

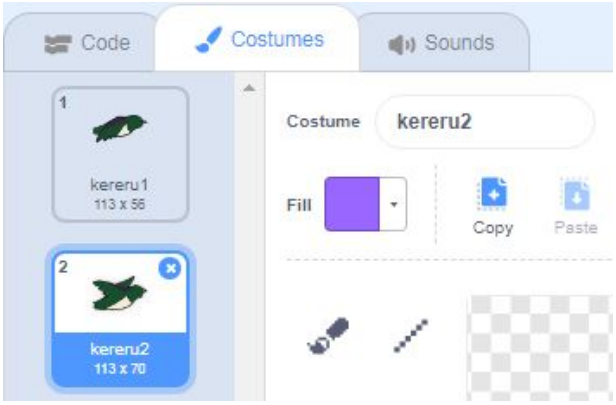
The 'move __ steps' block will move the sprite half the number of specified pixels in the direction that the sprite is facing (each step is half a pixel), while 'change x by __' will move the sprite the specified number of spaces along the x axis. (the x axis ranges from -240 to 240, the y axis ranges -180 to 180).

> What is a pixel? A pixel is the smallest unit of a digital display. An image displayed on a computer screen is made up of grid of tiny squares. Each square will be a different colour and together this makes up an image. In the scratch, the display area is 480x360 pixels.

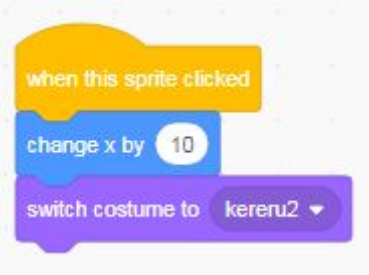
Here, we will use the 'change x by __' block. Drag and drop the block so that it is under the 'when this sprite clicked'. Now when you click the kereru, move to the right.



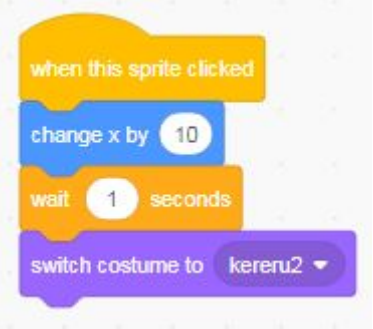

We can make the kereru look like it is flying by changing its costume. Head into the 'Costumes' tab. We can see that there are two costume of the bird in flight. We can write instructions so that the bird switches between the costumes.



Head back to the 'Scripts' tab. From **Looks**, select the 'switch costume to ____' block and drag it under your your 'change x by __' block. Using the drop down arrow, select kereru2.




At the moment, the sprite appears to move and change costume at the same time. To separate the two actions, head to **Control** and find the 'wait __ secs' block. Drag this block to that it is in between the two blocks.

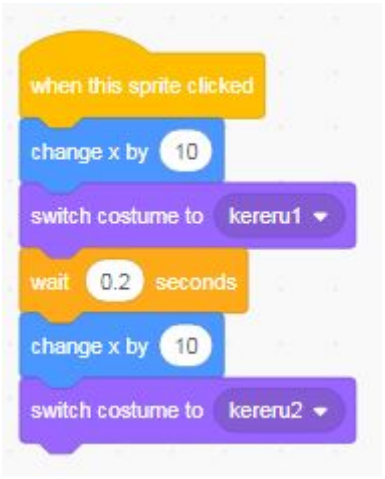



Try changing the order of the blocks. How does this affect the sprite? To speed up or slow down the program, change the number of seconds in the 'wait __ secs' block.

## Repetition

Sometimes we want to repeat a set of instructions. For example, we can get our sprite to move forward, change costume, move forward, and change costume again. To do this we can use another 'change x by __' block and another 'switch costume to ____' block.

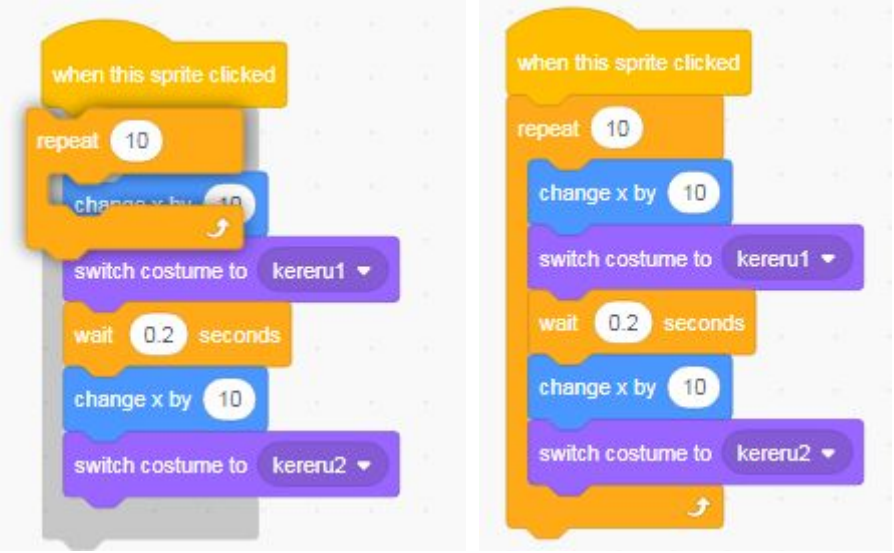
To delete blocks of code in Scratch, you can either right click on the block you want to remove and click delete, or drag and drop the blocks back into the centre code bar.




This is an easy way to repeat small sections of code, however if you want more than a few repeats, it would quickly become time consuming and messy. Luckily the computer has an easy way to repeat sections. In **Control**, there is a block called 'repeat.' Other blocks can be put inside this block, and will repeat the instructions for the number of time in the white area. This is called a loop.

> A loop is a section of instructions that are repeated until they are told to stop. The signal to stop can vary, for example if something has repeated a certain number of time or if a certain condition has been met.

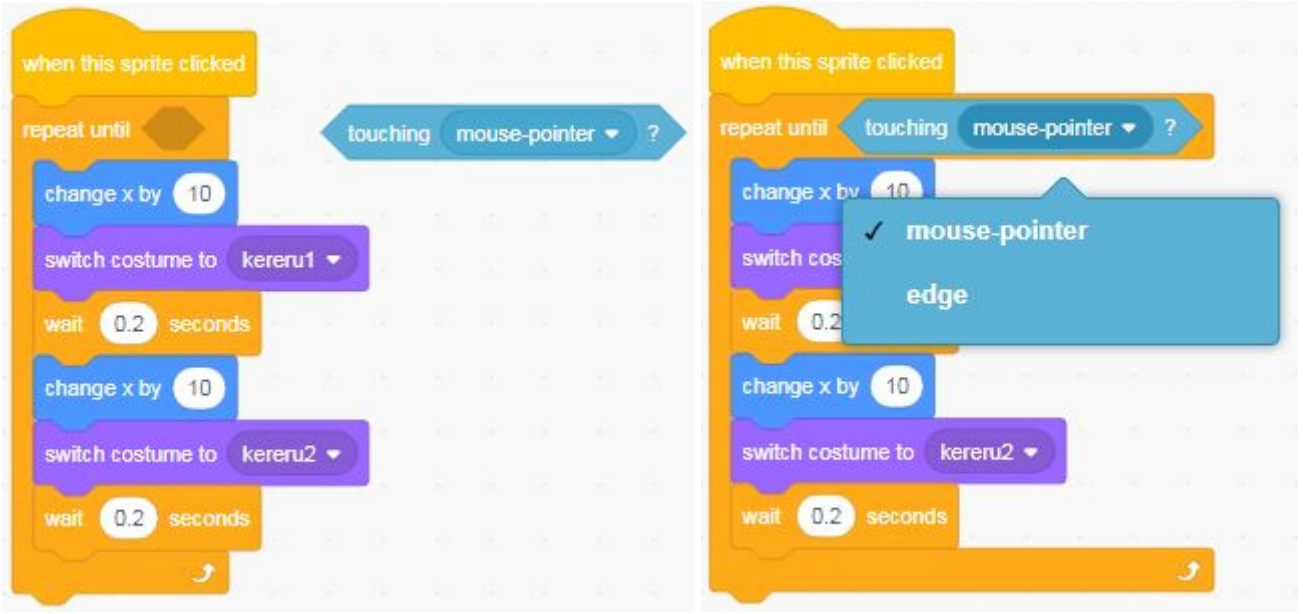
From **Control**, find the 'repeat __' block. Drag it to you program. The loop will automatically expand to contain your existing code.




> Extra: The Kereru's flight is not very smooth. Add a second wait block to improve the look of the movement

If we want our sprite to move across the whole stage, we could use the repeat and guess how many times we want to repeat our instructions, or we could use a 'repeat until ____' block.

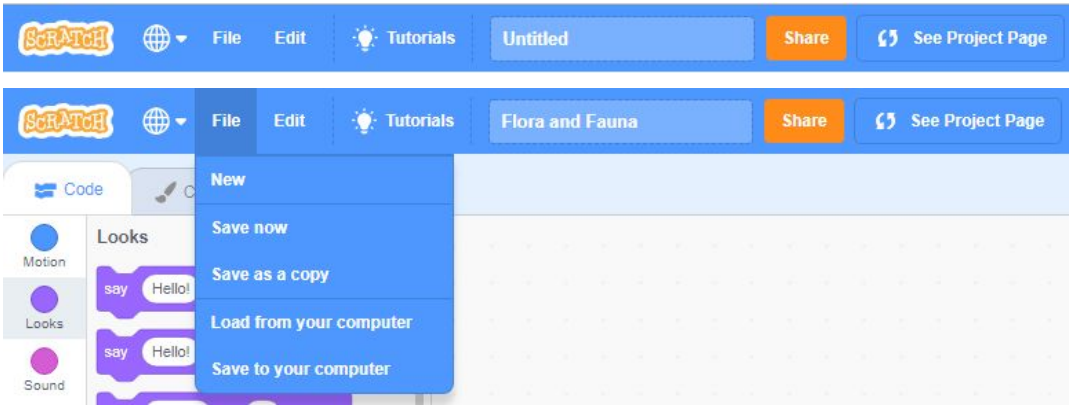
The 'repeat until' block requires us to provide an extra piece of information, which is when we want it to stop repeating. We can tell Scratch to continue repeating the instruction until the sprite reaches the edge of the screen, by going into **Sensing** and selecting the 'touching ____' block. Drag the 'touching ____' block into the space in the 'repeat until ____' block. Click on the small downwards arrow to open the drop down menu and select edge.




Extra: Explore the blocks in Looks, Motion, and Sound. Can you make the kereru and the kiwi have a conversation?

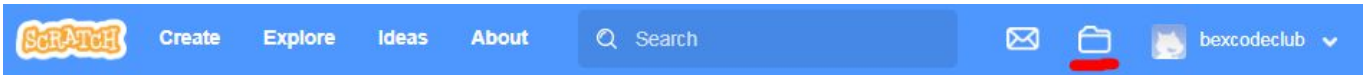
## Saving your work

It is important that you remember to save your work at regular intervals. Change the name of the file in the bar above the stage. Click file, Save now to save the project.



# SESSION 2 – ADDING DETAIL

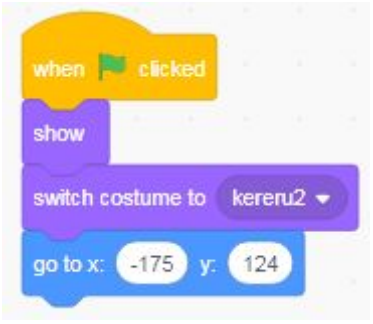
## Open your work




To open your work from last week, log in to scratch. Click on the folder icon next to your username to open 'your stuff'. Find the project from last session and click 'see inside'.

## Reset button

As our program gets more complicated, it can be useful for us to program in a reset button. From **Events** find the 'when flag clicked' block and add it to your programming area for every sprite. For each sprite, add 'go to x:__ y:__' block with the starting coordinates for that sprite and either a 'hide' or a 'show to the 'when flag clicked' block. It can also be useful to add a change costume to the starting costume.



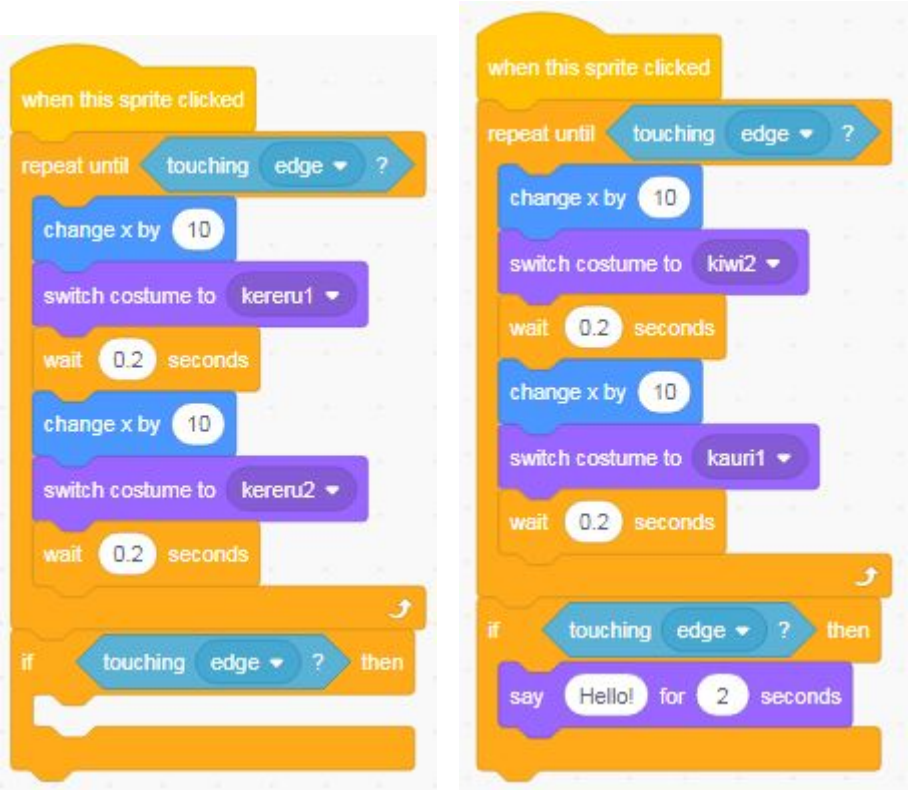

## Conditional statements

A conditional statement is a statement that a computer uses to help it make a decision.

We are going to create some code that causes something to happen when the kereru reaches the edge of the stage. From **Control**, find the 'if ___ then' block. Add it to your kereru script along with a 'touching ____' block from **Sensing** under your repeat loop.

Add a block from **Looks** or **Sounds** to test that your program works.

We use conditional statements all the time, for example "if you pick up all your toys, then you can watch tv" or "if you eat your broccoli, then you can have an ice cream or else you'll have to have an apple".




Extra: Can you replace the 'repeat until ____' block and 'if ____ then' block with an 'if ____ then else ____' block?
Tip: You will also need to use a 'forever' block and a 'stop ___' block.

## Broadcasting messages

Sometimes, we want to be able to send a message between sprites. This will pass a message to all the sprites and backgrounds in use and can be used as a signal for something to start or stop happening. To use broadcasting, we need to create an event that triggers a message. We can use our conditional statement as an event. From **Events**, find the 'broadcast _____' block and drag it into the if block. Click on the drop down arrow and click 'new message'. Name your message 'Plant Seed'.

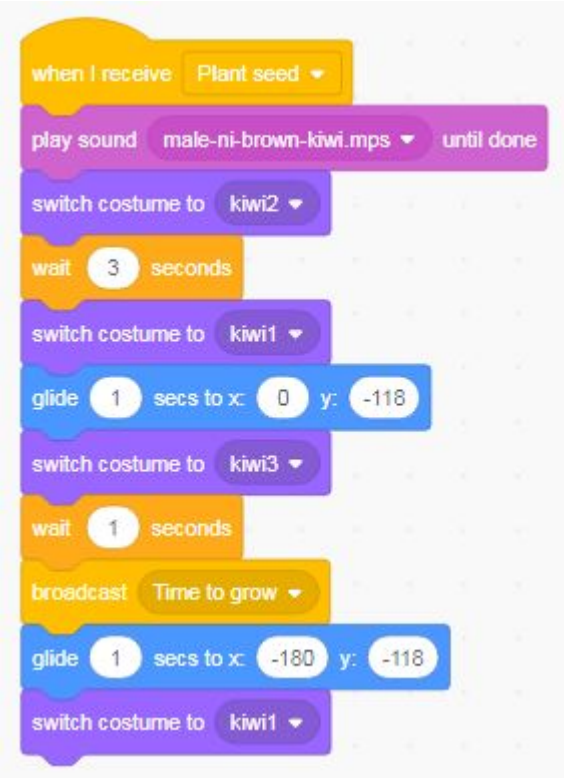

In the programming area of the kiwi, find the 'when I receive _____' block (also from **Events**). We can use code from **Motion**, **Looks**, or **Sounds** to make the sprite or background do something when it receives

the broadcasted message. Rearrange the kiwi's code so that it makes its call when it receives the message from the kereru.


> Extra: Try adding an extra block so that the kiwi changes its costume that it looks up at the kereru as it calls, and then looks back down again.

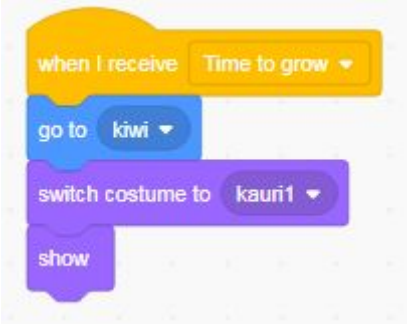
When the kiwi receives its message from the kereru, we want it to plant a kauri seed in the centre of the stage. Use the following blocks of code to add to your kiwi's code so that it walks across the stage and plants a seed.


We can use broadcast again to send a second message, this time to the kauri sprite which is currently hidden. Add a 'broadcast ____' block to the kiwi that sends a message whilst the kiwi is planting the seed.




Not working as expected? Check if you have you created a new message or are you broadcasting the same message again. When you re-broadcast the first message, it will re-trigger all the events you have programmed that happen after that signal!

We need the kauri sprite to become visible when it receives the message from the kiwi. Add a 'show' block from **Looks** to a 'when ____ received' block. At the moment, we aren't controlling where the kauri appears. We can use a 'go to ____' block from **Motion**, to get the kauri to meet the kiwi at the spot that it stops before it appears. We can also use a 'switch costume to ____' block to ensure that kauri is in seed form when it gets planted.



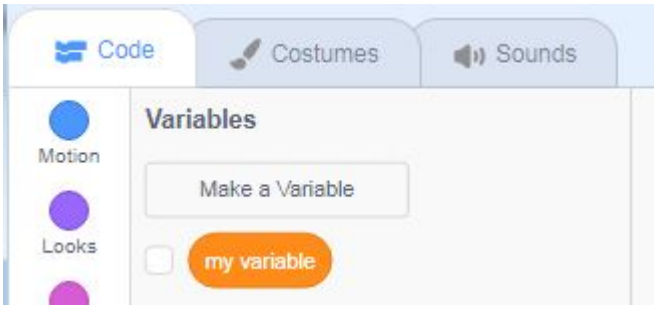

Extra: Can you add a new Sprite and turn it into a reset button by using the broadcast function?

# SESSION 3 – BUILDING COMPLEXITY

## Variables

A variable is a way for the computer to store small chunks of information like a name or a number. In Scratch, the default variable type is a number

Now that the kauri seed has been planted, we need to help it grow! Plants need lots of sunshine in order to grow big and strong. We're going to include sunshine in our program to ensure that our kauri will grow. Head into **Variables** and click make a Variable. Name your new variable Sunshine.

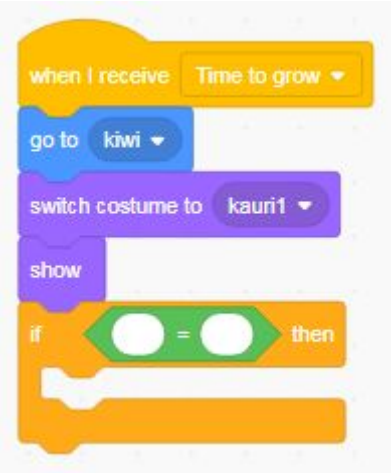




If there is lots of sunshine, we want our kauri tree to change its costume so that it looks like it is growing. To do this, we need a way of increasing and decreasing the number stored. From **Events**, find the 'when ___ key pressed' and drag it into the programming area. In **Variables** find the 'change ____ by __' block and drag it underneath the 'when ____ key pressed'. Adjust the values so that the variable, Sunshine, increases by 1 when you press the up arrow key.


Extra: Can you get the variable to decrease in value when the down arrow key is pressed?

We are now ready to write a program that changes the costume of the kauri depending on the value of the variable, Sunshine. Find an 'if ___ then' block, and attach it to the bottom of your 'when I receive ____' section. Head into **Operations**, find the '__ = __' block and slot it into the space in the 'if ____ then' block.




If the value of our variable, Sunshine, reaches a certain level we want the kauri to change its costume. From **Variables**, find the variable block, called Sunshine and drag it into the first space in the '__ = __' block. Type the number 5 in the other space. Add a 'switch costume to ____' inside the 'if ____ then' block.


Add 5 'if ____ then' blocks to change the costume depending on the value of the variable.

> Does the costume change as you increase the value of the variable? To get the program to constantly change depending on the value, add in a 'forever' loop.


## Boolean Operators

In **Operators**, there are six green blocks that are hexagonal in shape. These blocks (and every hexagonal block in Scratch) are Boolean Operators. These blocks give simple conditions that provide the computer information with which it can use to make a decision. They are commonly used with conditional statements like the 'if ____ then' block.

At the moment, our tree will only change costume when the variable is equal to a certain value. Using '__ > __', '__ < __', and the '__ and___' blocks, change the code so that the tree changes its costume when the variable is within a range of values.


when I receive Time to grow
go to kiwi
switch costume to kauri1
show
if Sunshine < 5 then
switch costume to kauri1
if Sunshine < 5 and Sunshine > 10 then
switch costume to kauri2
if Sunshine < 10 and Sunshine > 15 then
switch costume to kauri3
if Sunshine < 15 and Sunshine > 20 then
switch costume to kauri4
if Sunshine > 20 then
switch costume to kauri5

# SESSION 4 - THE NEXT LEVEL

## Modulo

We currently have have our kauri tree growing dependent on how much sunshine it gets, but in reality it takes years for a mighty kauri tree to grow big and strong. We're going to introduce seasons into our program so that the tree only grows once per year.

Create a new variable called Seasons. We want our tree to only change costume once per year, or once for every four seasons. To do this, we need to be able to identify when the value of the seasons variable is divisible by four. In Scratch, we have a block designed especially for this. In **Operations**, find the '__ mod __' block. Mod stands for 'modulo', and is a mathematical operator that can be used to find the remainder after division. We can use this with an '__ = __' block and an 'if ____ then' block to create a condition that something happens when the remainder of a division is equal to a particular value.

Not working? Head to the costume tab, are your Sprite's costumes in the order you wanted?

We want to create a section in the programming area for the kauri that changes the seasons variable by one when a key is pressed, and then checks if the season is a multiple of four. If it is, we want to change the costume to the next costume.

# More variables

We can keep track of the year with a variable called, ‘year’. If four seasons have passed, we want to change the year by one:


We can get the sprite to the change its costume based on the year, so remove the ‘next costume’ block.

To the bottom of your section of the program, add another ‘if ____ then’ block, and place an ‘__ = __’ block in the space. If we plant our kauri seed in the year 2018, we’d expect the first signs of growth in 2019. Add 2019 into the ‘__ = __’ and a ‘switch costume to ____’ block into the if statement.

Add in more if statements so that the kauri grows (changes costume) every year.

> Extra: Use ‘change size by __’ blocks to write a section of code that makes the final costume of the kauri get bigger with each passing year.