CoderDojo

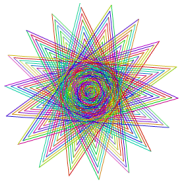
PEN TOOL

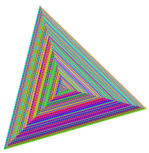
Card 1 of 5

I'm Learning: Scratch

1

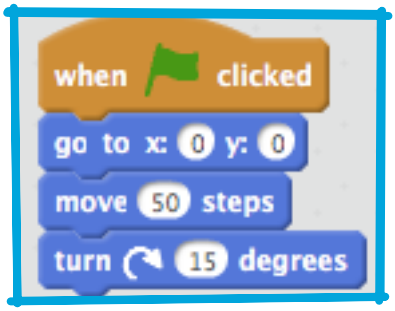
Now you've learned the basics of Scratch (and if you haven't, check out the Beginner Scratch Sushi Cards) and made your first Scratch game. In this series you're going to learn a few more cool tricks and make one of my favourite Scratch projects: It draws colourful patterns and, if you set it up right, can be really cool to watch.





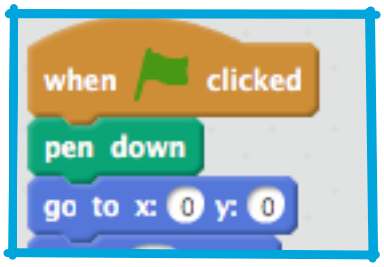
It all relies on the **pen** tool, which is controlled by blocks in the **pen** group. These blocks draw a line behind the centre of a sprite as it moves. You're going to learn to use it now!

Start a new Scratch file, select the Scratch Cat **sprite** and drag in a few of the blocks you're familiar with from the beginner cards, until it looks like this:



2

Cool! Now, time to test out the pen! From the **pen** section, select the "pen down" block and add it to the start of your program, like this:

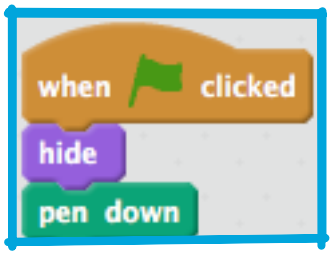


Now click the green flag a few times and watch what happens!

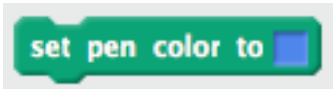
CoderDojo

3 If you can see the lines behind the cat then the pen is working and you can start making it draw really cool patterns!

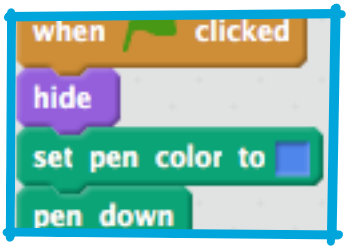
First, you should get rid of the cat! It's getting in the way of the drawing! Just add a "hide" block from **looks** to the start of the program and it'll disappear.



4 Now, you can change the colour of the pen with another block from the **pen** section, but this block is a little different to the others you've seen. It's the "set pen color to " block and looks like this:



Drag one out onto your **sprite panel** and snap it in ahead of the "pen down" block.



Now, click on the box of colour (in the code above it's the blue one) and you'll see the mouse pointer turn into a hand. Move it around the screen and notice how the colour in the box changes to be the colour of whatever is underneath the pointer! When you click, Scratch will save that colour as the colour of your pen.

5 If you've been clicking on the green flag to test your code, you'll have noticed that the drawings made by the pen don't go away. Add a "clear" block from the **pen** section to the start of your code to take care of that!

CoderDojo

DRAWING LOOPS

Card 2 of 5

I'm Learning: Scratch

1

Now you've got a program that draws a line, but it only draws one line. That's a bit dull! You can use a loop, like "forever", to draw over and over again. Of course, if you just use "forever" then you'll get drawings that go off the stage!

So you want to use a different loop, which you'll also find in the **control** section, called "repeat until" which will do something over and over again, until a true/false condition becomes true. Wrap it around your "move" and "turn" blocks like so:

2

Now click the green flag to run the program a few times and see what happens! You'll notice two things: It always starts by drawing a line into the middle of the **stage and i**t doesn't stop at the edge .

The first is because the first **motion** block that runs after the "pen down" is "go to x: 0 y: 0". You can fix this just by moving the "go to x: 0 y: 0" block before the "pen down" block and adding, from **pen**, a "pen up" block right at the start of your code.

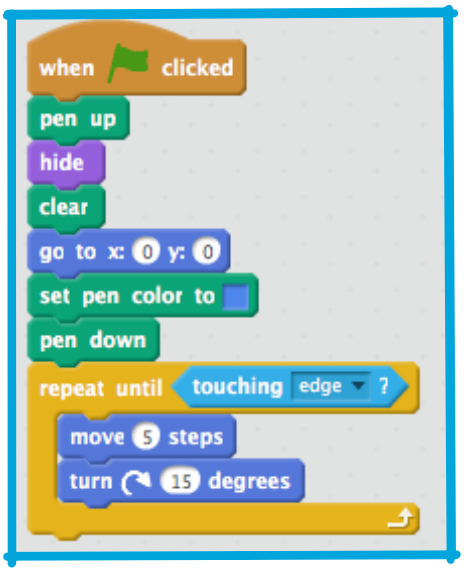
The second is because you haven't yet told it what it's checking, so the answer will never be true. Basically, right now, it's working like a "forever" loop.

3

Time to fix your "repeat until". You're looking to figure out if the (invisible) sprite is touching the edge of the **stage**, so you need a **sensing** block. In this case, the "touching ?" block. Snap it into your "repeat until" and select "edge"

CoderDojo

4 Change the number of steps in your program to 5 and check that it matches this one:



5 When you run the program now you'll see that it has turned into a circle drawing program! At least it stays on the **stage**! The problem here is that those 15 degree turns eventually add up to 360 and you turn a full circle. What needs to happen is that you take slightly longer steps each time, so you spiral out. For this, you're going to need a **variable**.

You've seen **variables** before, in the Beginner series. They're basically labeled places to store numbers that you care about. You can create them in the **data** blocks category and then find their associated blocks there too.

Make a **variable** called "steps" and use its value instead of the 5 in "move 5 steps", then add "set steps to 0" at the start of your program and "change steps by 1" as part of your loop (does it matter where you put it?).

Now run it, and try changing the number of degrees around (try 76 and 120)!

CoderDojo

DRAWING LOOPS

Card 3 of 5

I'm Learning: Scratch

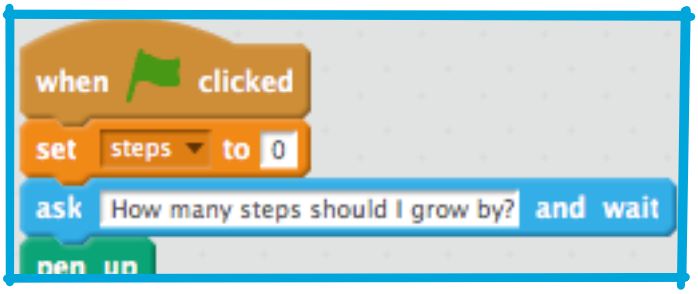
1

Ok, this is getting pretty cool, but it's a bit of a headache to have to edit your code every time you want to see a different pattern. Wouldn't it be good to get the program to ask you for them? You can do that!

First, create **variables** in **data** for "degrees" and "increase" and add them to your code like this:

2

Now you need to ask for values for these two **variables** and store them. You do this using a sensing block called "Ask and wait", which you can type a question into. Pull one into your **sprite panel** and change the question to "How many steps should I grow by?", then add it to your program, just after you set steps to 0, like this:

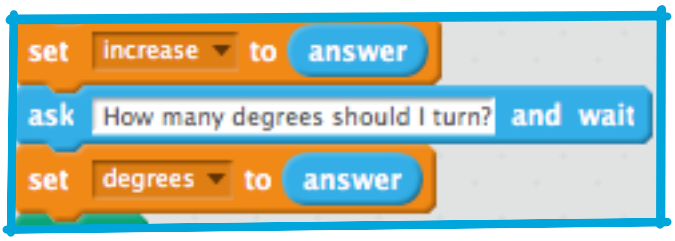


3

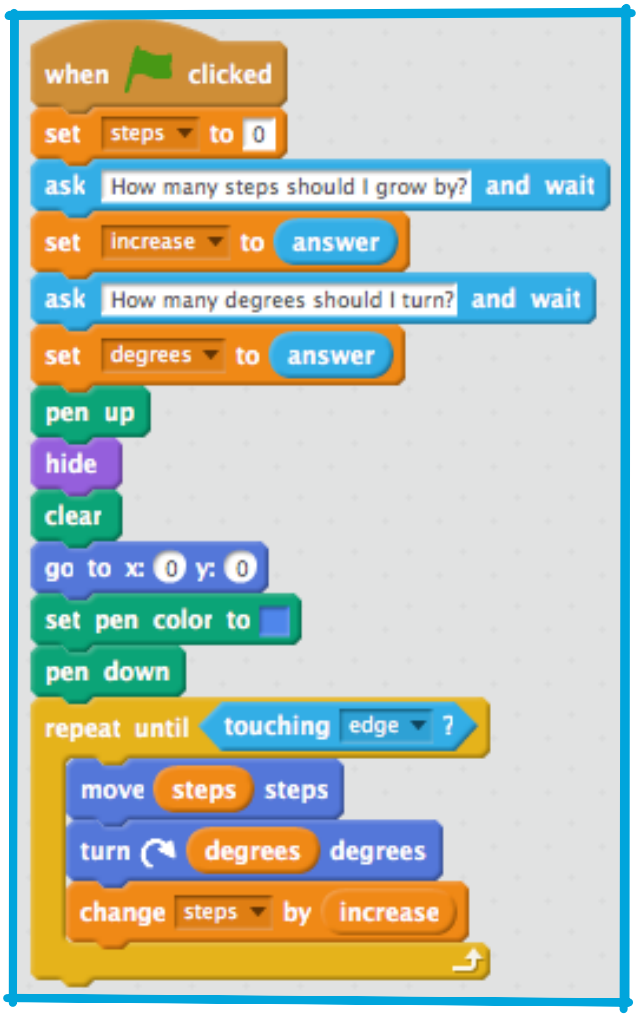
Once you've got Scratch asking a question, you need it to remember the answer! It turns out that Scratch has a special **variable**, called "answer", where it puts the most recent answer it's received. You can find it among the **sensing** blocks. Using a **data** "set to " block, take the value from "answer" and give it to "increase" like so:

CoderDojo

4 Now, do the same thing with "degrees", asking "How many degrees should I turn?" and storing the answer in "degrees".



5 Check your program now looks like the one below and run it a few times, trying different numbers. Write down the answers that make the coolest pictures. You'll need them on a later card!



CoderDojo

COOLER LINES

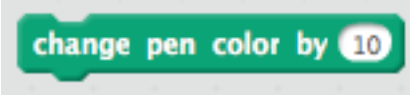
Card 4 of 5

I'm Learning: Scratch

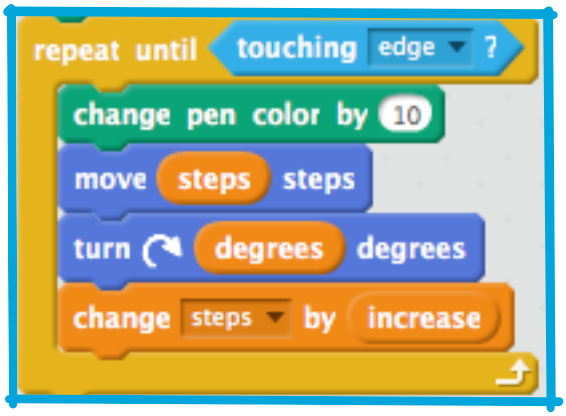
1

Time to add colour! Right now, your line is one colour, but the **pen** has blocks that can change its colour. With the right **operator** blocks, you can change it randomly.

The colour changing block you're looking for is "change pen color by ":

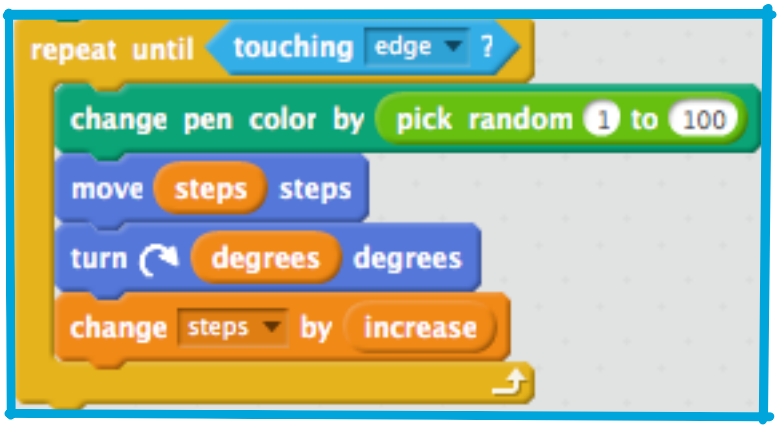


Grab one of those and put it into your "repeat until" loop, like this:



2

That's cool, but a bit predictable. You can make it a bit more fun if you add a random number into it, so the colour changes randomly. Just put the random number operator into the "change pen color by " block and pick some values to go in it. I'd try 1 and 100 to start.



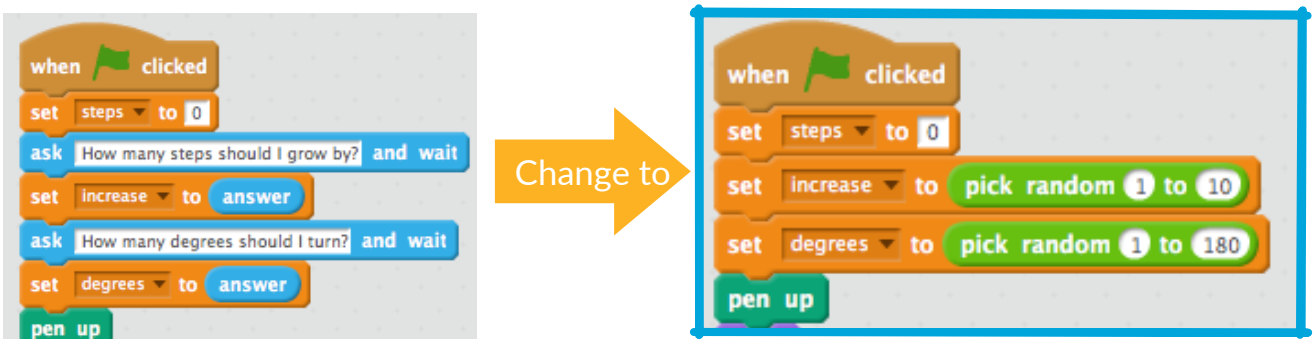
Try running it again, watch the random rainbow!

CoderDojo

3

You can actually use random numbers to make the whole program run itself over and over, changing the pattern each time! It'll look a bit like screen savers did in the 1990s... which you won't remember, but ask one of your Dojo Mentors!

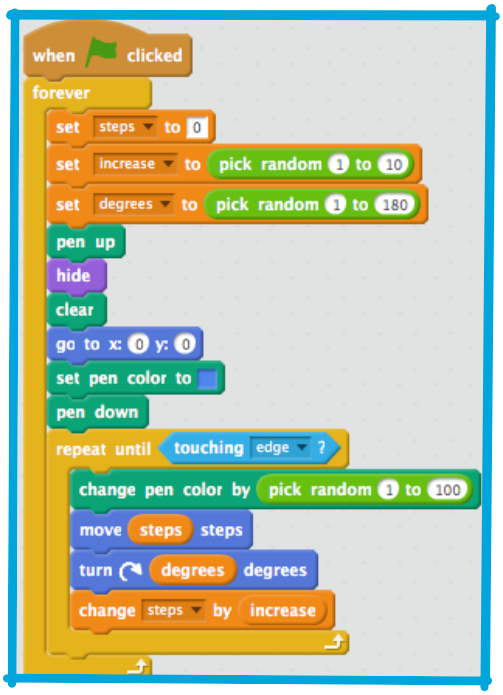
You need a few changes to make this happen. The first one is that you need to set the "increase" and "degrees" variables randomly, rather than asking for them from the user. So you need to change some of your code blocks:



4

If you run that, you'll find that the program does draw randomly, but only once. Why do you think that is?

It's because the loop only runs until it reaches the edge of the **stage**. You need *another* loop, that runs forever (so a "forever" block then!) outside the current one to keep it going over and over! Just drag one out of **control** and wrap your code in it.



CoderDojo

HELPING THE COMPUTER

Card 5 of 5

I'm Learning: Scratch

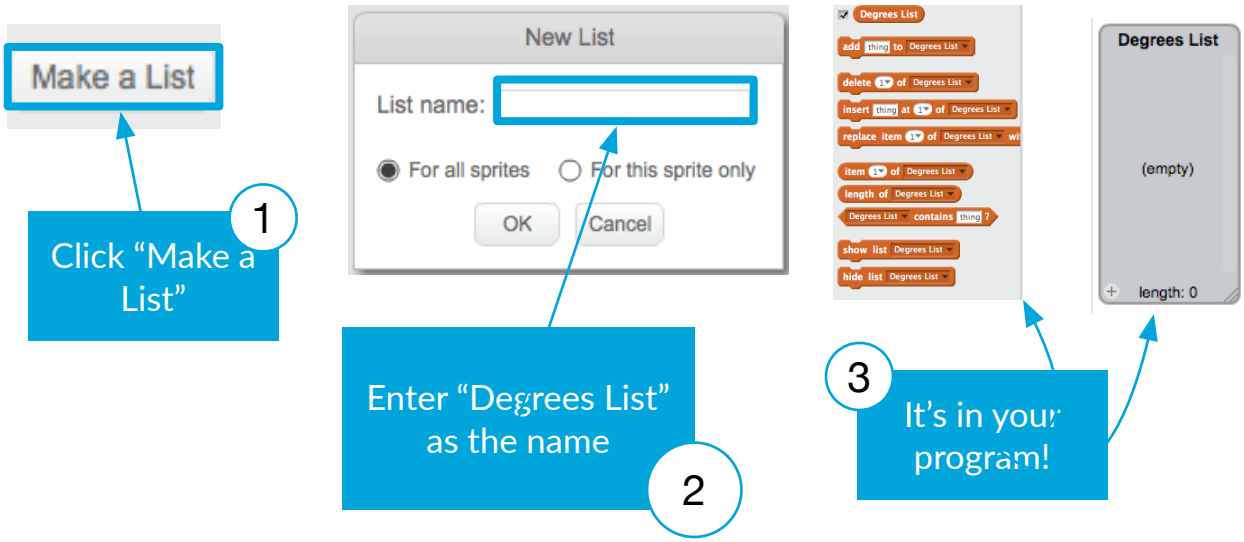
1

Now you've really got something awesome to look at! However, you may notice that, every now and then, the computer draws something that looks pretty... bad. This is because some numbers for some of those variables are just bad choices, and some *combinations of those numbers* are also bad choices.

Do you remember a few cards back when I told you to write down some of your favourite values for "increase" and "degrees", the ones that gave the best looking pictures? If not, don't worry, you can just watch the random program run for a while and write down the combinations that give great results.

You're going to teach Scratch those combinations of values, so it can use them to make nothing but awesome pictures!

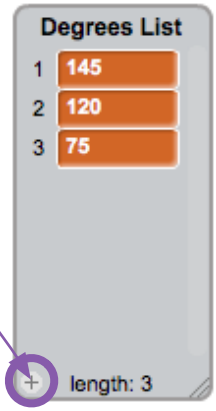
To do this, you'll need a **list**. You'll find them with **variables**, under **data**. Just like you did with the variables, you'll need to create your list.



2

Make another one called "Increase List" and then, by clicking on the little plus (+) at the bottom of the lists, type in the pairs of numbers.

Make sure that the degrees value at a position in the Degrees List matches the increase value at the same number position in the Increase List!



CoderDojo

HELPING THE COMPUTER

Card 5 of 5

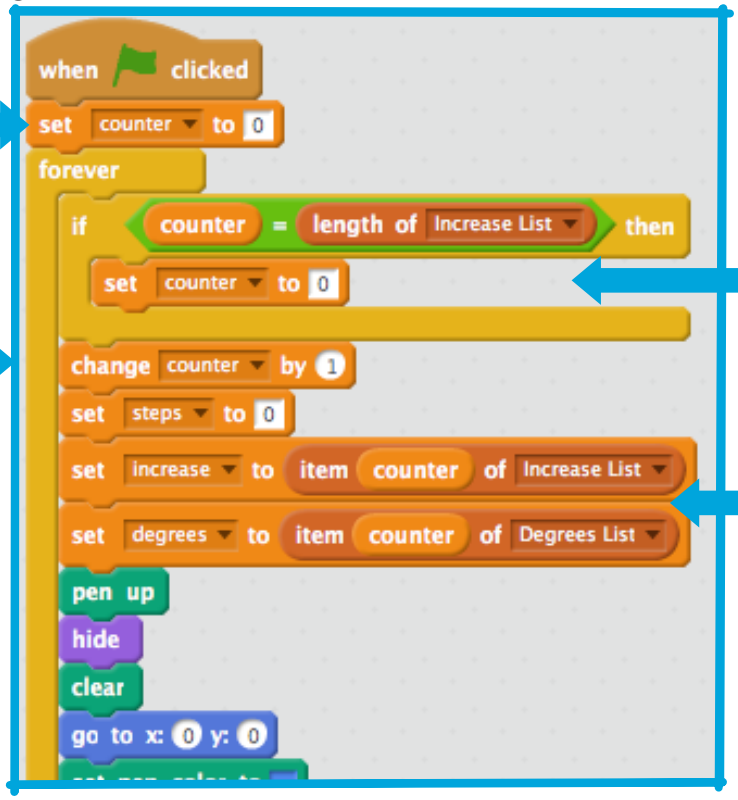
I'm Learning: Scratch

3

You have lists, now you just need to get your code to read them and loop over them! To do this, you're going to use a new **variable** called "counter", some incrementing and an "if then" **control** block. Just update your code to look like this one and you'll get it:

Set counter to 0, outside all the loops

Add 1 to counter



Check if counter is the length of the list. If so, set it to 0. This means that counter will always be the number of an item on the list.

Pick the counter-th item from Increase List and put it in the increase variable. Do the same for the Degrees List and degrees variable.

If you want to understand what's going on

Imagine your lists only have two pairs of values on them. This is what happens:

1. Set *counter* to 0
2. Start the *forever loop*
3. Check *if counter* (0) is the same as the length of *Increase List* (2). It isn't
4. Change *counter* by 1. Now *counter* = 1
5. Set *steps* to 0
6. Get the get the item at position *counter* (1) in the *Increase List* and put it in *increase*
7. Get the get the item at position *counter* (1) in the *Degrees List* and put it in *degrees*
8. Do all the stuff related to drawing the patterns
9. Restart the *forever loop*
10. Check *if counter* (1) is the same as the length of *Increase List* (2). It isn't
11. Change *counter* by 2. Now *counter* = 2
12. Set *steps* to 0
13. Get the get the item at position *counter* (2) in the *Increase List* and put it in *increase*
14. Get the get the item at position *counter* (2) in the *Degrees List* and put it in *degrees*
15. Do all the stuff related to drawing the patterns
16. Restart the *forever loop*
17. Check *if counter* (2) is the same as the length of *Increase List* (2). It is!
18. Set *counter* to 0
19. Continue from **step 4** of this list, in a never-ending loop!