

# Chatter coding – first steps

## Introduction

## Installation

### Welcome to the Chatter coding tutorial

Thank you for supporting CircuitMess, and welcome to the Chatter coding tutorial.

We'll use **CircuitBlocks** for coding your newly-assembled encrypted wireless communicators.

CircuitBlocks is a custom-made coding app that we've designed.

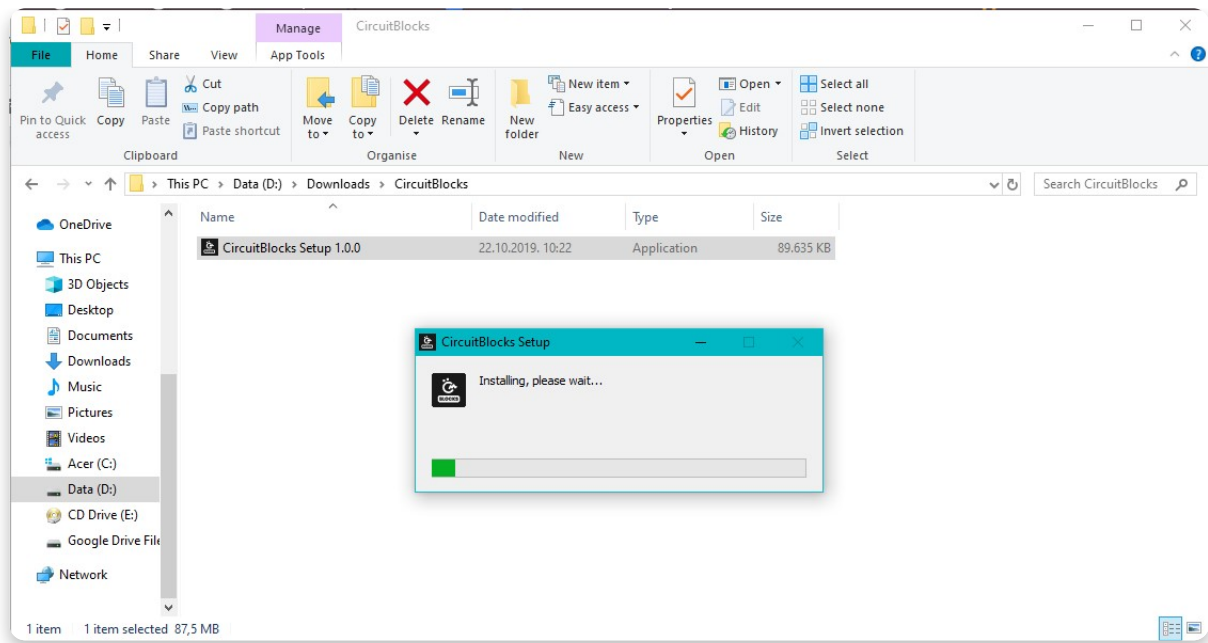
You will code your Chatter in CircuitBlocks' graphical block-based coding interface that will help you make your first steps in the world of physical computing.

## Installation

CircuitBlocks currently runs on Windows, Linux, and Mac OS computers.

### If you have a Windows computer

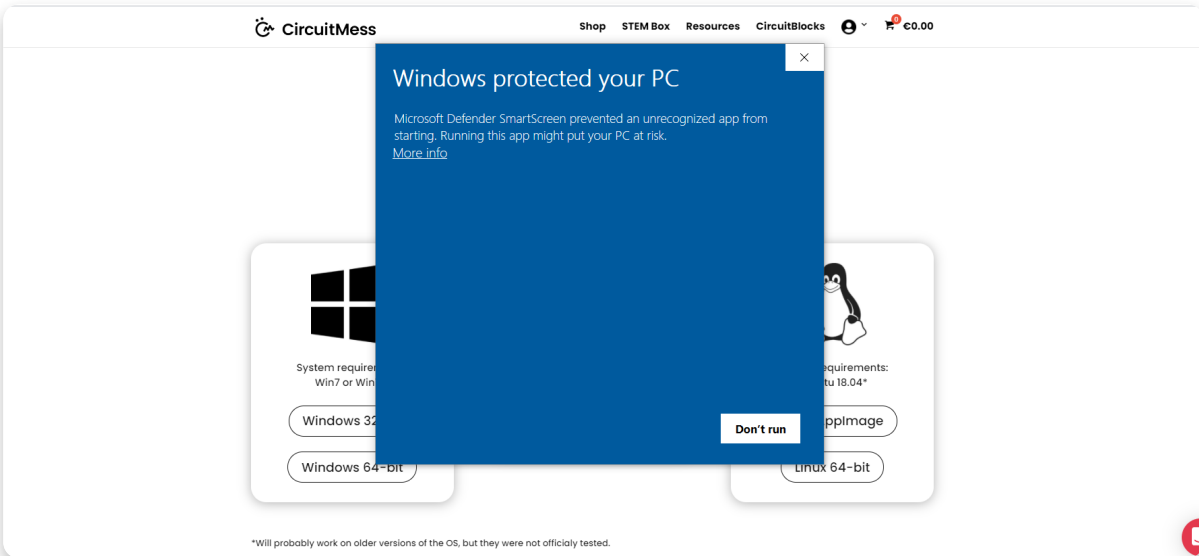
1. Go to the [CircuitBlocks download page](#)
2. **Download the latest version for Windows** – Check if you have a 32 or 64 version. Go to Settings on your PC, click on the System option and find the About section where you'll see the system type.
3. Double-click the downloaded file named "CircuitBlocks"
4. CircuitBlocks will automatically install and create a new desktop shortcut



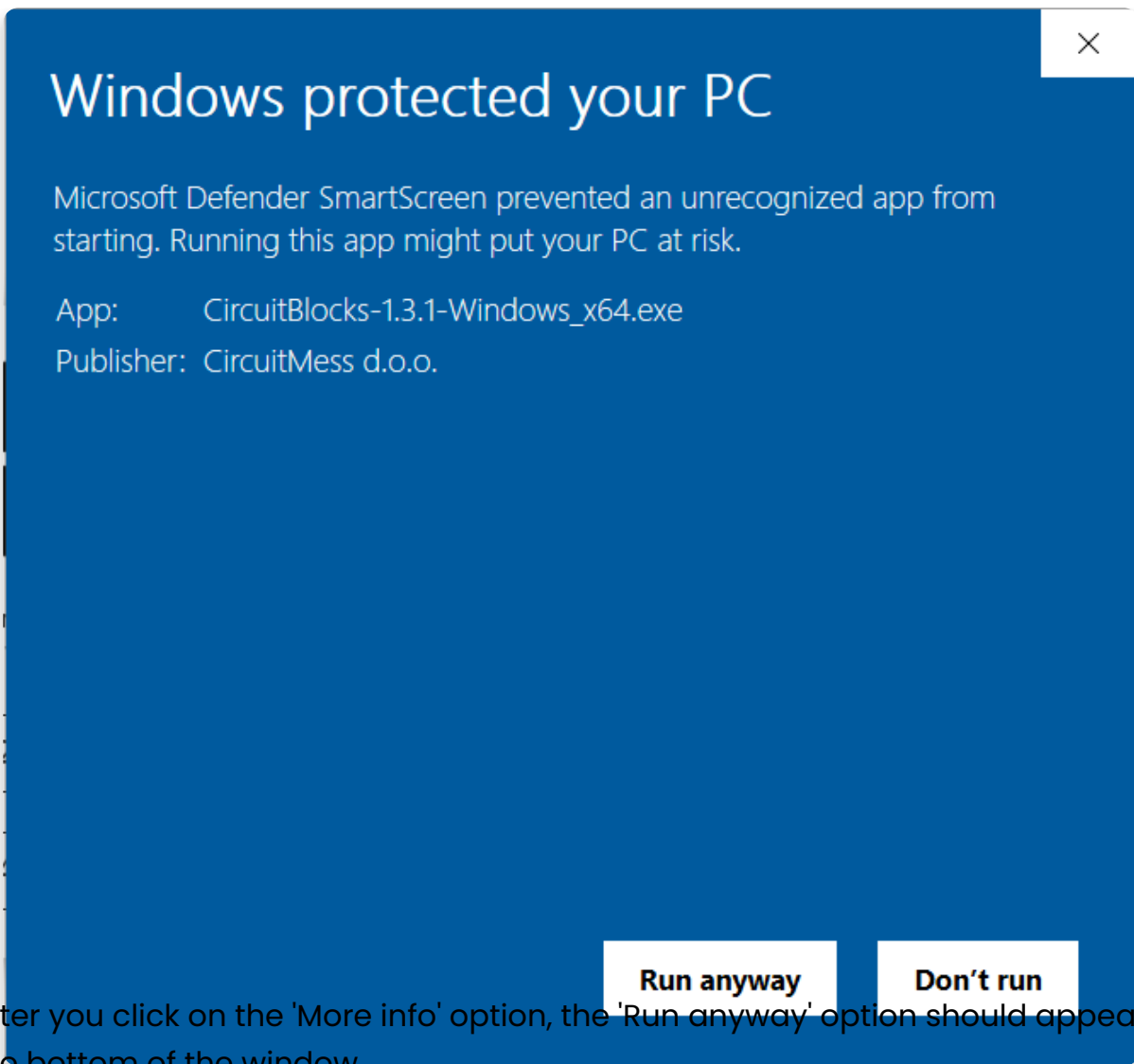
## Your PC is not at risk!



There is a possibility that a notification that says your PC is at risk may pop up when you try to install CircuitBlocks. Don't worry; this happens regardless of CircuitBlocks being safe to run. See the instructions below on how to handle this notification.



This is the message you might get when installing CircuitBlock on your PC. Windows reports a threat despite the program being safe to download and run. Please proceed with the installation by clicking on the 'More info' option.

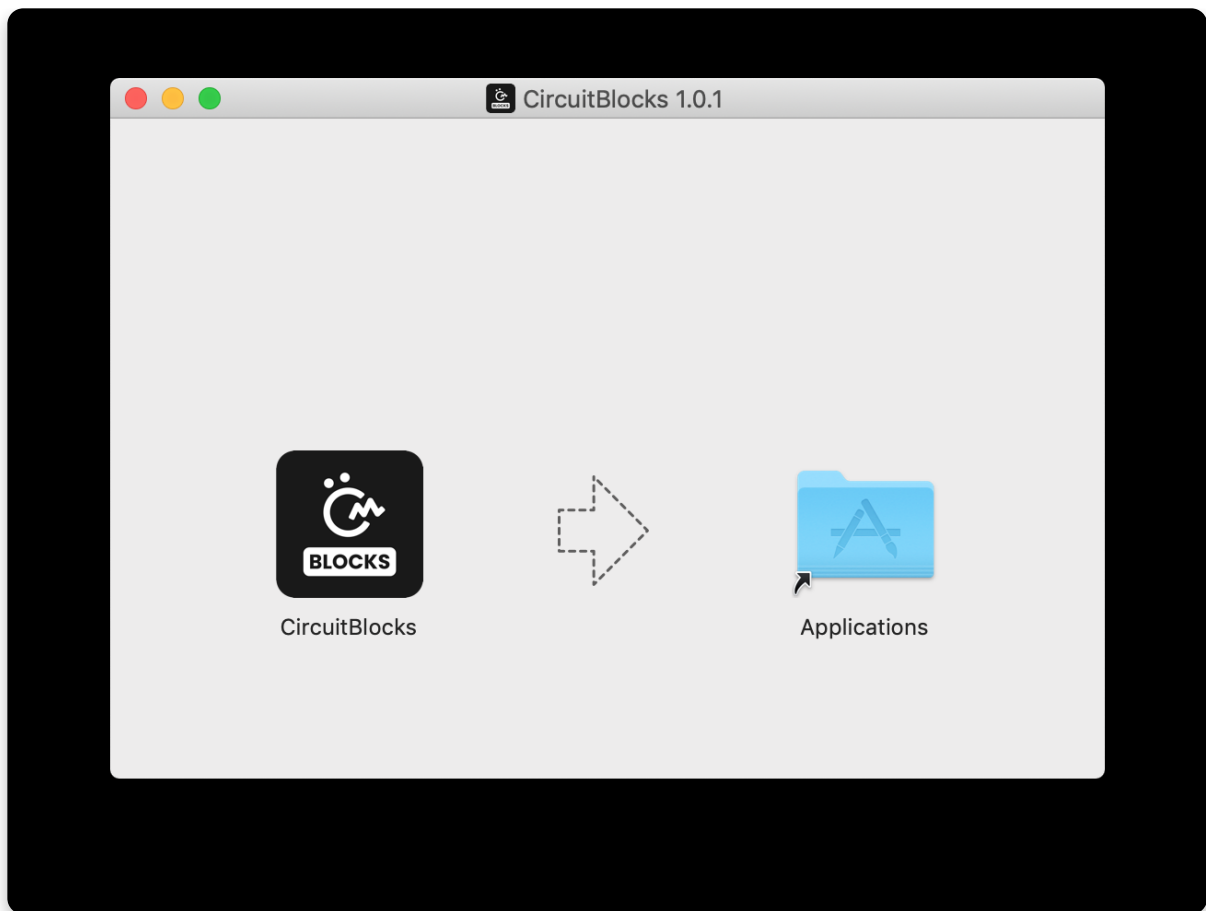


After you click on the 'More info' option, the 'Run anyway' option should appear at the bottom of the window.

Proceed by clicking on 'Run anyway'.

## If you have a Mac computer

1. **Go to the** [CircuitBlocks download page](#)
2. **Download the latest version of CircuitBlocks** for Mac OS (the file named "CircuitBlocks-1.0.1-Mac.dmg" or similarly should be downloaded)
3. Move the files to the 'Applications' folder
4. CircuitBlocks will be installed automatically



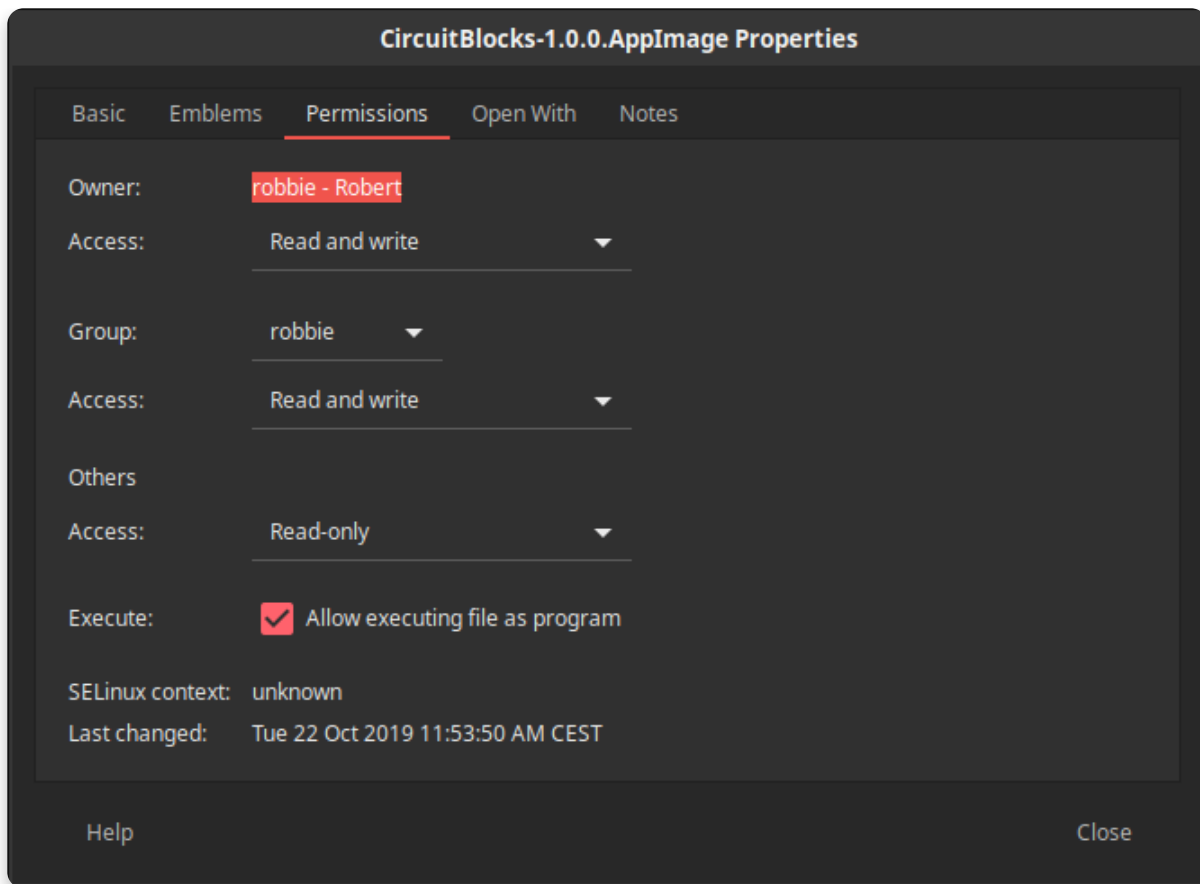
## If you have a Linux computer

There are two ways of installing CircuitBlocks on Linux

1. Go to the [CircuitBlocks download page](#)
2. Press the "Linux 64-bit" download button
3. Double-click the file to run the installation (Ubuntu)  
or  
Open the terminal and write `sudo dpkg -i <path to the downloaded file .deb>` (Other Linux distros)
4. CircuitBlocks will automatically install and create a desktop entry

### Stand-alone (AppImage):

1. Go to the [CircuitBlocks download page](#)
2. Press the "Linux AppImage" download button
3. Right-click on the file and select 'Properties'
4. Go to 'Permissions' and tick 'Allow executing file as program'
5. Double-click the file, and the installation will complete automatically



If you encounter any issues with the installation, please reach out to us via email at [contact@circuitmess.com](mailto:contact@circuitmess.com) and provide a screenshot of your issue and any information you find relevant.

## The basics

### User interface

When you open CircuitBlocks, you will see a window that looks like this.

It's pretty simple – starting a **new project (we also call them "sketches")** can be done by clicking the 'New project' button.

**Saved sketches** will appear right next to that button and you can access them at any time.

If you encounter any kind of an issue with CircuitBlocks, press the '**Send error report**' at the bottom of the main screen. You'll get an error report number – please reach out to us via [contact@circuitmess.com](mailto:contact@circuitmess.com) and provide your error report number.

# Creating a new project (sketch)

Press on the big "New project" button.

You'll get an option to choose the device and sketch type.

For the device, pick **Chatter**.

For the Sketch type, choose **Block**.

**Press the Create button.**

You'll get a screen that looks like this:

On the top of the screen, there is a **toolbar** with a few buttons.

The **block selection bar** is located on the far left – you can take the blocks from there and drop them into the "drawing" area in the middle of the screen.

In the middle of the screen is where you'll be "drawing" your code with colorful blocks.

On the right side of the screen, you will see **code written in C++** appear magically by itself when you drag and drop the colorful blocks.

**C++** is one of the most popular programming languages, but it's fairly complex to understand if you've never coded before.

That's why we've created CircuitBlocks – here, you can drag and drop colorful blocks that represent parts of code and see what your program would look like in C++. When you get skilled enough, you will be able to switch directly to textual coding in C++ without the need for colorful blocks.

## Toolbar

Here's a short explanation of what the buttons in the window toolbar do:

1. **Back to the main menu** – returns you to the home screen without saving
2. **Save/Save As** – saves your sketch, make sure to press this button from time to time, and before closing CircuitBlocks
3. **Chatter connection indicator** – the red dot turns green if your Chatter is connected to your computer via a USB cable

4. **Sprite editor** – for drawing characters you want to have on your Chatter.
5. **Export to binary** – saves a binary file of your code to your computer. This is a more advanced function that you won't need for now
6. **Serial monitor** – this button opens a window that we call the "Serial monitor". "Serial" is a nickname for a type of communication that is happening between Chatter and your computer. In this window, you will later be able to see the messages sent from Chatter to your computer via the USB port.
7. **Close code** – with this button, you can close or re-open the code window on the right of the screen. This is useful if you need more screen space for seeing your colored blocks.
8. **Run** – This button will translate the code you have constructed in CircuitBlocks to *machine code* that Chatter understands (beep boop beep boop 1011100101) and send the code to your Chatter via the USB port

## Code window

The so-called "Code window" has the following parts:

1. **Main code screen** – code written in C++ will appear here as you drag and drop colorful blocks on the left side of the screen.  
You'll see that some parts of the code are colored in funny colors. Programmers call this *syntax highlighting*. Basically, what is happening is that different categories of code commands are colored differently so that programmers can understand the code more easily.
2. **Light/dark theme switch** – you can toggle the background and text color of the code window with this button.
3. **Expand** – stretches the code window across the entire screen. Press it again to resize it to the half-screen again.
4. **Close** – closes the code window, the same functionality as the toolbar's 'Close Code' button.

## Drawing board

The drawing board is where the magic happens.

It has the following parts:

1. **Search bar** – type the component's name you are looking for here.
2. **Component selector** – the blocks are divided into different categories here. Each category has a specific color designated to it.
3. **Drawing area** – you will drag the blocks from the component selector and drop them into the drawing area. This is how code is made. Easy peasy!
4. **Center tool** – if you get lost when scrolling across the drawing area, press this button, and it will center your view on the blocks you have dropped on the drawing area.
5. **Zoom buttons** – to zoom in and out of the drawing area.

## Types of blocks

There are a total of **nine** block types in CB. Each of them is represented by their color. Every block translates to code, which is then compiled and uploaded to the phone, just like on every Arduino based platform.

Pressing on every block type will open a section from which you can drag and drop those blocks into the drawing area.

Also, pressing on '**More**' will open even more blocks that are not so commonly used.

There are two main functions of every Arduino code – **void setup()** and **void loop()**.

Everything that goes into the **void setup()** the function will run only once. It is primarily used to start the software, initialize and declare variables, and run functions that only have to run once (ex., Intro screen in a video game).

The **void loop()** is where everything else takes place. It basically runs every bit of code inside it repeatedly (speed depends on the device – just imagine it's ultra-fast!). It should pretty much follow the screen's refresh rate and make the program do things accordingly.

Every block you place automatically goes into the **void loop()** function.

If you wish to put something in the **void setup()**, you have to drag the main block from **Functions** and place your blocks inside as you wish, but more on that a little bit later.

## Elliptical blocks



**Elliptical blocks** represent variables. Whether we're talking about integers, strings, or other variable types (other than Boolean), they can all be recognized by the same shape.

Also, larger blocks with elliptical shapes return either integer or float values.

Whenever you find circular "holes" inside some blocks, you can insert variables. It's most commonly found in **comparison** or **action** blocks.

## Triangular blocks

Triangular blocks represent boolean variables.

Both variables (true and false) and functions that return boolean values have the same shape.

Regardless of color, each of these blocks returns either true or false.

Triangular "holes" require boolean blocks to be inserted.

## Building blocks

Everything else is basically a building block. Those are functions that have no return value (they return *null*). Both elliptical and triangular blocks must first be placed inside the building blocks to act as part of the program.

They have a specific "puzzle" shape and can be stacked inside each other.

The main **building block** is located inside the **'Functions'** section.

It basically gives you two main building blocks sections.

Everything placed inside Arduino runs **first** goes into **void setup()**, and everything placed inside **Arduino loop forever** goes into a **void loop()**.

## Inserting blocks

Now, this is the main part.

The whole point of blocks-like IDE is connecting blocks and placing them inside another.

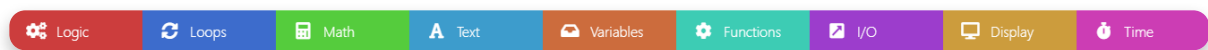
It is all done by simple **drag-and-drop** action.

Here is an example of a program that will set the variable **Var** to **1** and then **increase that variable while it is smaller than 10**.

At the end of the program, **Var will be 10**.

This is just a simple example, and block-building will be further explained in the following chapters.

## Block sections



There are a total of nine sections in CircuitBlocks. We've organized them so that you'll be able to find everything in a maximum of two clicks.

The sections themselves are pretty self-explanatory, but we'll go through them all to get a little better understanding of the whole concept.

Some of the sections also have additional blocks (in the **'More'** menu) where you'll be able to find some of the functions that are not used that often but can still be useful.

## Logic

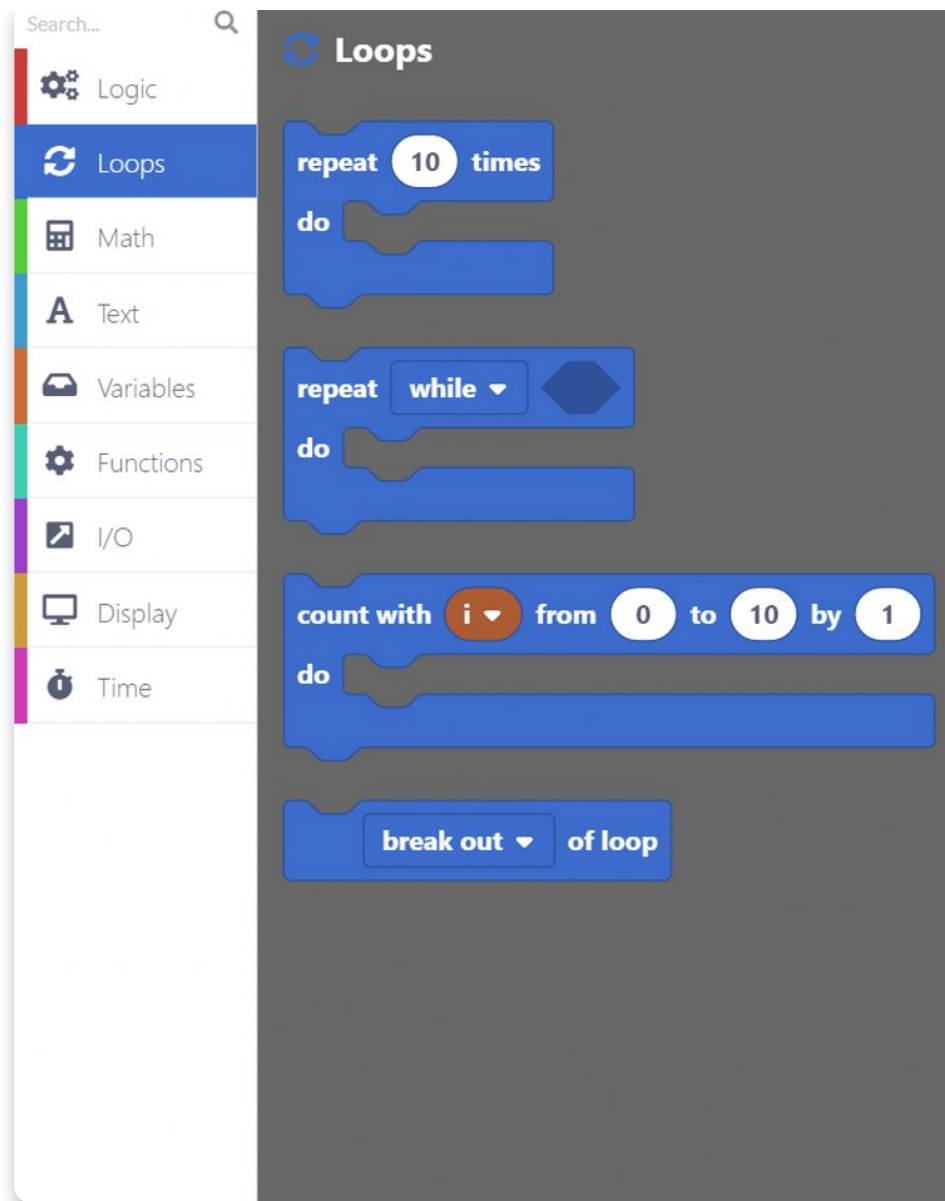
This is where the base of every code is located.

Every **if, if-else, else** function, comparisons, **and/or, not, true/false** and other logical operators.

## Loops

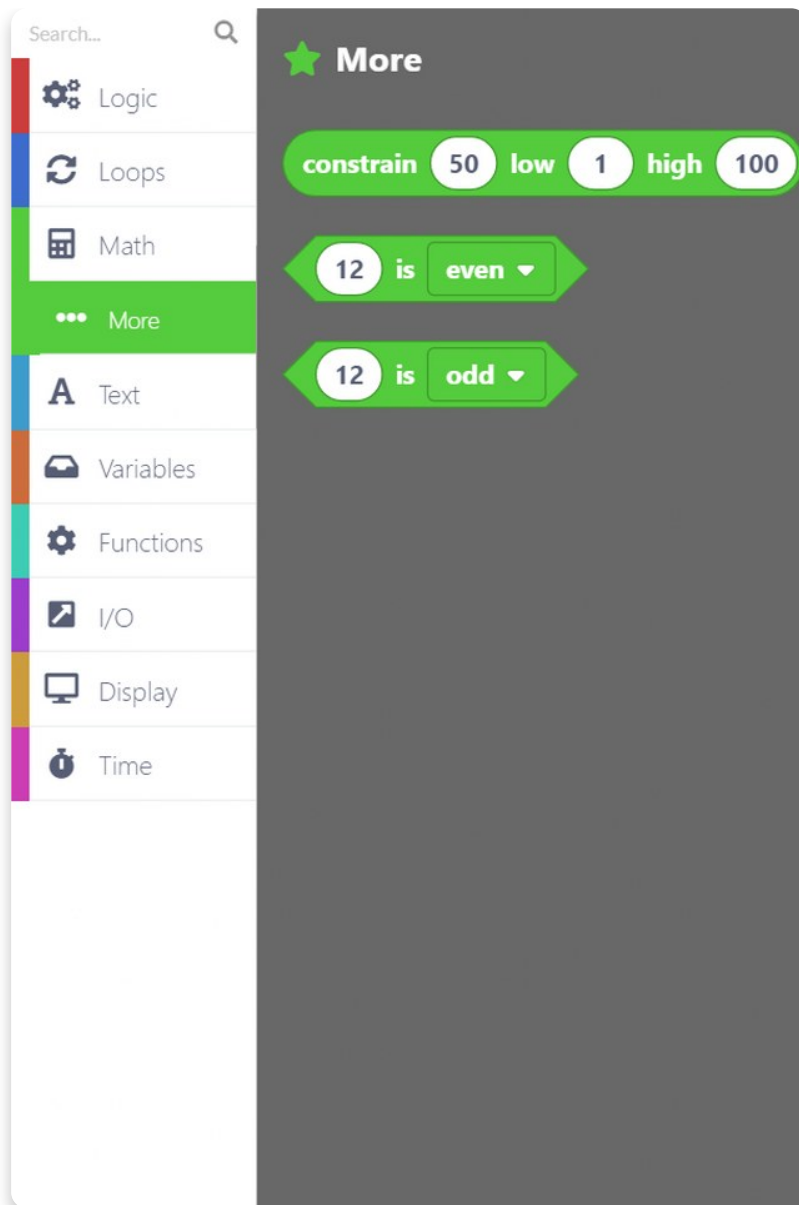
Loops are functions that repeat everything inside for a specific amount of time.

They can have conditional and repeat for as long as that condition is met or have a pre-determined number of repeats.



## Math

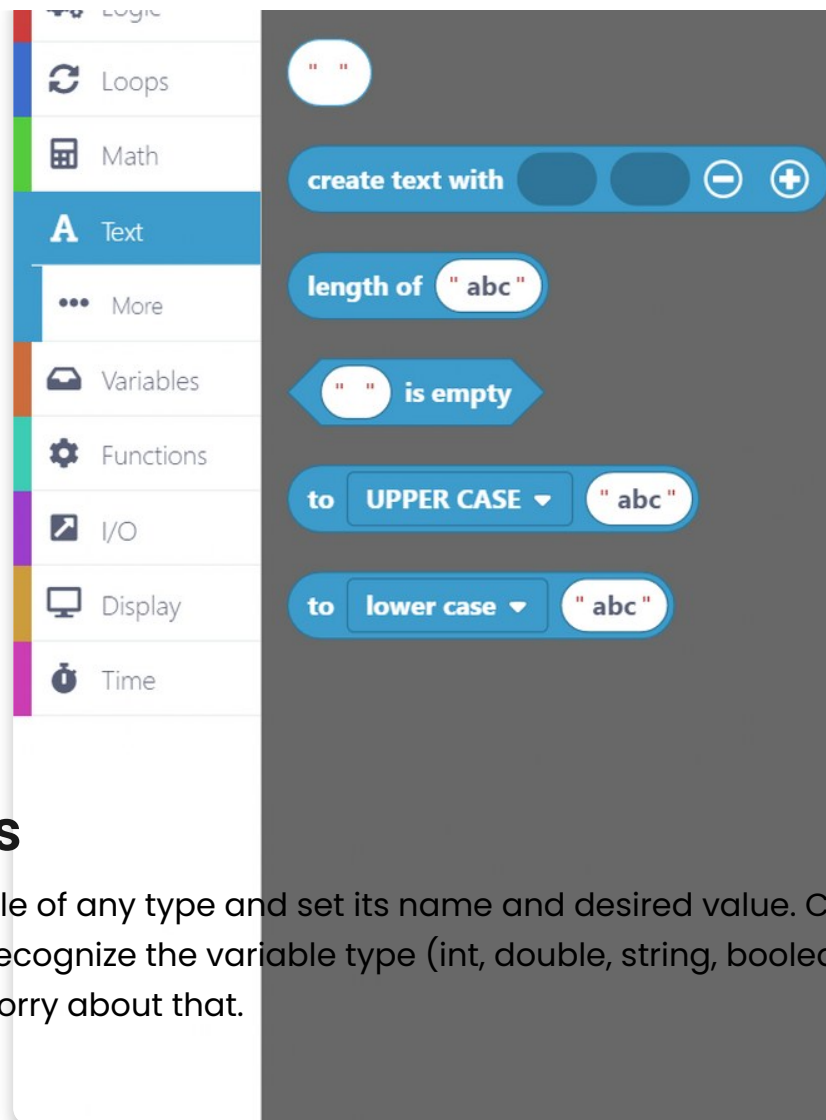
Pretty much every math function is located here. From basic operations to rounding numbers and working with angles, you will easily trigger your inner Einstein or Pythagoras in a matter of seconds!



## Text

Strings, characters, and string manipulation. Great place for creating new text and implementing it in your programs.





## Variables

Create a variable of any type and set its name and desired value. CB will automatically recognize the variable type (int, double, string, boolean), so you don't need to worry about that.



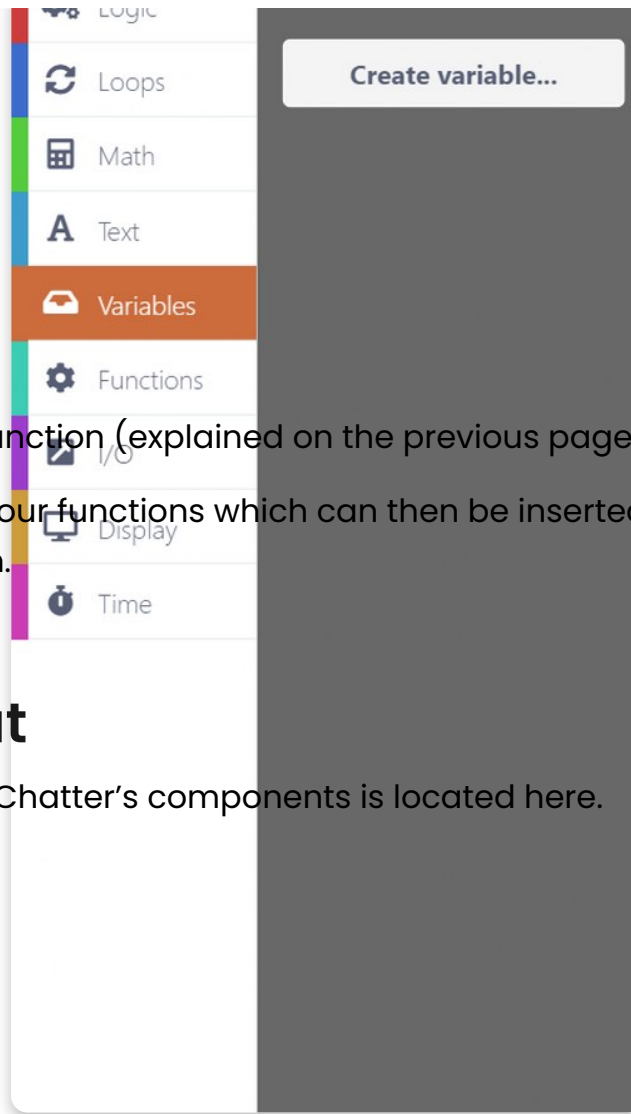
## Functions

The Default Arduino function (explained on the previous page) is located here.

You can also create your functions which can then be inserted as one of the main parts of your program.

## Input/Output

Everything regarding Chatter's components is located here.



## Display

Well, all these blocks are really not important if you don't see anything on the screen!

Here is where all the magic translates to those colored pixels. You can create so much through these blocks.

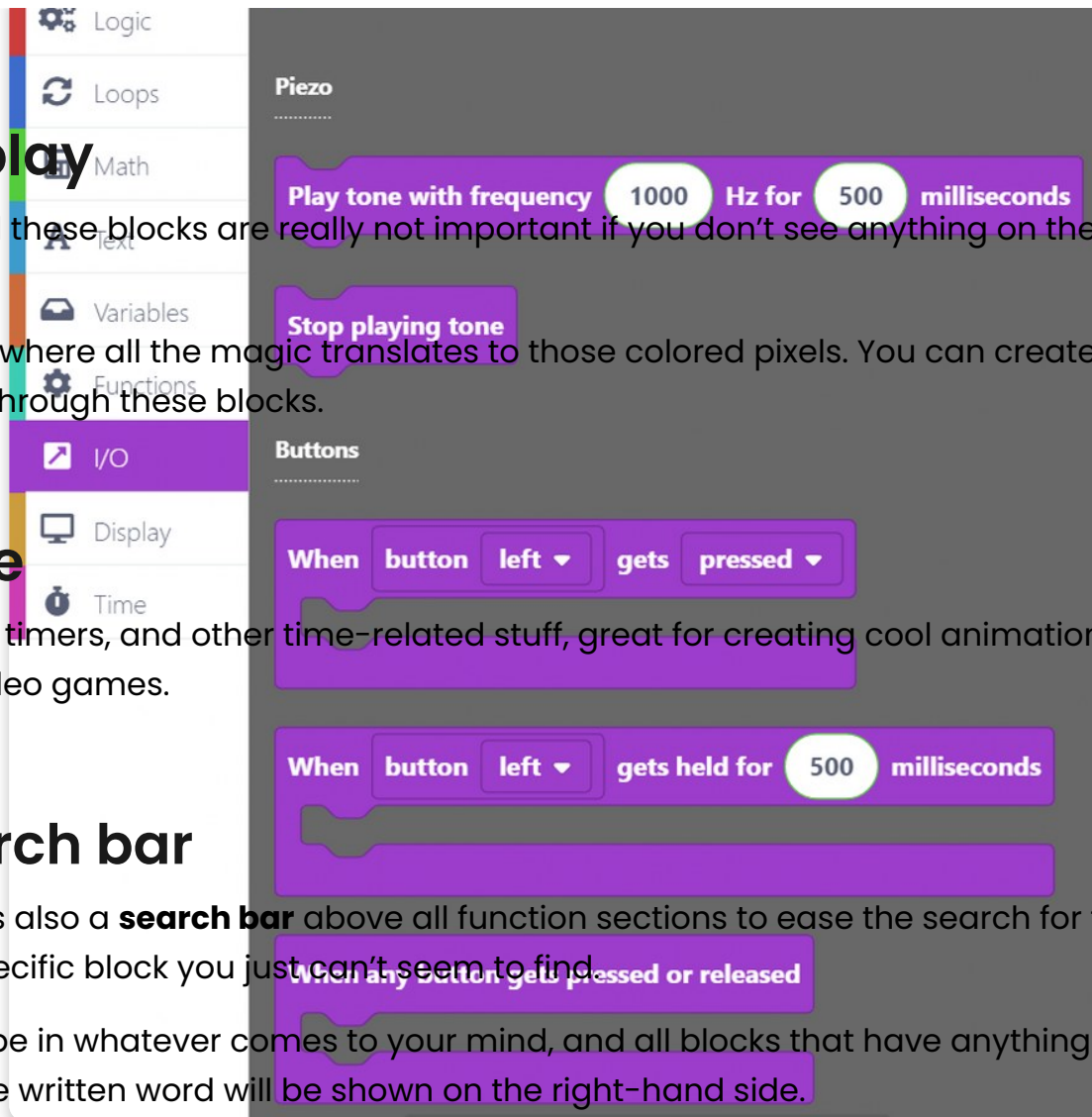
## Time

Delays, timers, and other time-related stuff, great for creating cool animations and video games.

## Search bar

There is also a **search bar** above all function sections to ease the search for that one specific block you just can't seem to find.

Just type in whatever comes to your mind, and all blocks that have anything to do with the written word will be shown on the right-hand side.



Now, you really can't say that it's impossible to find something.

You've learned everything about the blocks!

It's time to move on to the next lesson...

Let's start! Step by step

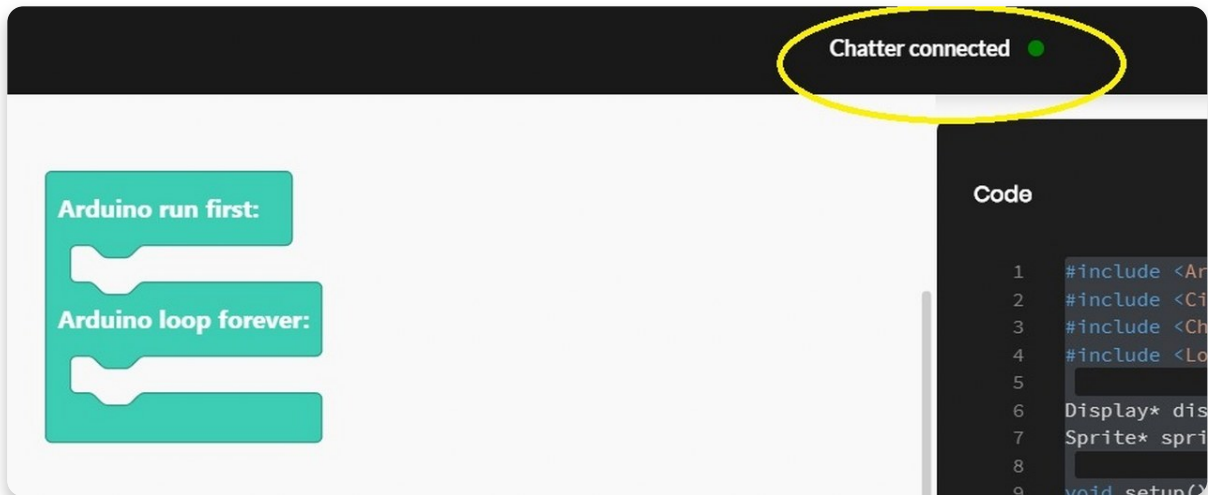
## Let's write on the display!

Let's get down to business!

Before doing anything, you need to connect your Chatter to your computer's USB port and turn it on.



If everything is okay, CircuitBlocks should say "**Chatter connected**".



If CircuitBlocks didn't recognize your Chatter, please check if the USB cable is plugged in properly and if you are using a working USB port on your computer.

If you still cannot get CircuitBlocks to recognize your Chatter, something possibly went wrong with the driver installation on your computer. Drivers are these little programs that help your computer communicate with Chatter, and they sometimes act funny. Reach out to us via email at [contact@circuitmess.com](mailto:contact@circuitmess.com) if you cannot get your computer to recognize your Chatter.

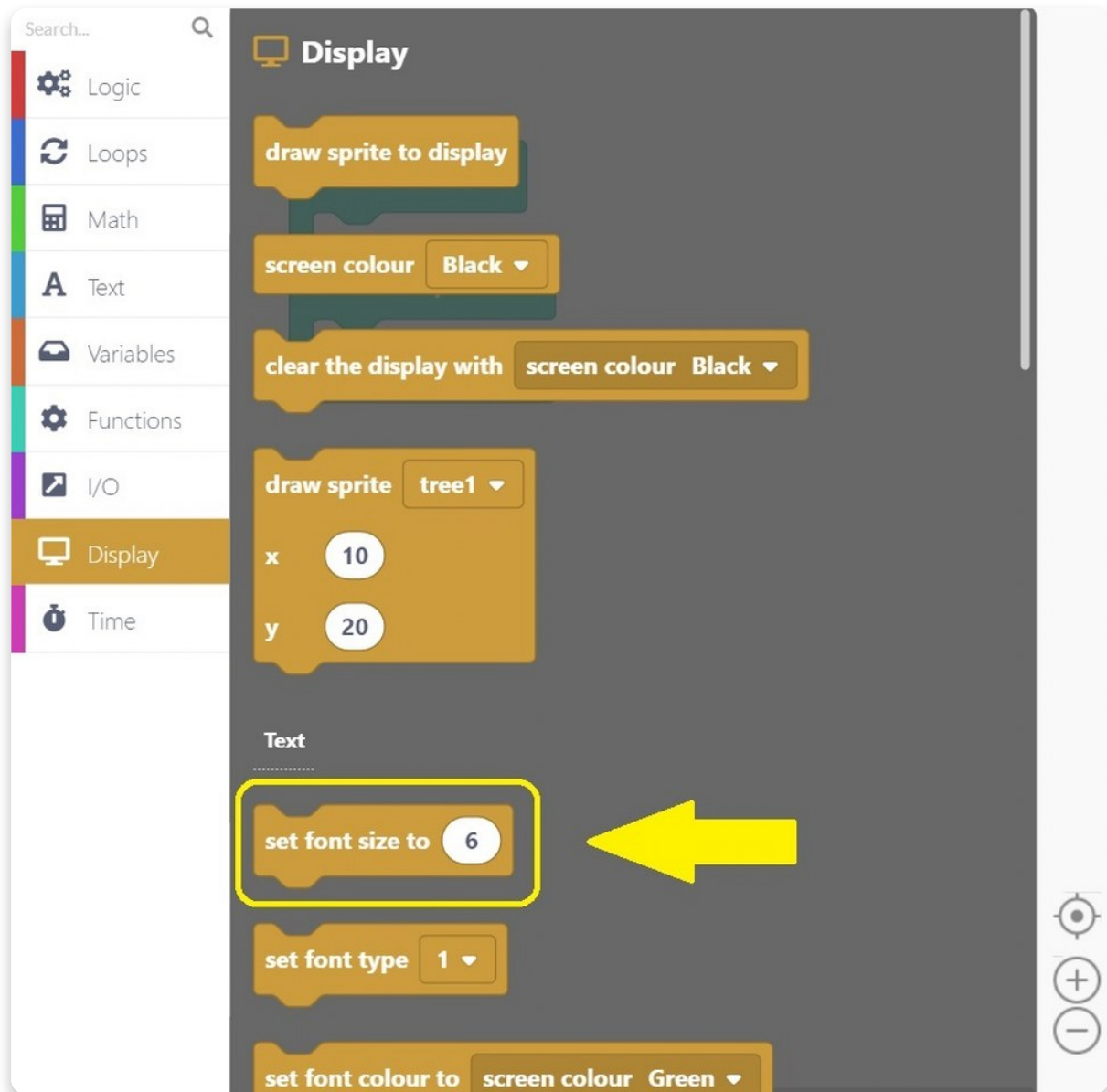
### **Let's write something!**

We will kick things off as simply as possible.

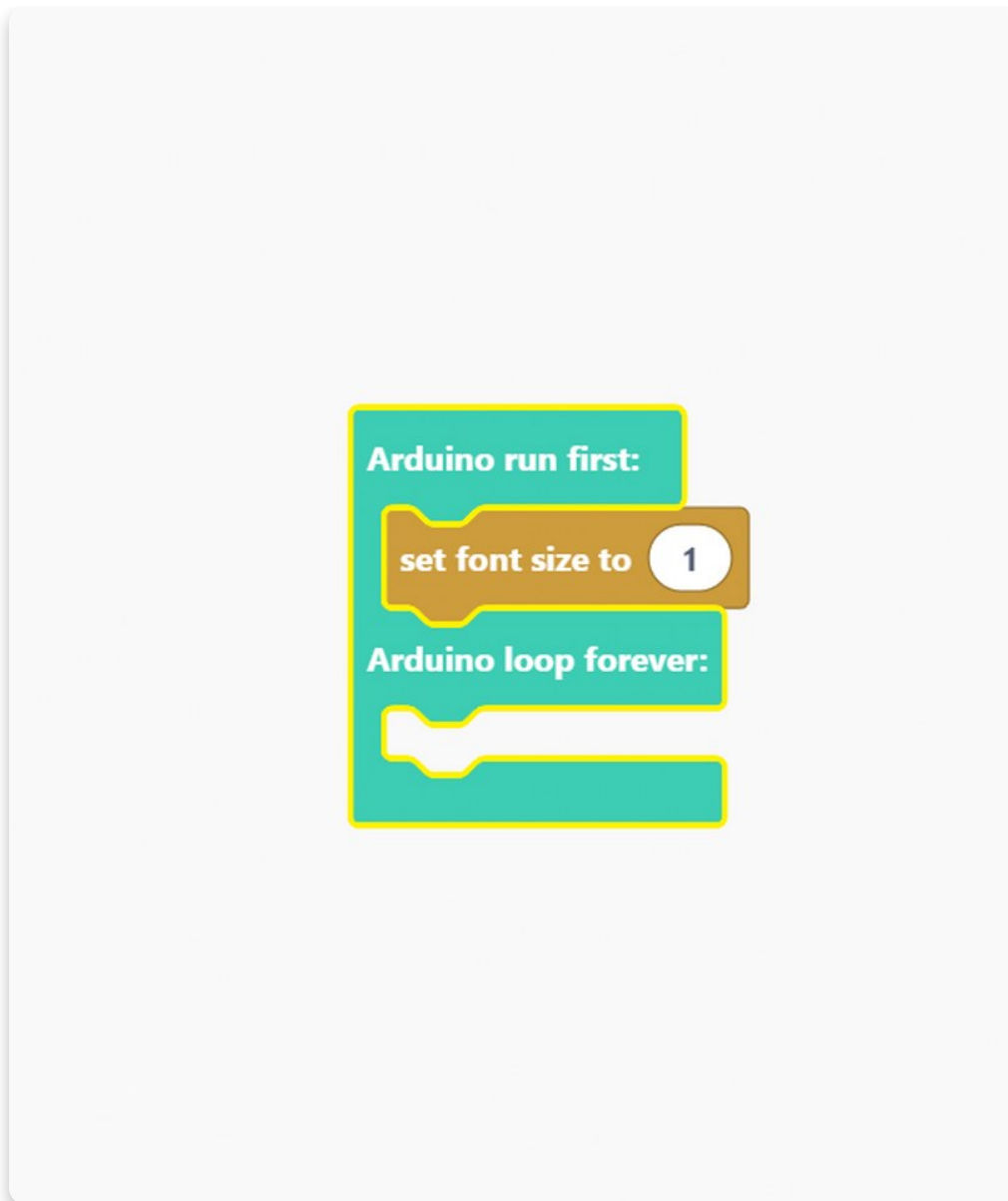


The first thing we're going to learn is how to **clear Chatter's display** and **write** on it!  
You'll only need to use a **Display block section** to do that.

Please click on the mentioned section, and choose a "**set font size to 6**".



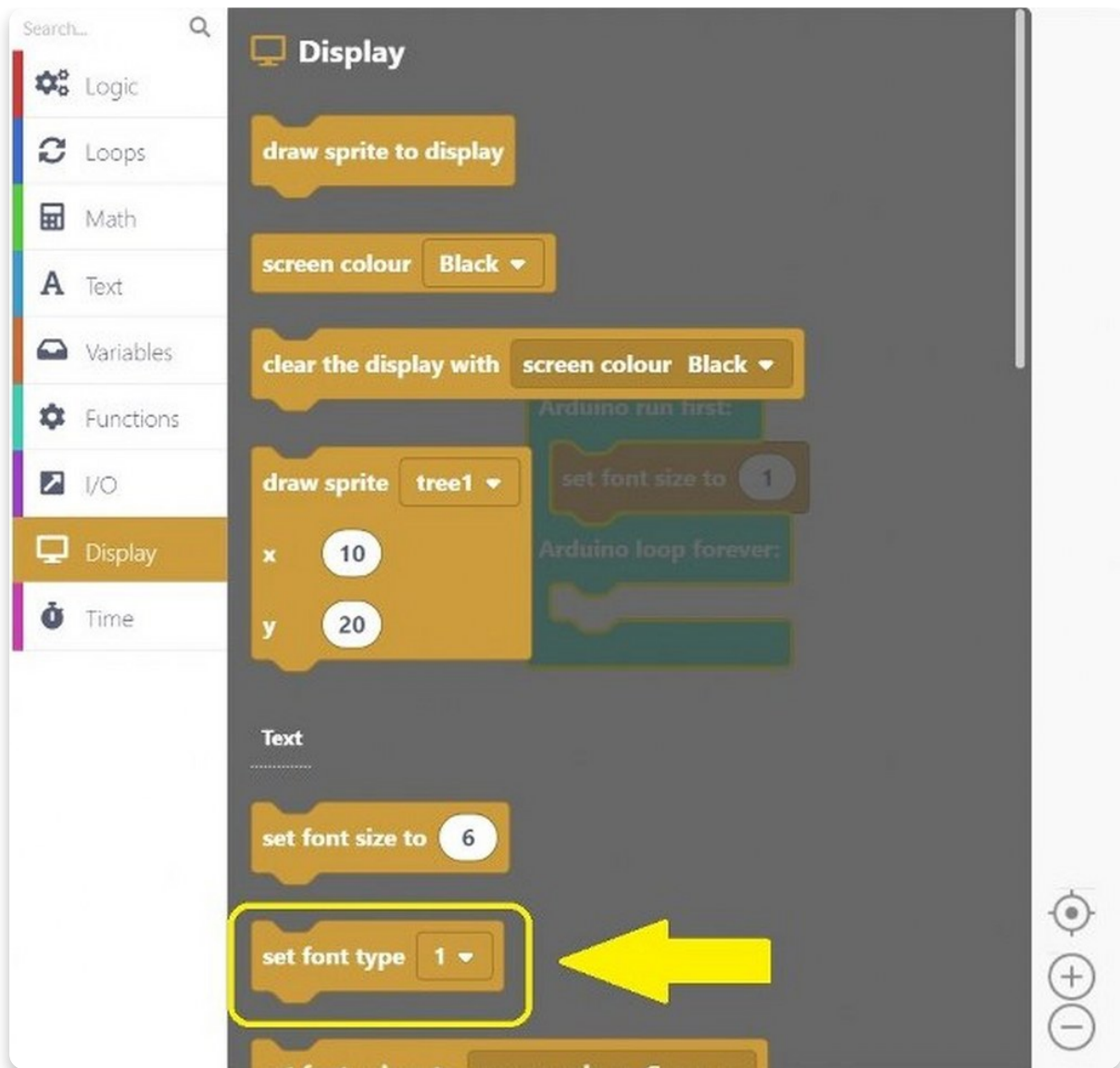
Once you click on the block, it will appear on the **drawing area**, and you'll need to drag it into a bluish "**Arduino run first**" block.



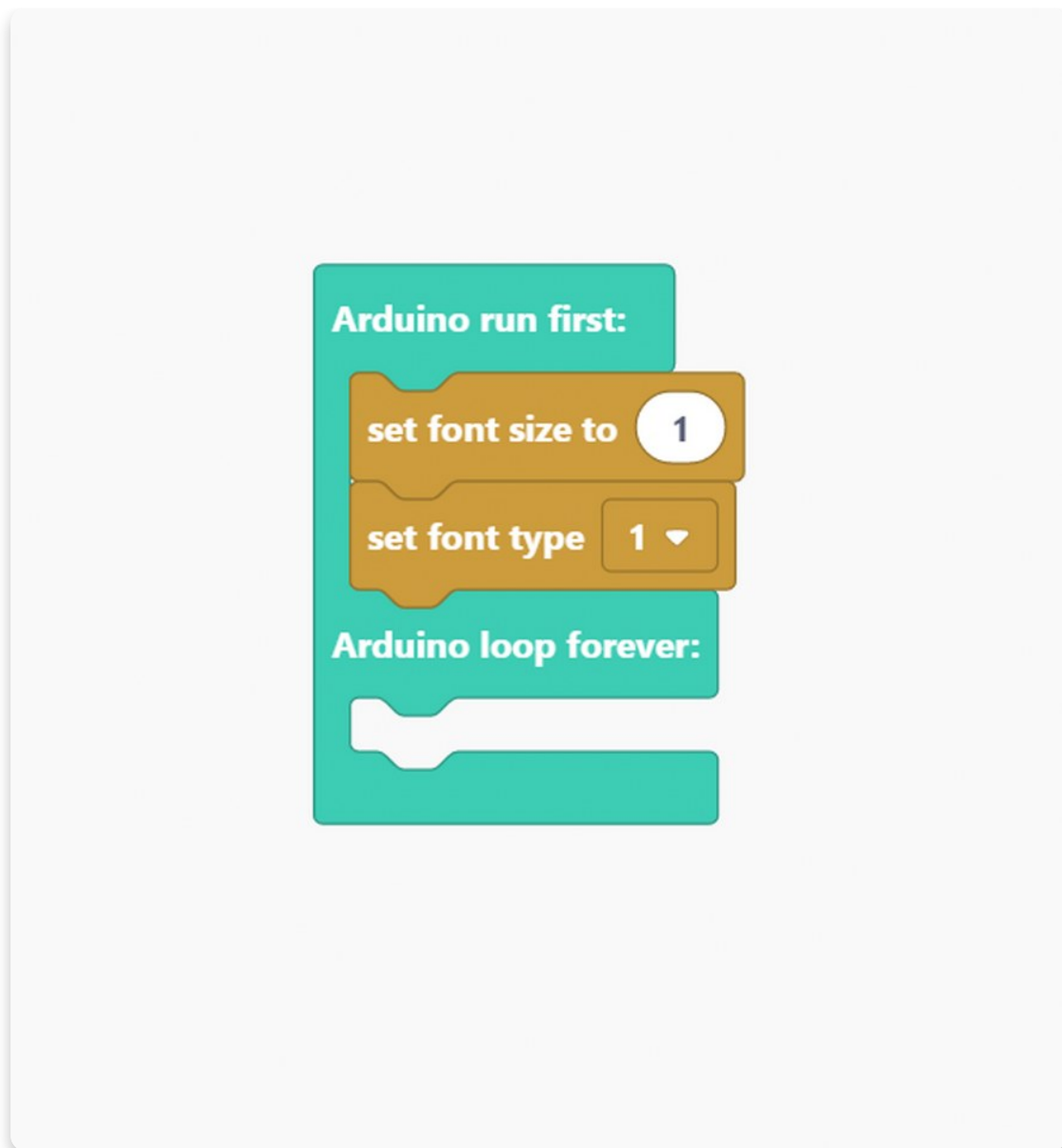
We think the font size 6 might be too big, so we changed it to 1. You can do that by simply deleting 6 and writing 1 instead.

Easy, right?

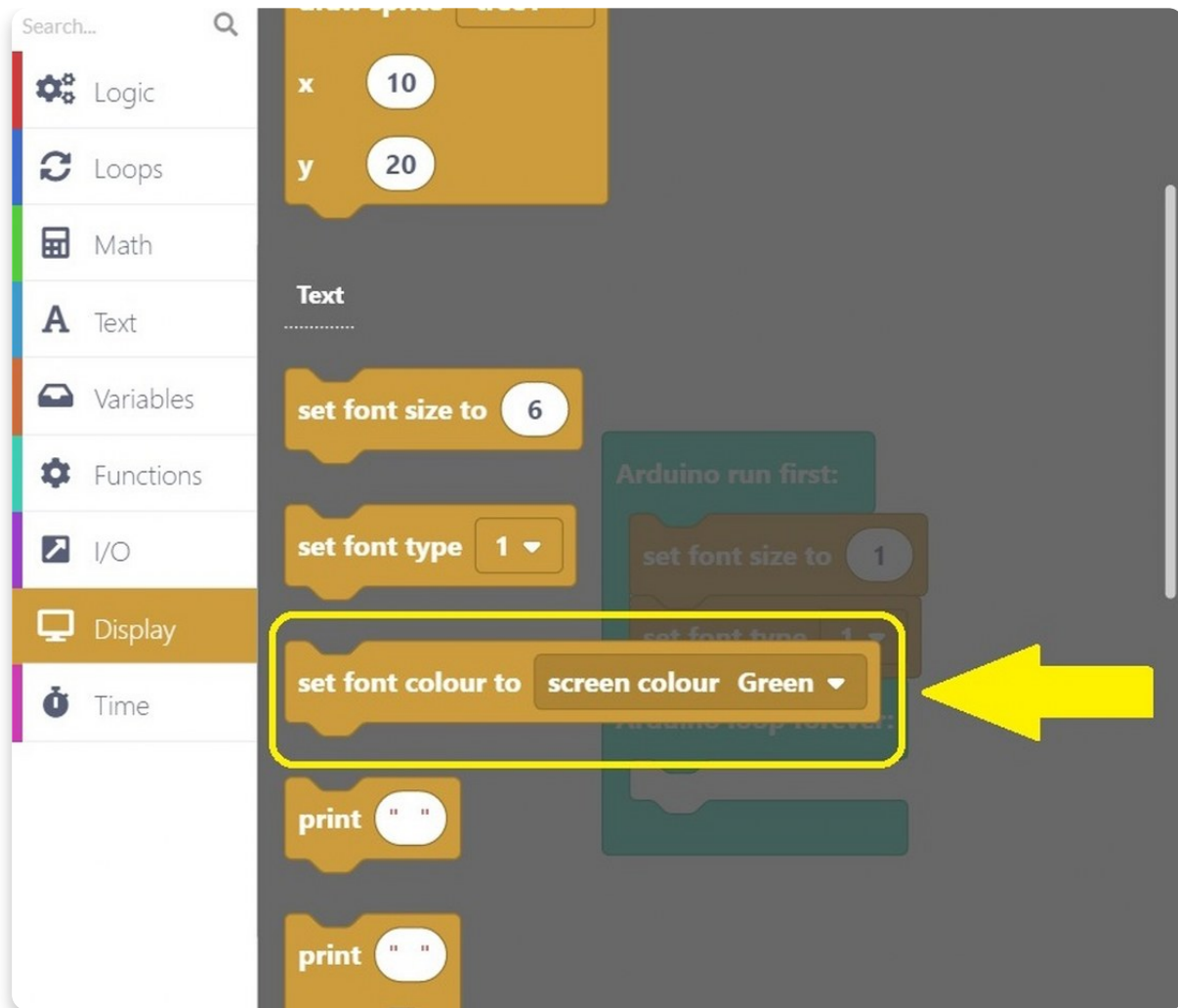
The next thing we'll have to set is a **font type**. You can find that block in the **Display section** also.



Font type doesn't need any change, but if you want to experiment, feel free to!



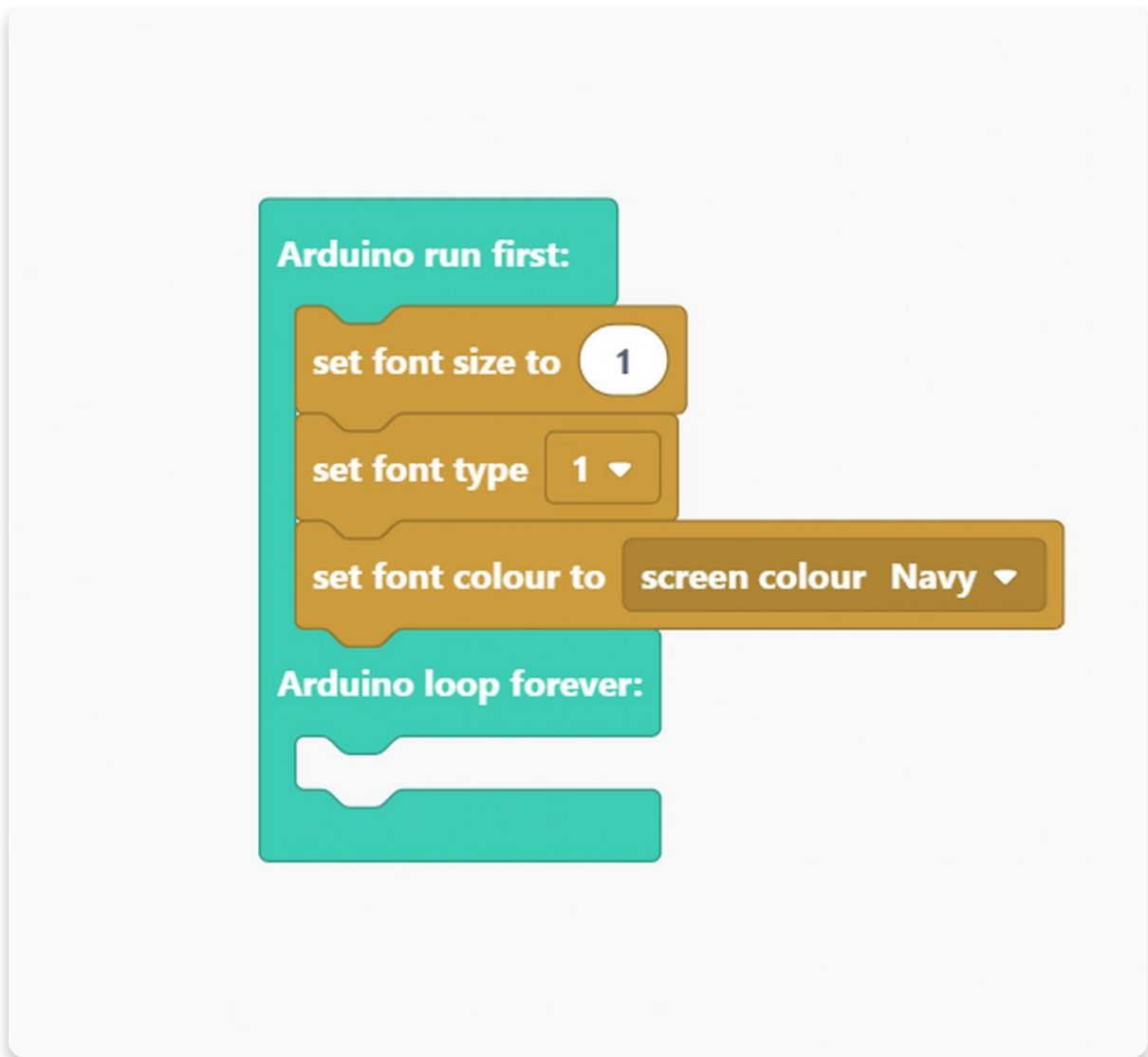
The third thing we'll use from the **Display section** is a "**set font colour to screen colour**" block.



Instead of green, we choose the font color to be navy.

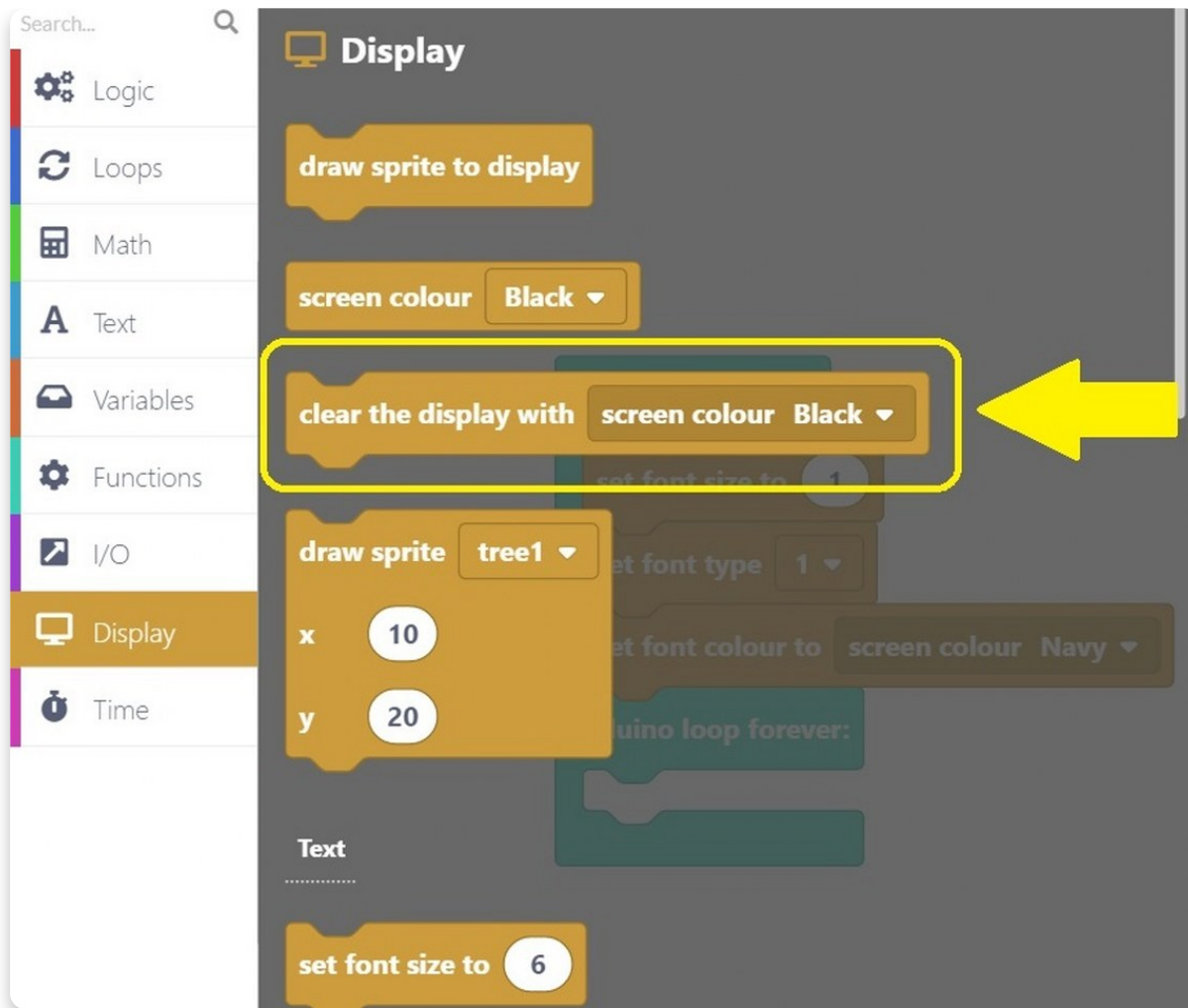
Note that you can choose any color you prefer.

Your sketch should look like this now:



Now that you set all the necessary sizes, colors, and font types, it's time to **clear the display** with some color so we can clearly see the sentence (or word) you'll print.

Like in the previous block, you can choose any color you want.



Click on the circled block, and drag it in the **Arduino run first** like the rest of the blocks.

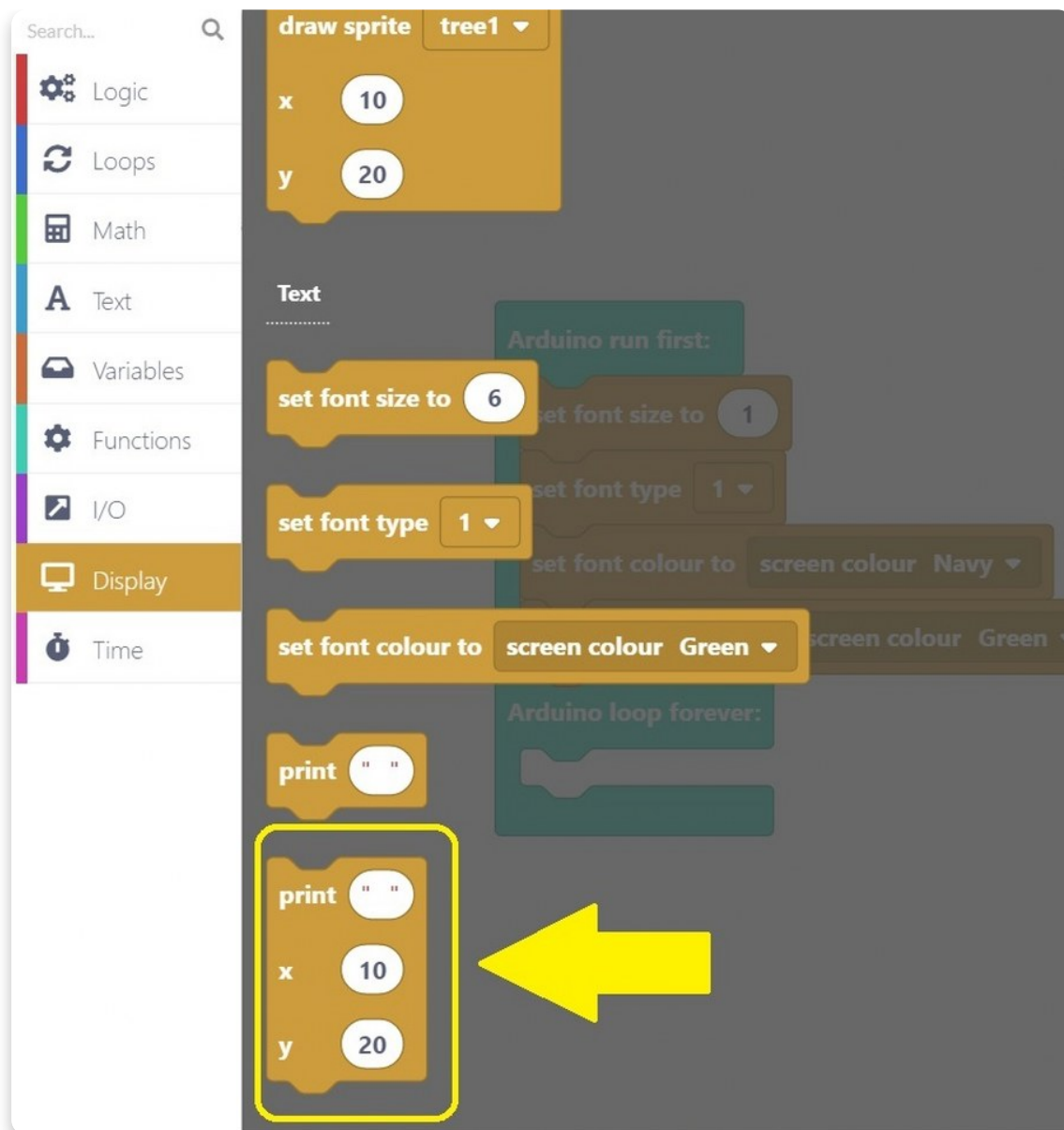
As you can see, we decided the display to be green.



Let's make the main part of this sketch - write a sentence (or a word) you want to print on display.

To do that, you'll have to use this block:



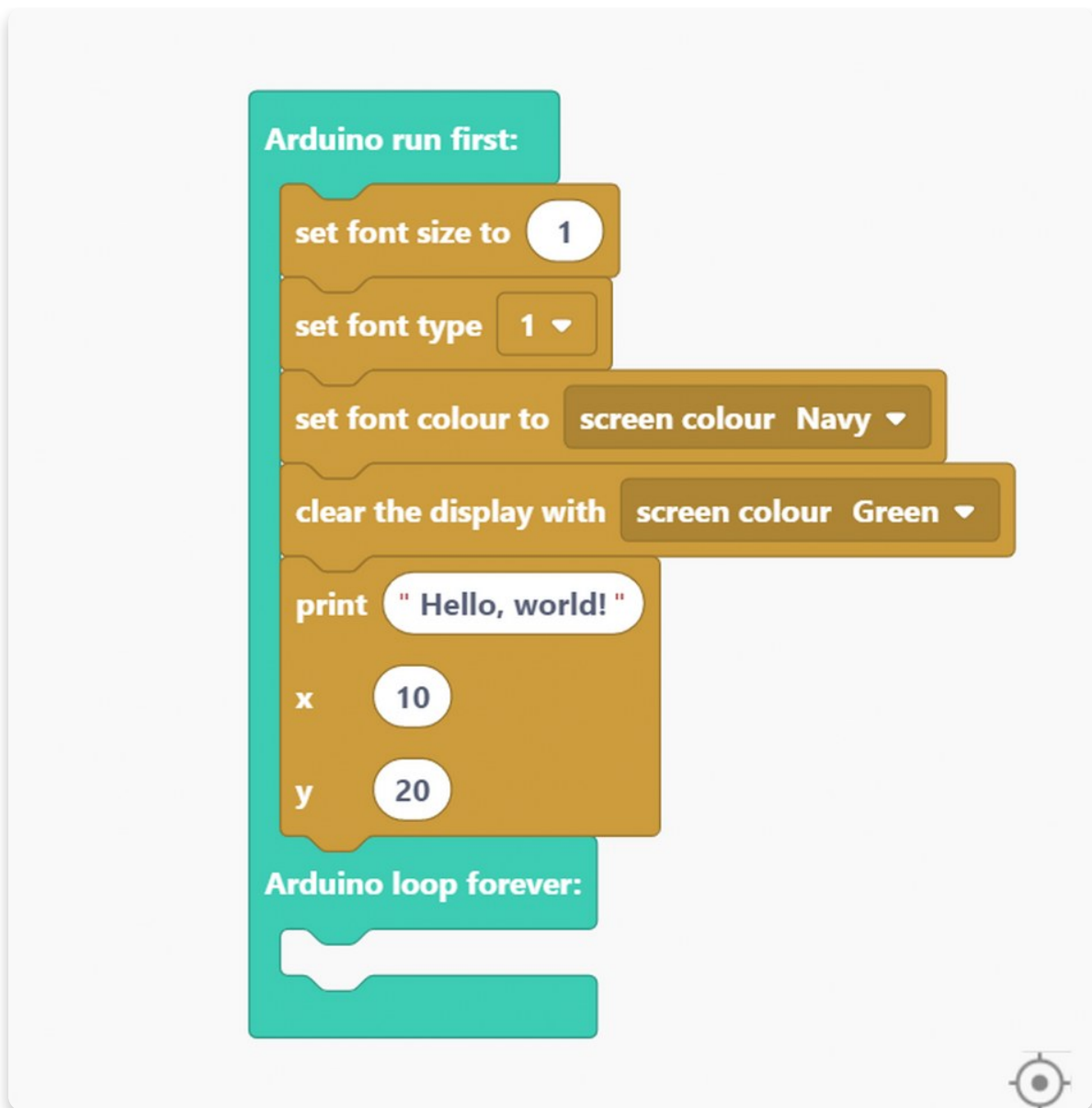


As you can see, there are three white circles used for writing.

The first one, next to print, determines the sentence that will appear on display.

The **x** and **y** mark **coordinates** where the text will appear on display.

We decided to write "**Hello, world!**" on the screen, but we kept the coordinates.



The last thing you need to do in this sketch is clicking on a **"draw sprite to display"** block.

We need to use this block to ensure this code will appear on display.

Search...

- Logic
- Loops
- Math
- Text
- Variables
- Functions
- I/O
- Display**
- Time

### Display

**draw sprite to display** ←

screen colour **Black** ▾

clear the display with **screen colour Black** ▾

draw sprite **tree1** ▾

x **10**

y **20**

Text  
.....

set font size to **6**

to run first:

set font size to **1**

font colour to **screen colour Navy** ▾

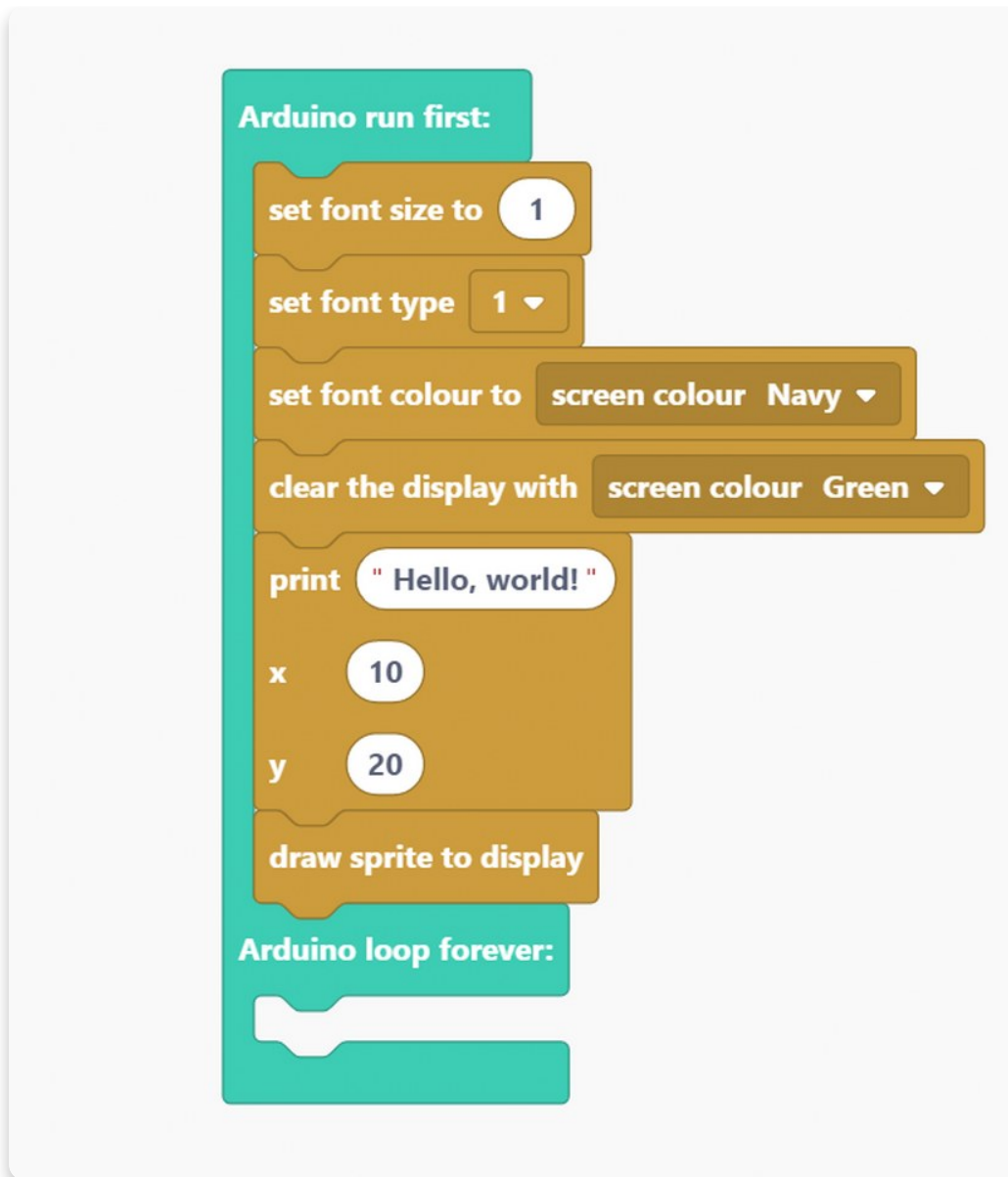
clear the display with **screen colour Green** ▾

print **"Hello, world!"**

x **10**

y **20**

Arduino loop forever:

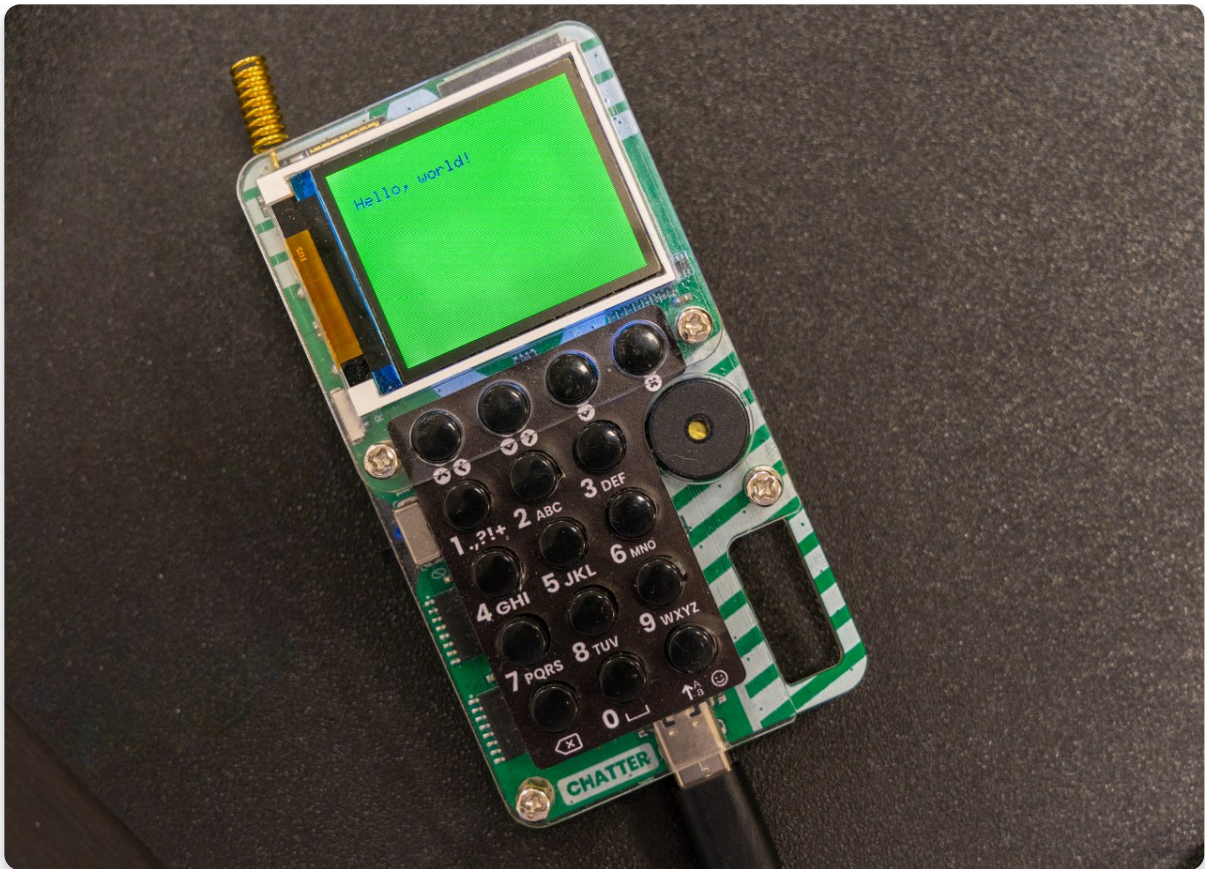


Hit the big red **Run** button and wait for the code to **compile!**

If you're doing this for the first time, the code can take up to a minute to compile. But, don't worry, after that, the compiling should be faster.

When you hit the Run button, the red line will appear under the Toolbar representing the percentage of code that's compiled. Once the code is compiled, your Chatter will restart, the display will turn green, and there'll be a text in navy saying, "Hello, world!"

You can see what will happen in the photos below:

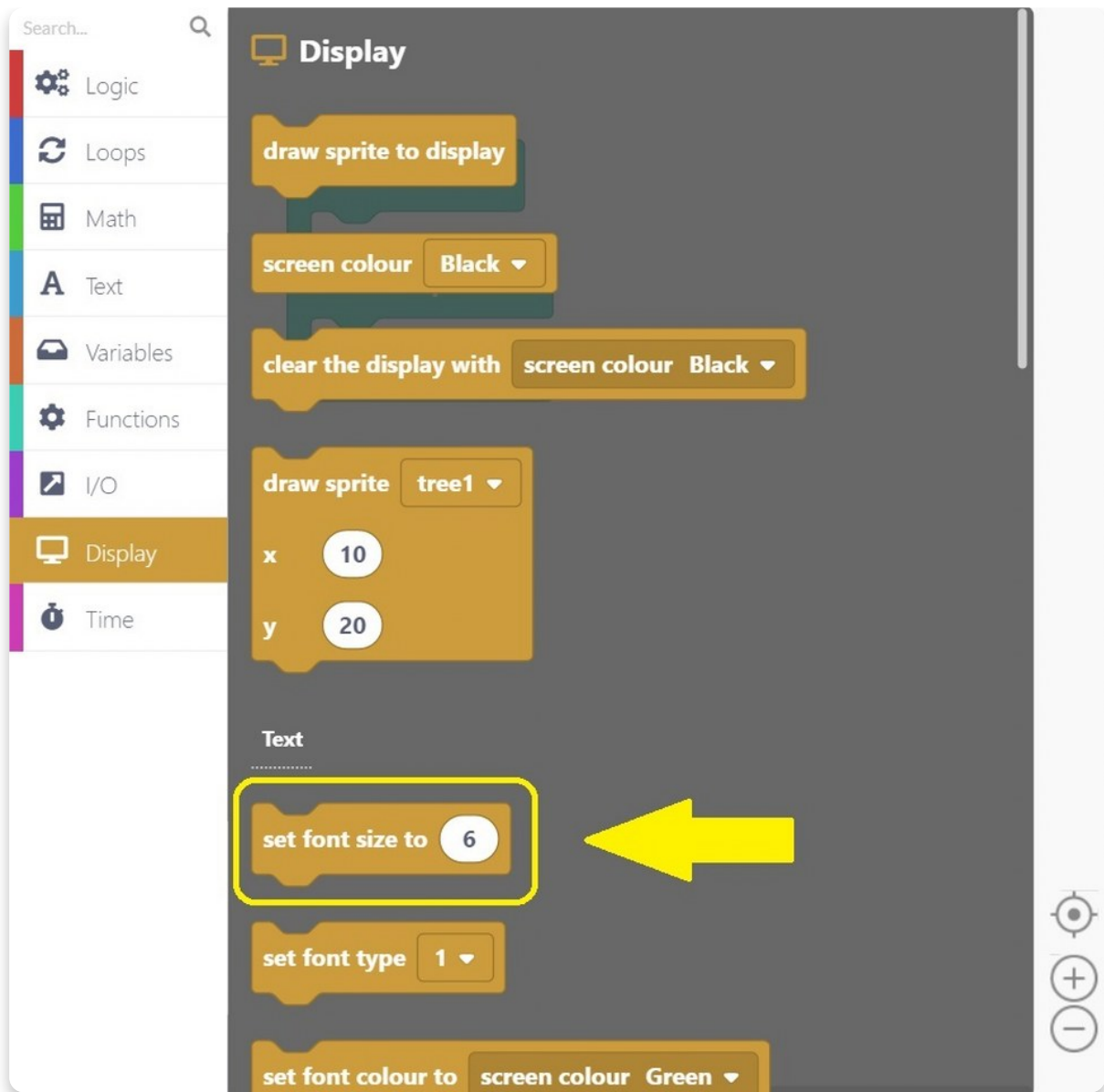


## Click, click...

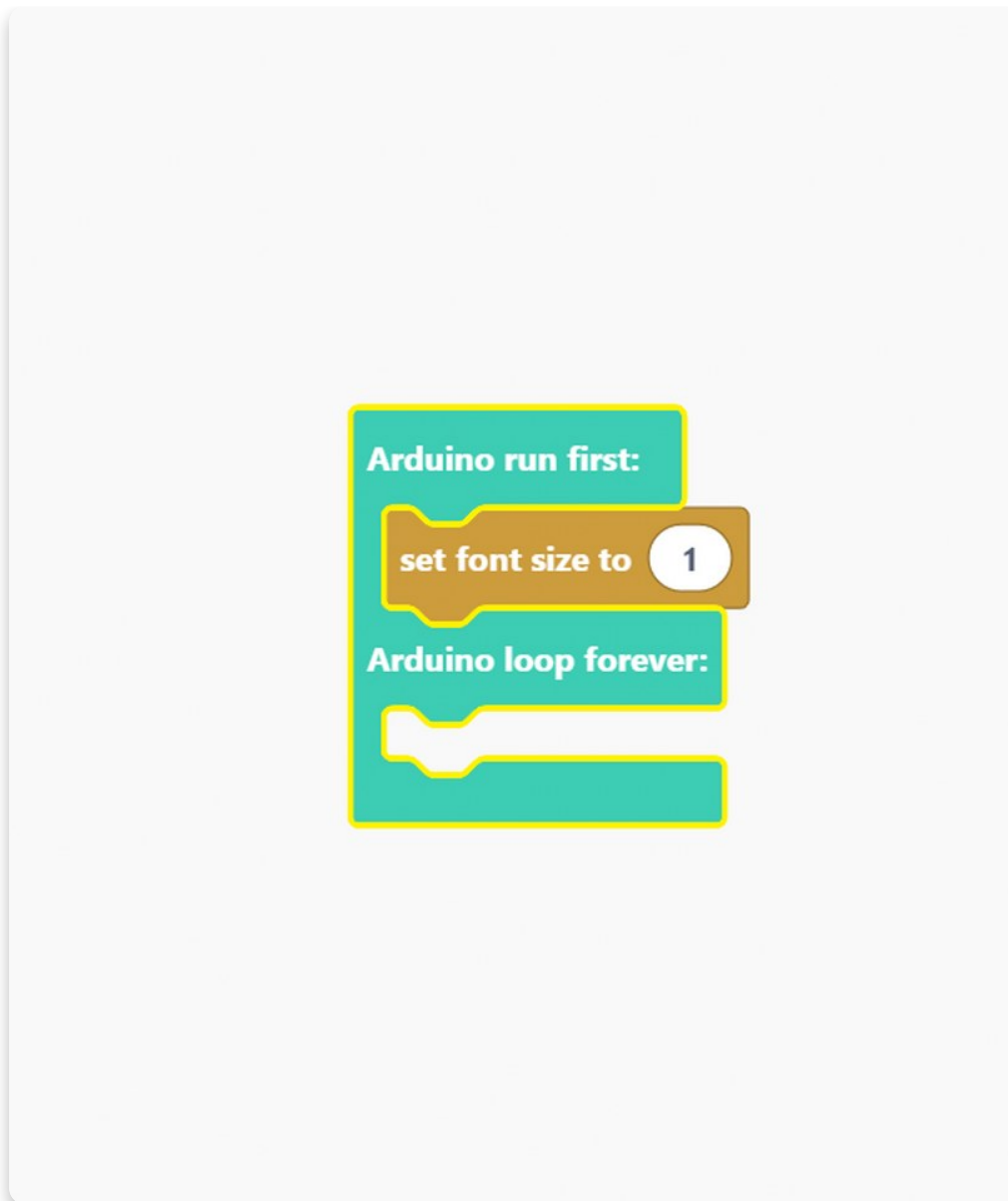
Now that you know a thing or two about CircuitBlocks, it's time for **a bit more advanced sketch**.

The first few things we'll have to do are the same as the previous sketch.

Go to the **Display section**, and click on the "**set font size to**" block.

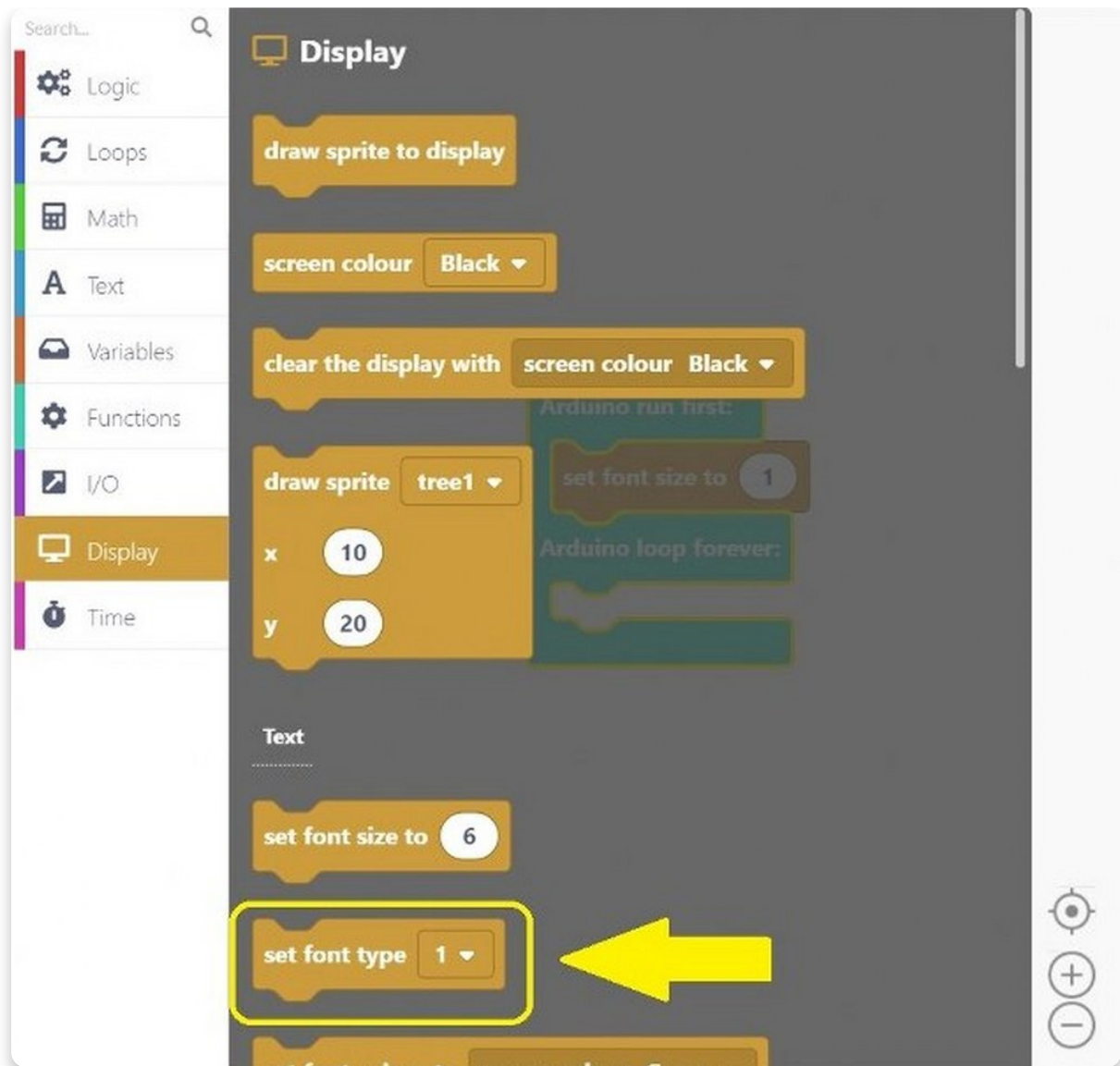


Once again, we'll change the **font size** to **1**.



The next thing to set is a **font type**.

As you already know, you can find that block in the **Display section** as well.

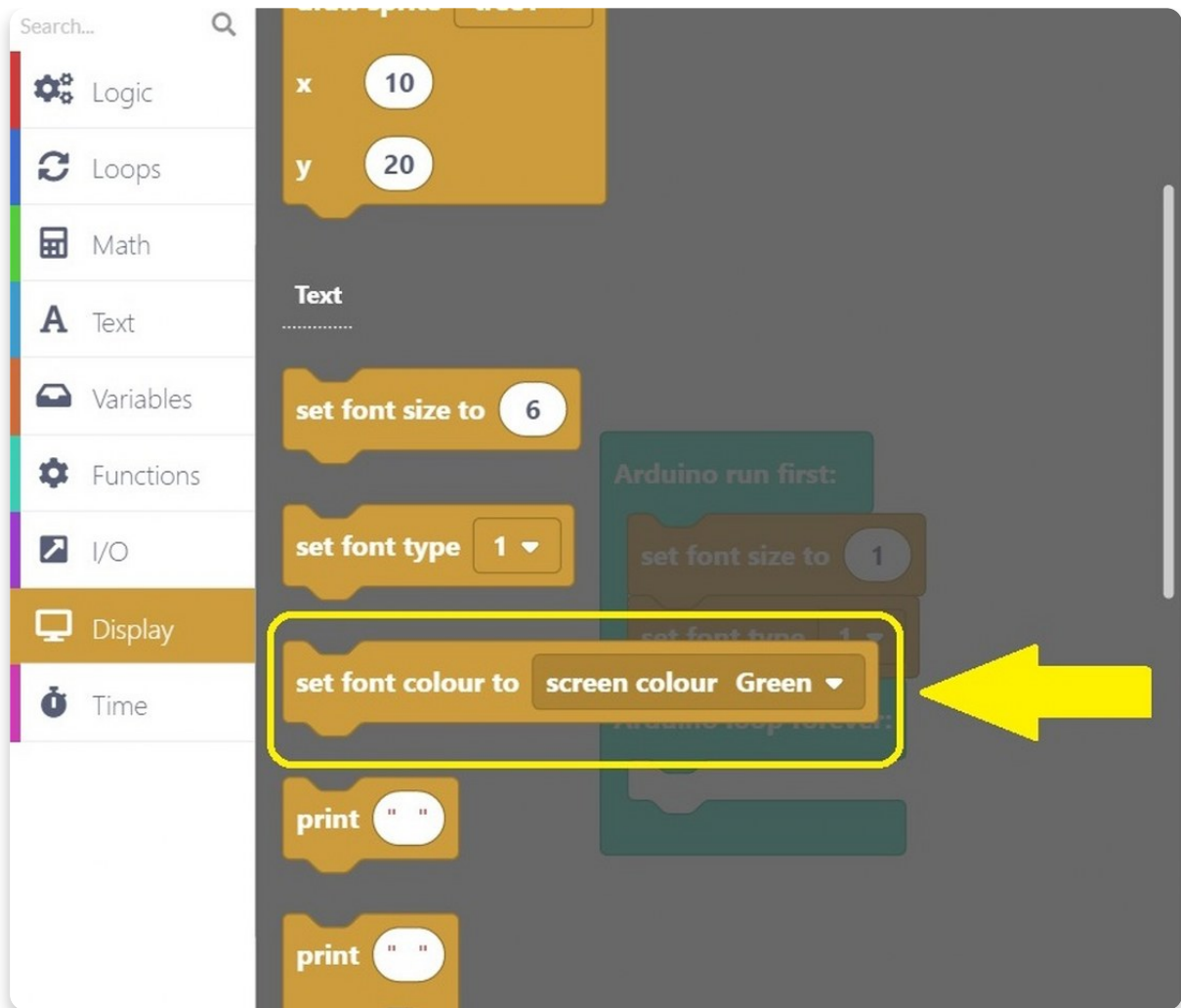


Click on the block and drag it into the drawing area.

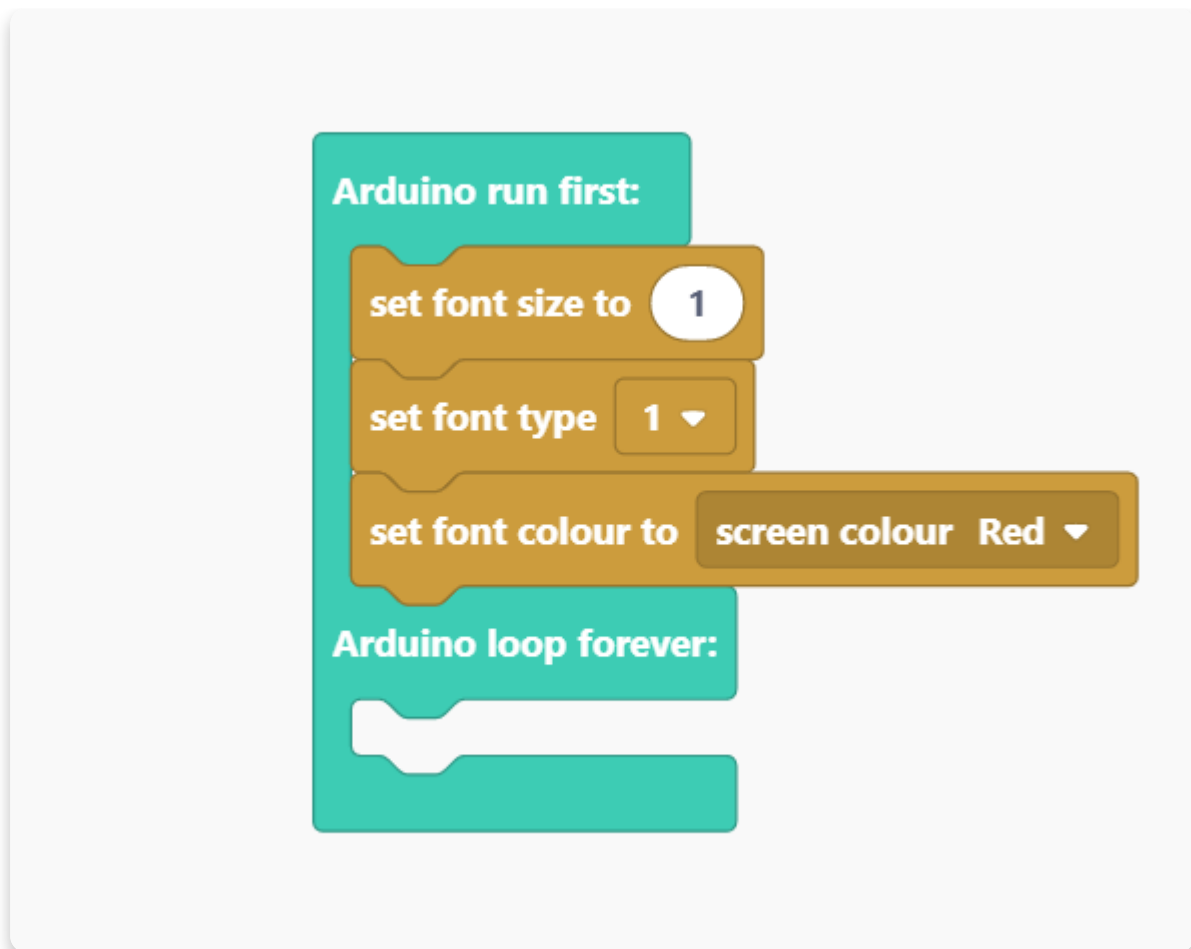




And now, let's set the **font color**.



You can choose whichever color you want, but we choose Red.



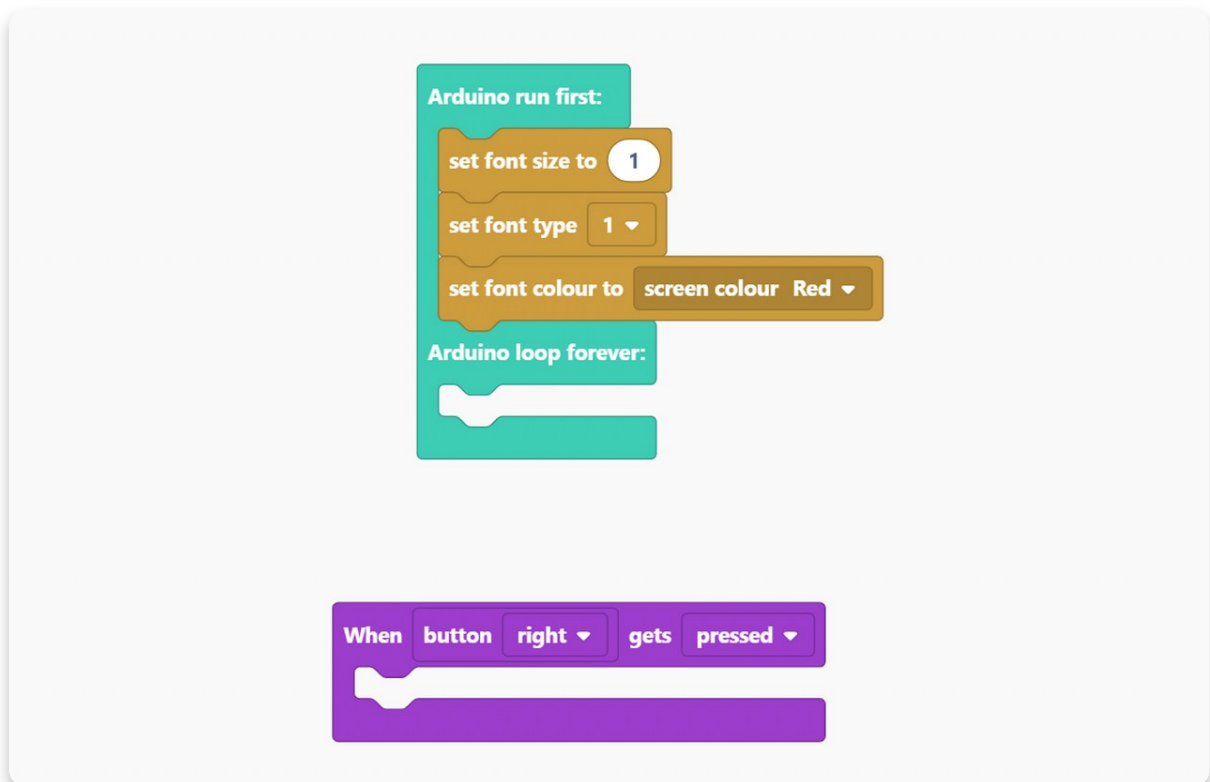
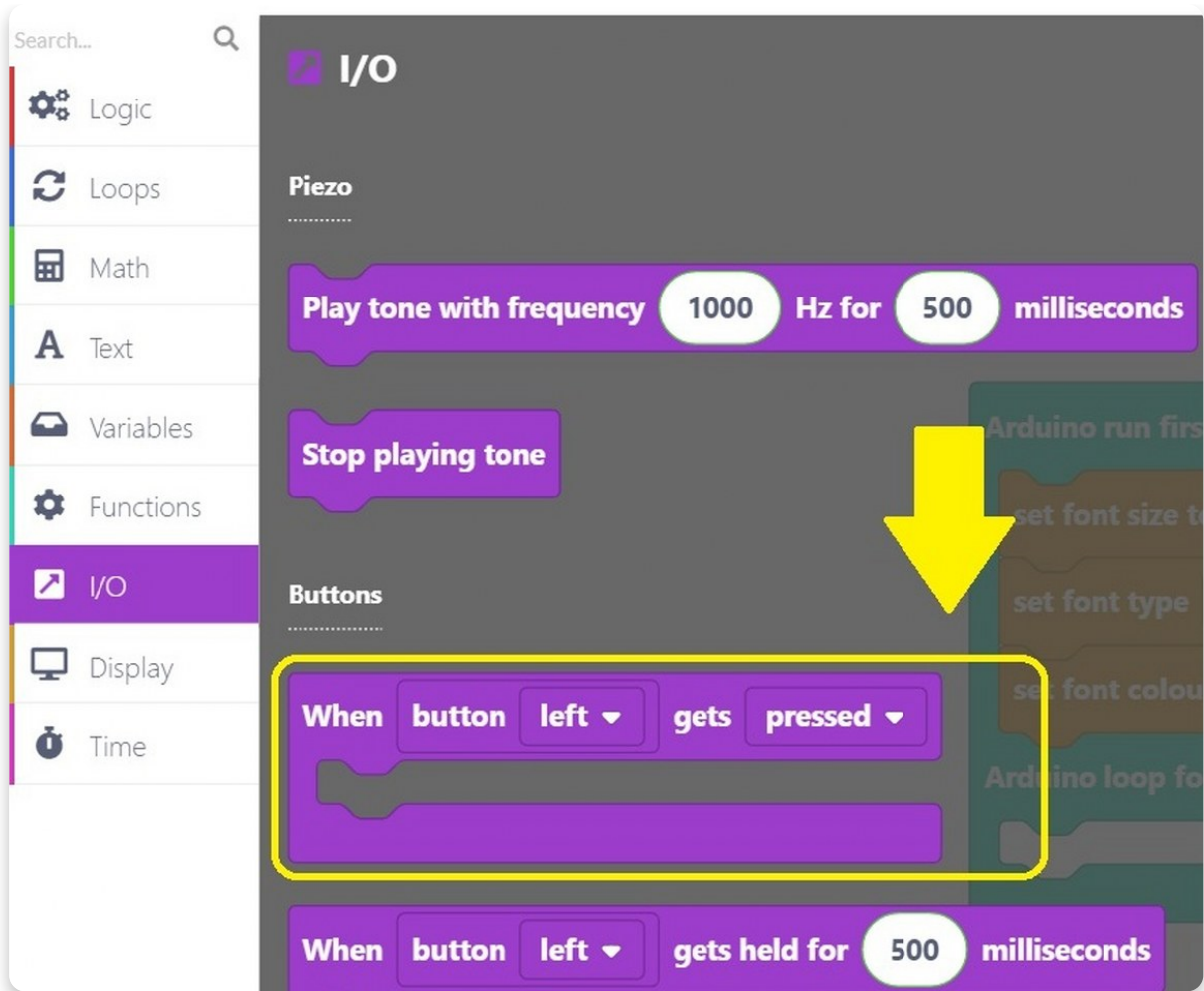
Now that we have a font created, let's change what happens when a particular event is triggered.

When you press Chatter's buttons, we trigger a particular event.

Luckily, we have a specific block defined for that, and it's under the **I/O** section. I/O stands for "**Input/output**".

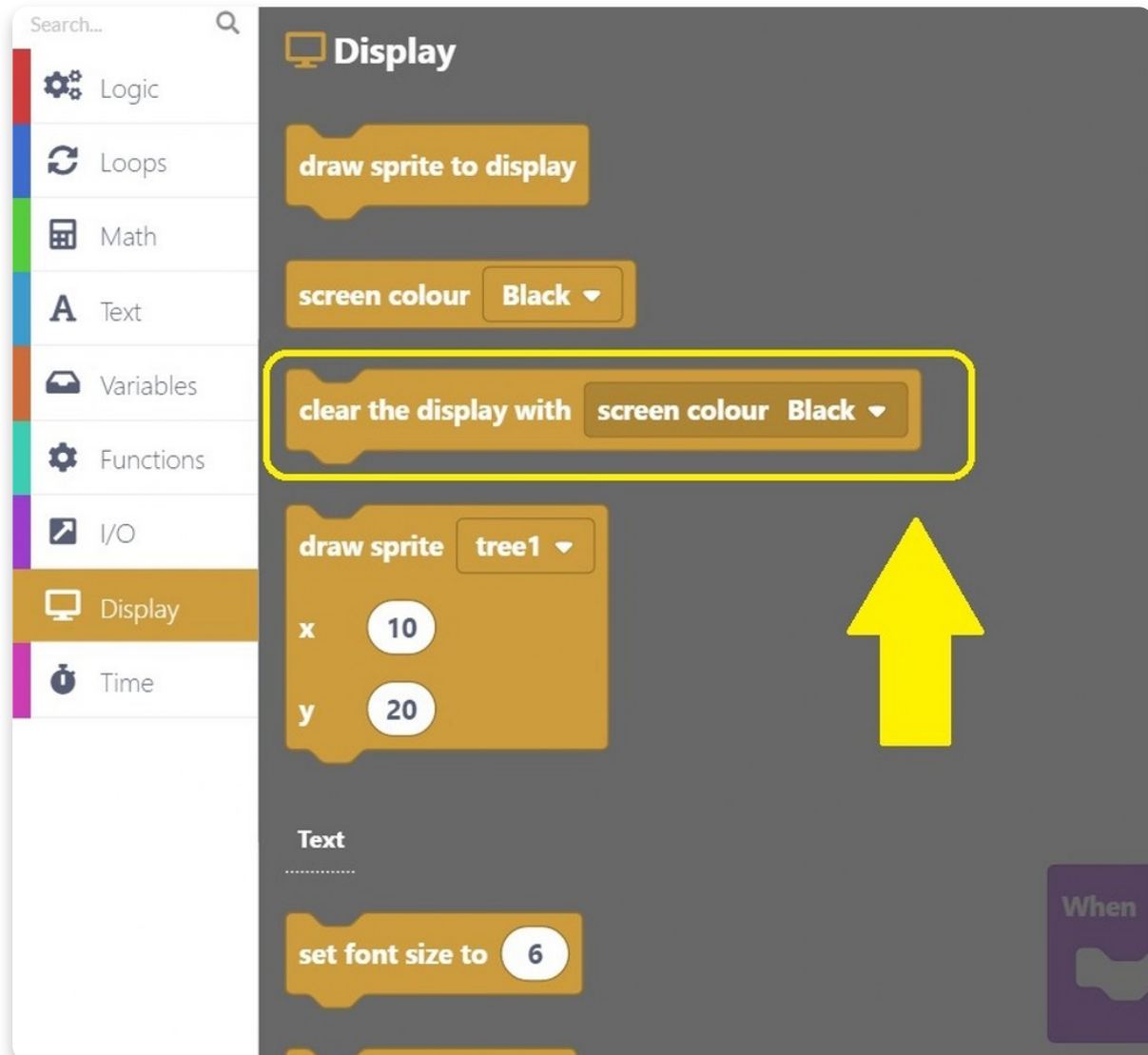
Chatter's buttons are the so-called input devices because they send an electrical impulse to Chatter's computer when triggered. Chatter's display is an example of an output device because Chatter sends signals to it to display information.

You need to find this purple block named "**When button left gets pressed**". Place the block onto the drawing area.

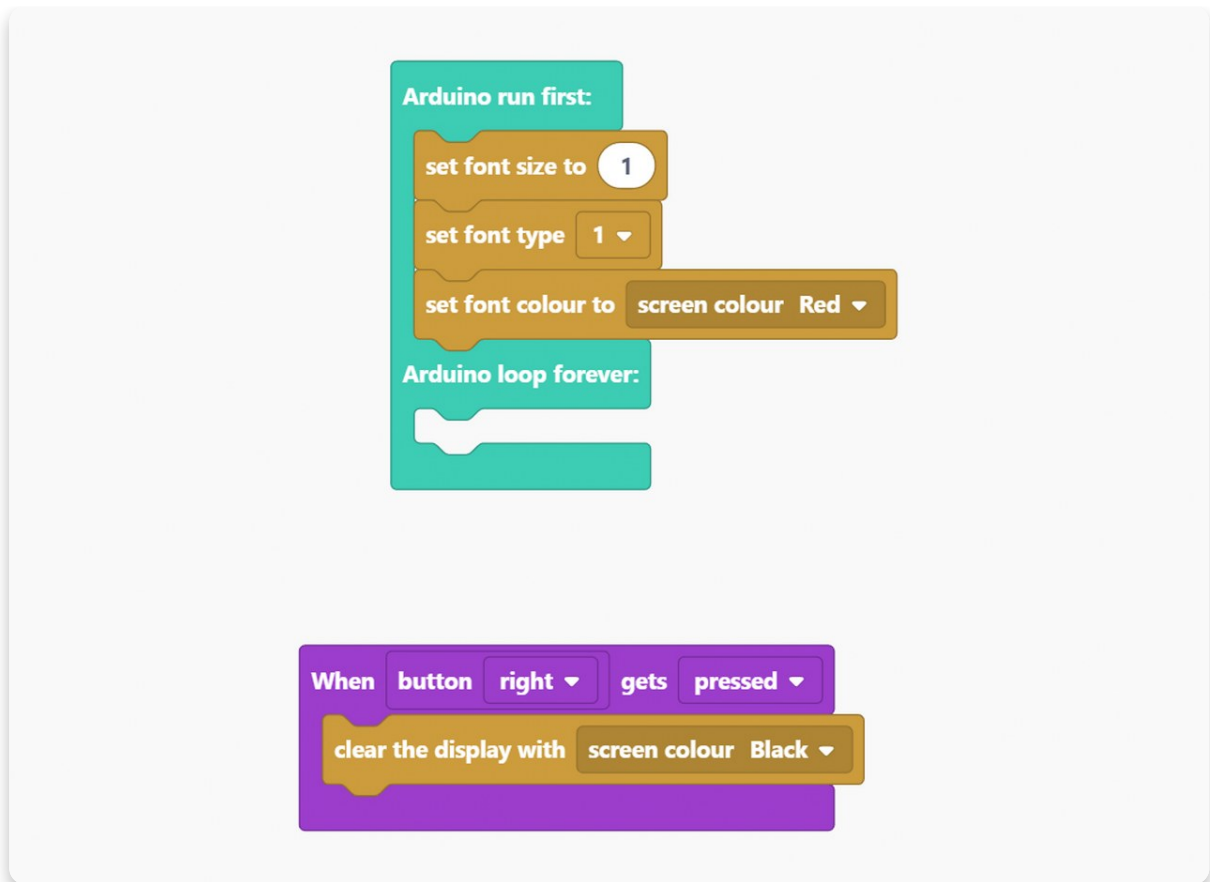


We decided to start with the **right** button, so you can change the button you're working on to keep up with us.

Now, go back to the **Display section**, and find a "clear a display with screen color" block.

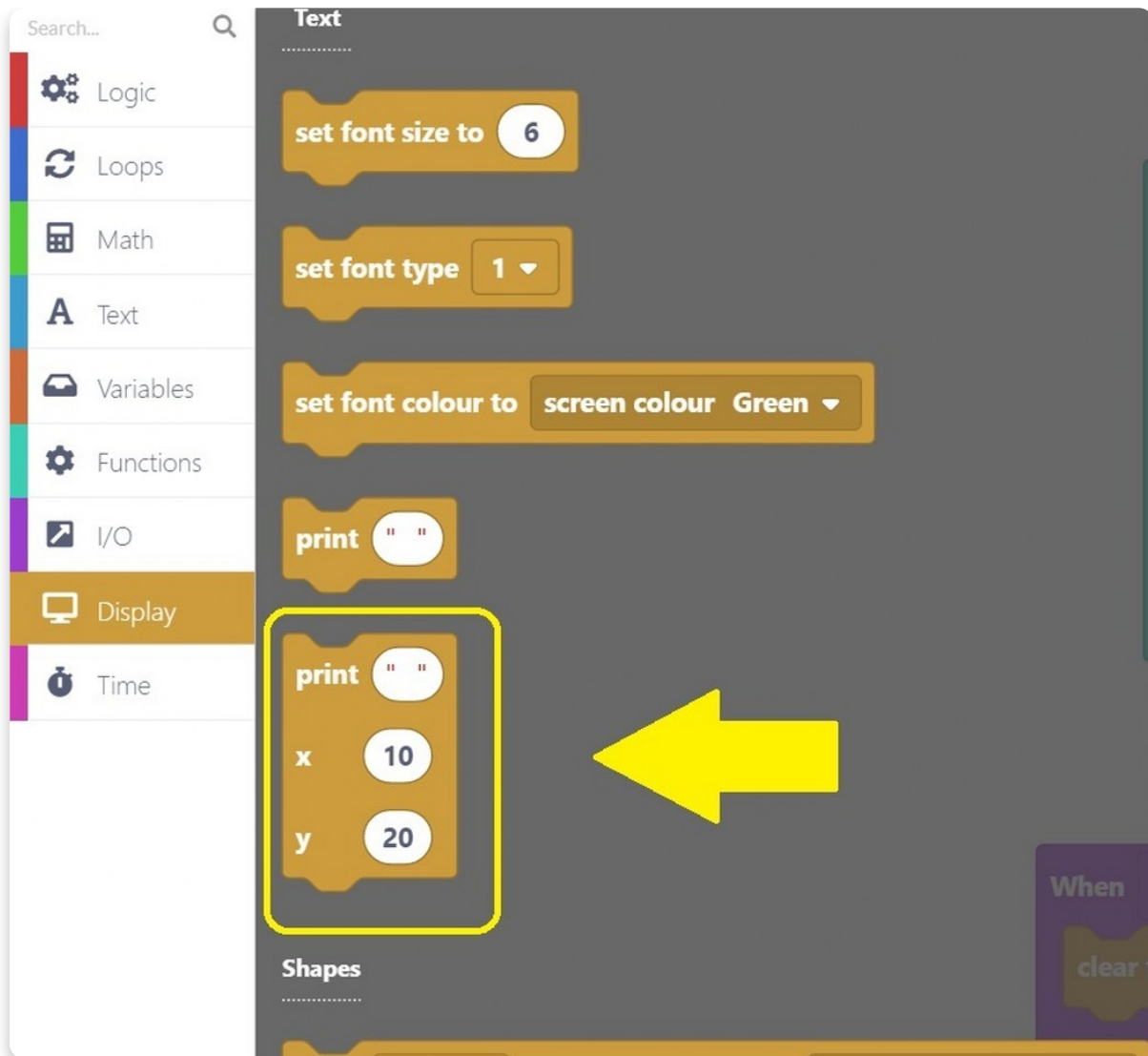


Drag it into a purple **I/O block**.



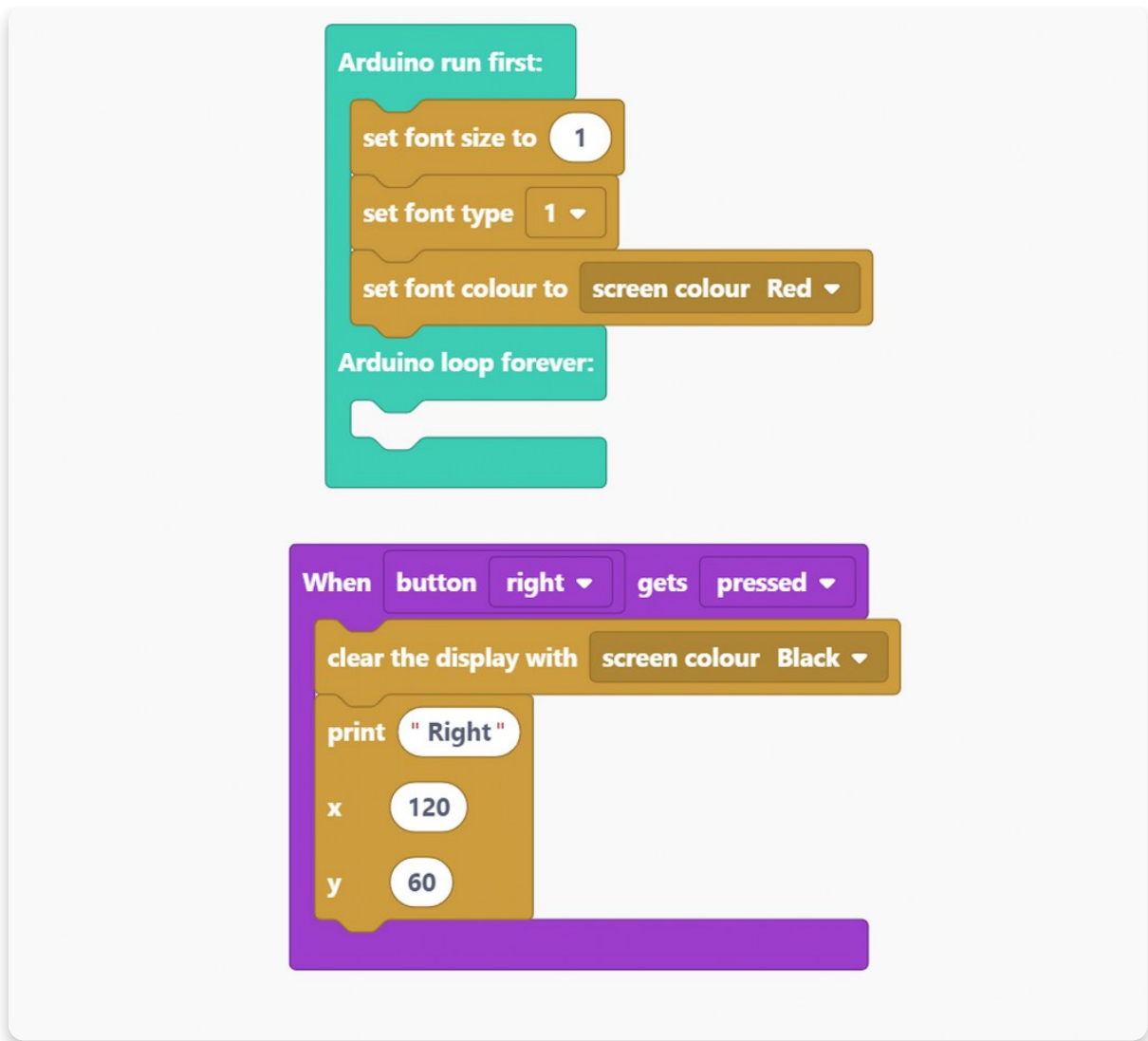
So, what you did now is you gave Chatter a command that when you press the right button, the first thing that'll happen is that the display will turn black.

The next block we'll search for is the print one (with the three white circles determining the text and the coordinates).



Drag it into a purple I/O block.

Change the **print** part into the "**Right**" and the **coordinates** into **120** and **60**.



And as you have already learned, to make this code work, you need to put a "**draw sprite to display**" block.



Search...

- Logic
- Loops
- Math
- Text
- Variables
- Functions
- I/O
- Display**
- Time

### Display

**draw sprite to display** ←

screen colour **Black** ▾

clear the display with screen colour **Black** ▾

draw sprite **tree1** ▾

x **10**

y **20**

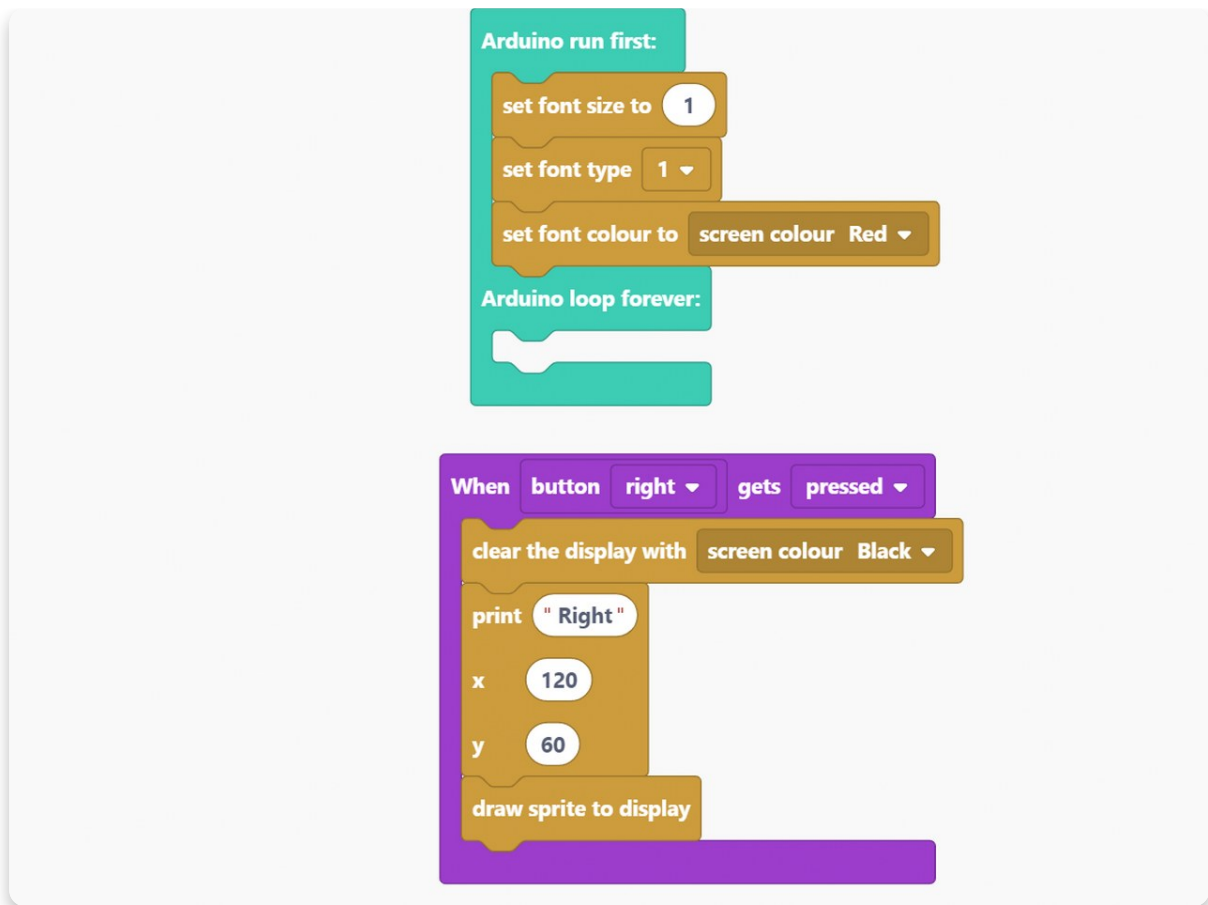
Text

.....

set font size to **6**

Arduino r  
set font  
set font  
set font  
Arduino lo

When butto  
clear the d  
print " Rig  
x 120  
y 60



Now we have to do the same code for the rest of the buttons, and the easiest way to do that is to **duplicate** the block you already made.

To **duplicate** the block, click the right mouse button while placed on the purple part of the block. The menu should say "**duplicate 5 blocks**". Click on that, and an identical block should be created.

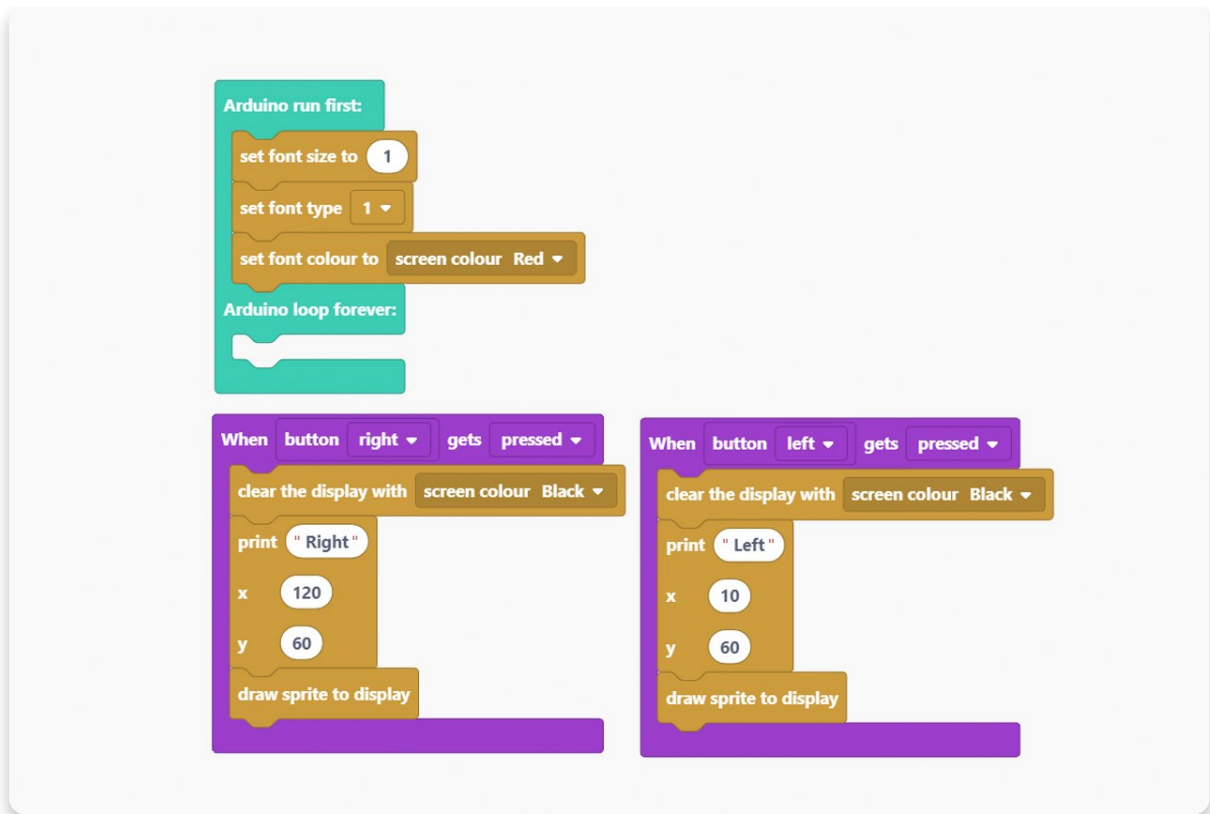
Drag it on the drawing board, and let's change it slightly.

So, now we'll code what would happen if the **left** button gets **pressed**.

We decided to leave the screen color black, but now we'll print "Left" on the screen.

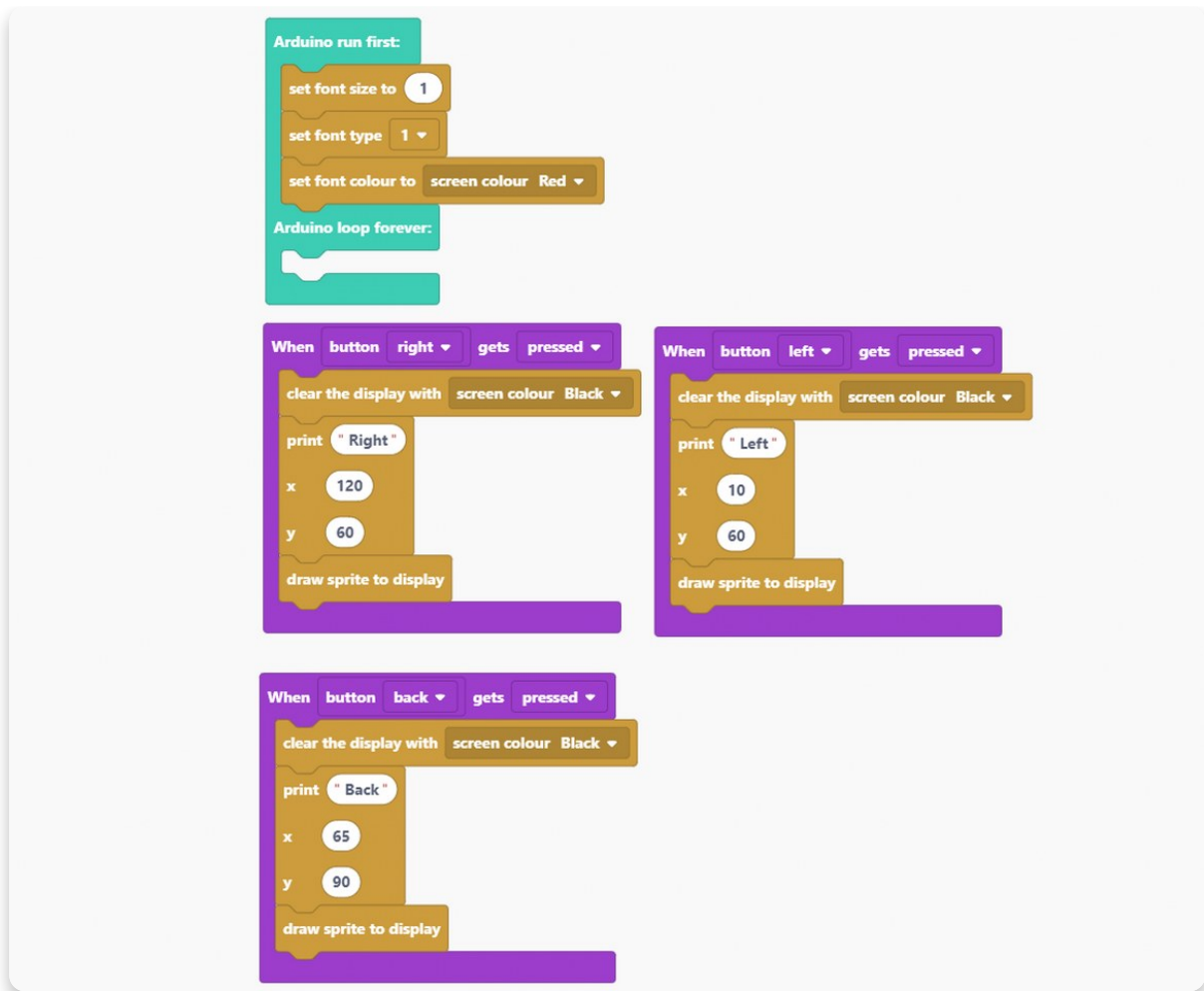
Also, we changed the **coordinates** of the text to be **x=10** and **y=60**.

In the end, once again, there's a "**draw sprite to display**" block to make sure the code will appear on display.



You'll have to duplicate those five blocks three more times.

Let's code the **button** you are using to go **back** on Chatter for a start.



As you can see, we decided to print "**Back**" and put coordinates to be **x=65** and **y=90**.

Now, we'll make two duplicates for the enter button.

One input code will decide what will happen if the **enter** button gets **pressed**, and the other input code will determine what will happen if the **enter** button gets **released**.

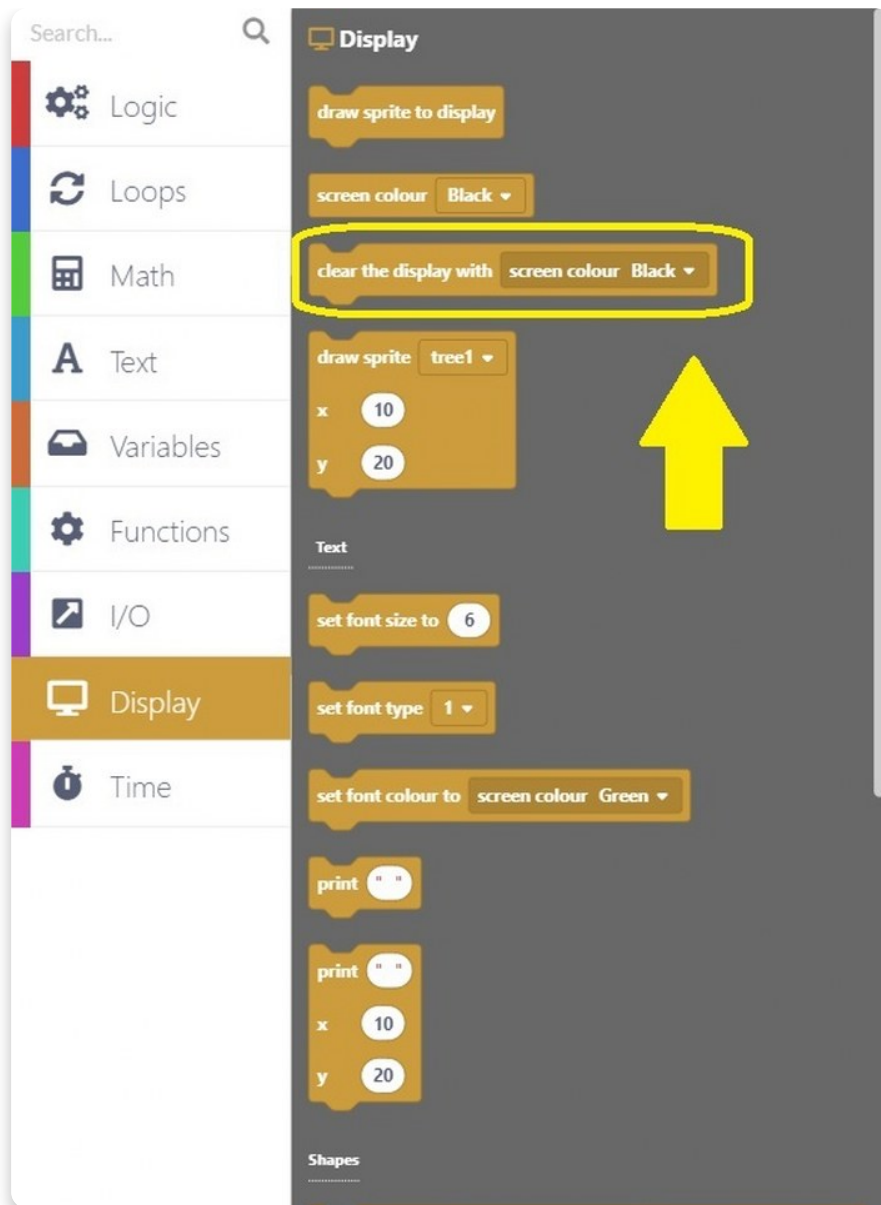


Please, make sure that the block you'll use while releasing an enter button prints **"Enter release"** and the other one **"Enter press"**.

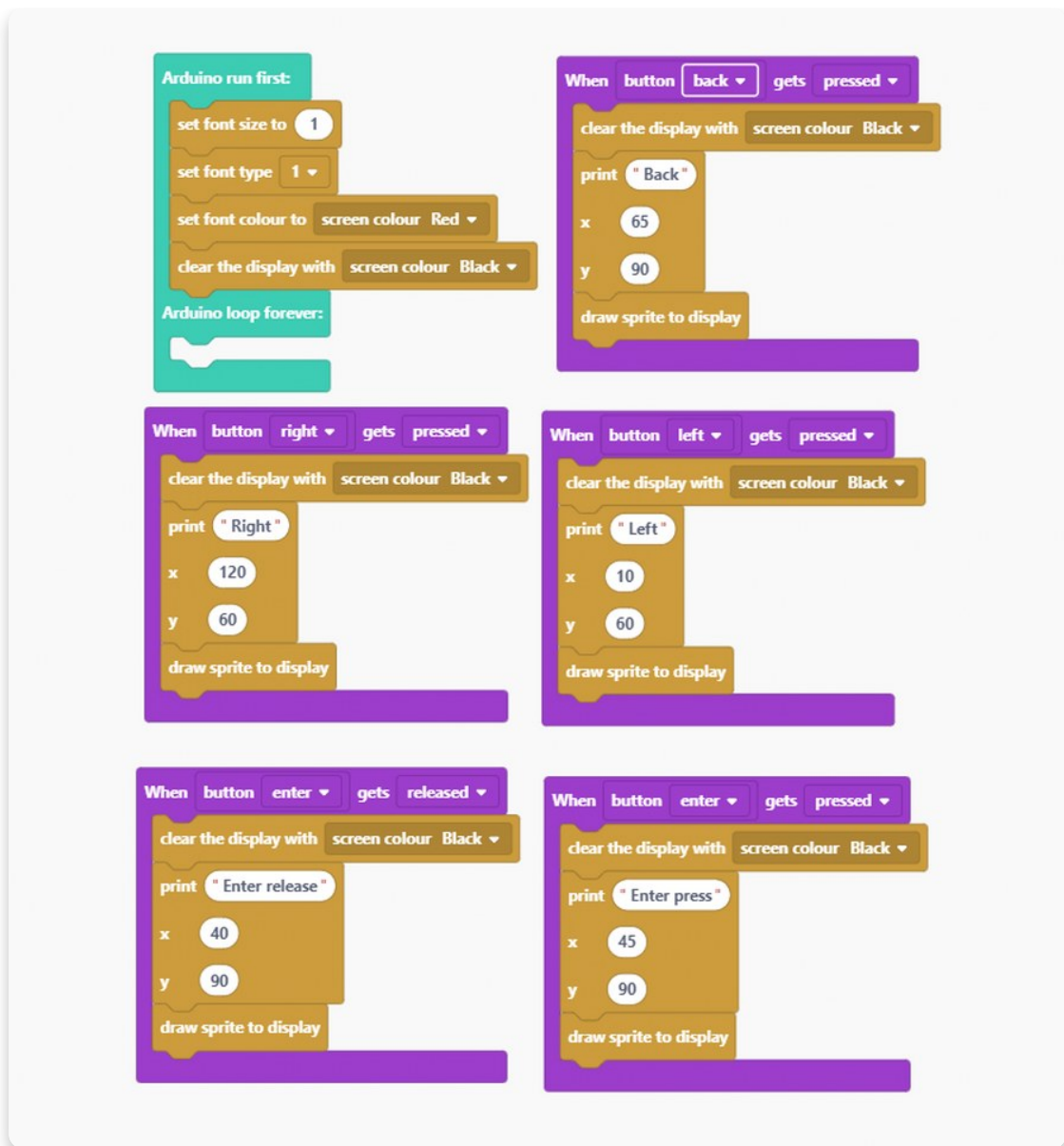
**Check the coordinates in the photo above.**

If you did all the **input/output blocks**, it's time to get back to **"Arduino run first"** block.

Open the **Display section**, and choose the **"clear the display with screen colour Black"** block.

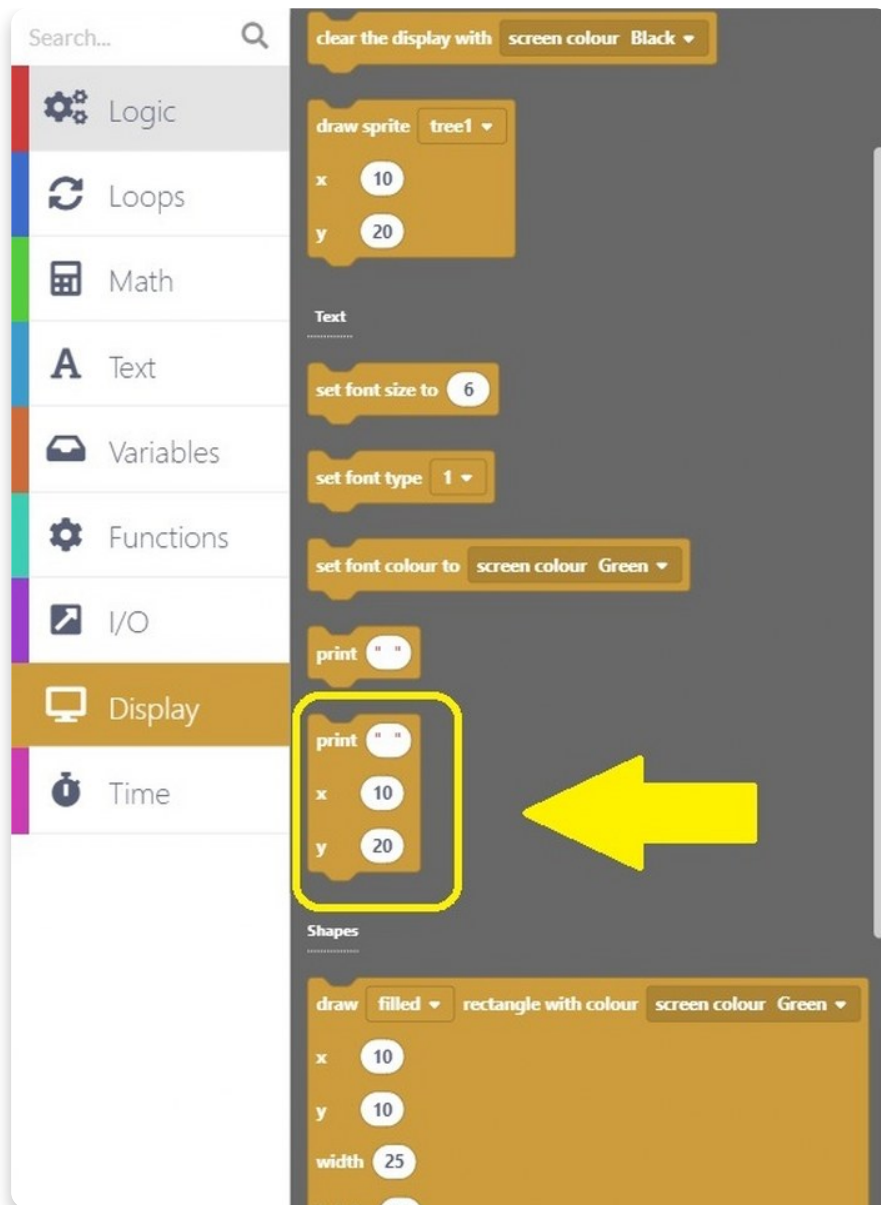


Click on the block, drag it to the drawing area, and place it under the "**Arduino run first**" part of the sketch.



The next thing we'll need is also one of the well-known blocks.

Go to the **Display section** and choose this particular **"print"** block. We'll need two of them, but you can duplicate the other one easily.



Your sketch should look like this:



```

Arduino run first:
set font size to 1
set font type 1
set font colour to screen colour Red
clear the display with screen colour Black
print "Press one of the"
x 30
y 60
print "top row buttons!"
x 31
y 70

Arduino loop forever:
When button back gets pressed
clear the display with screen colour Black
print "Back"
x 65
y 90
draw sprite to display

When button right gets pressed
clear the display with screen colour Black
print "Right"
x 120
y 60
draw sprite to display

When button left gets pressed
clear the display with screen colour Black
print "Left"
x 10
y 60
draw sprite to display

When button enter gets released
clear the display with screen colour Black
print "Enter release"
x 40
y 90
draw sprite to display

When button enter gets pressed
clear the display with screen colour Black
print "Enter press"
x 45
y 90
draw sprite to display

```

Make sure to print **"Press one of the"** in the one block and **"top row buttons!"** in the other one.

Also, please check the coordinations we put for each block.

Why did we print those two last sentences?

We choose to print those out simply to make this code look better and more organized. When your Chatter turns on after the code compiles, the first thing that will happen on display is **"Press one of the top row buttons!"** text will appear. After that, you can indeed press those buttons and check if the rest of your code is working properly.

For the end, make sure you place a **"draw sprite to display"** block at the end of the **"Arduino run first"** section for the **"Press one of the top row buttons!"** to appear.



Ta-daa!

**Hit a big red button, wait for the code to compile, and test the buttons.**

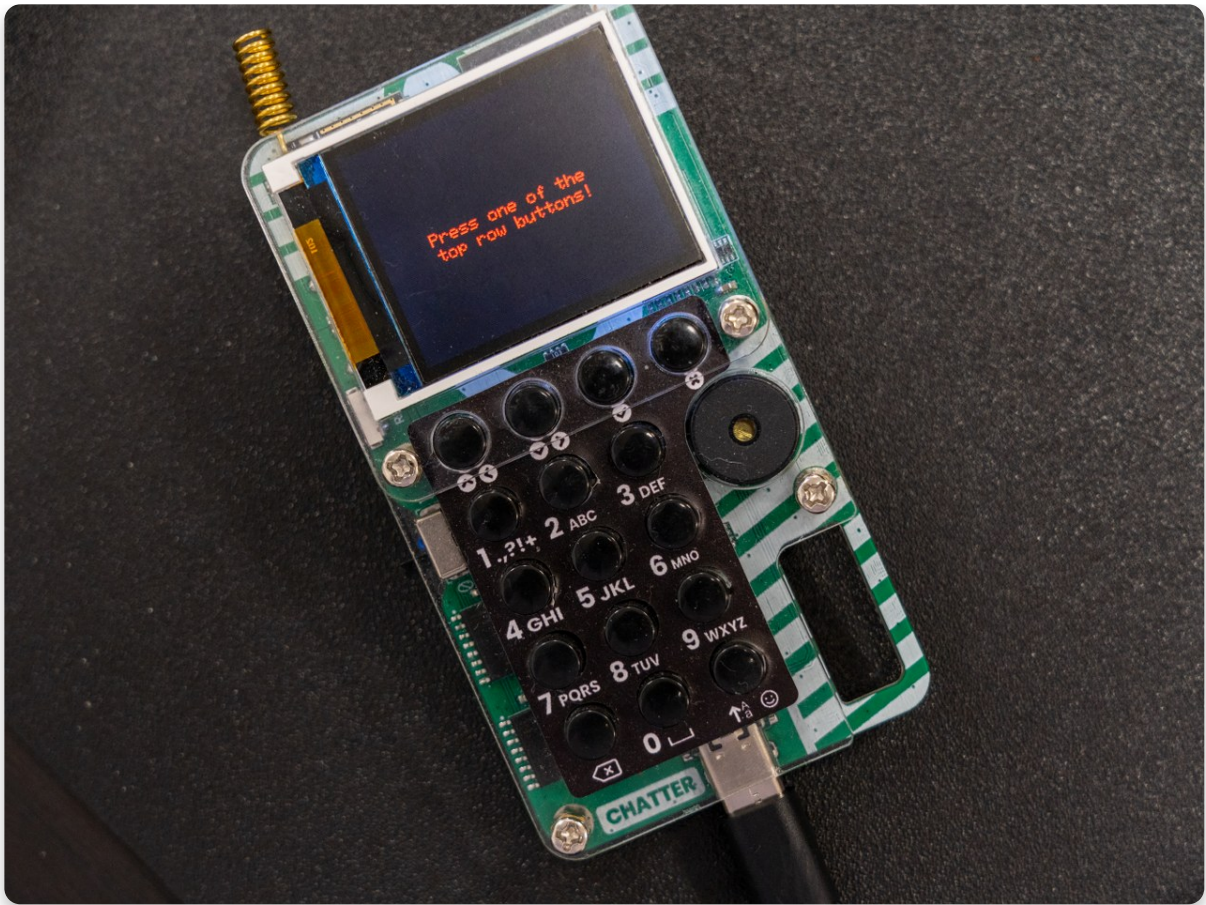
Once the code is compiled, the Chatter will restart, and the screen will turn red with a black text saying, "Press one of the top row buttons!".

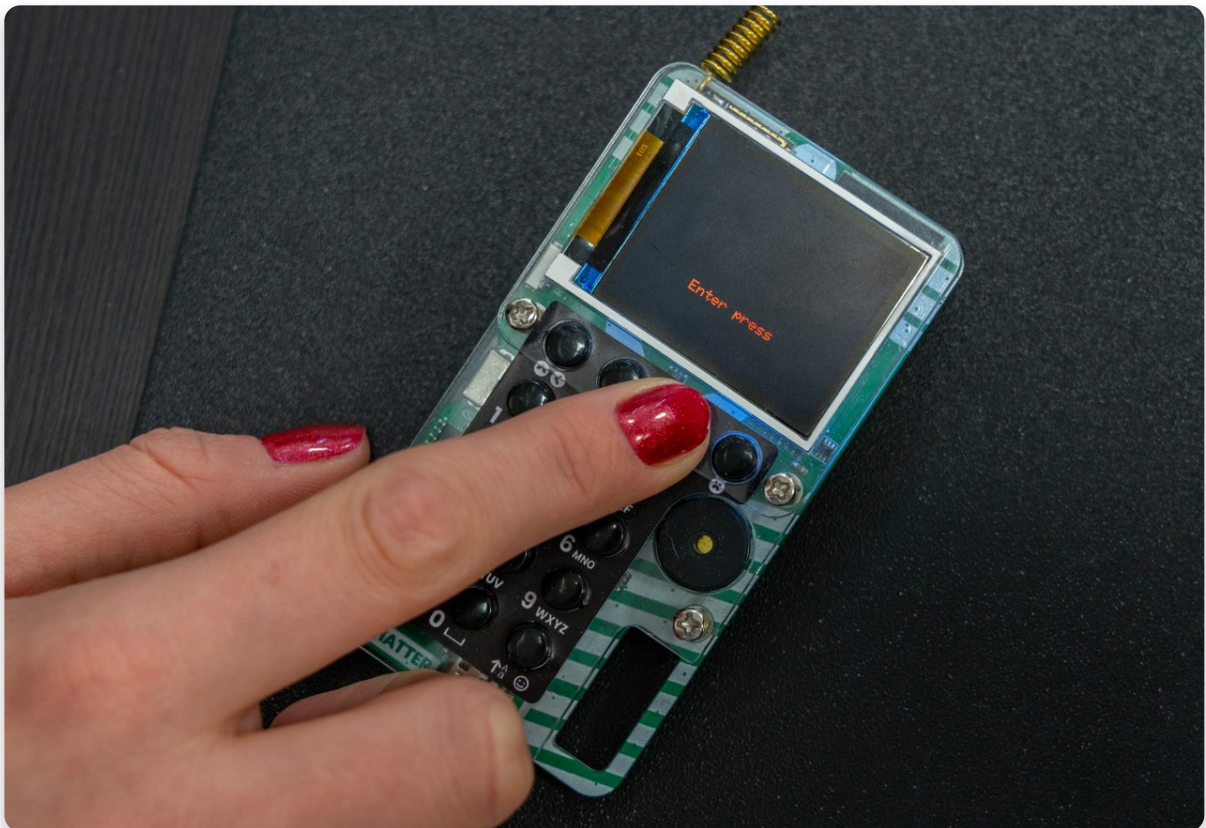
Now, you can do as said and press one of the top row buttons. Once you press one of the buttons, the screen will turn black.

Each button will trigger different text saying which one you pressed. So, if you press the right button, the screen will display "right", and if you press the left one, it will display "left". Also, we put the coordinations so that the right button triggers text on the right side, and the left button triggers text on the left side.

It's a bit different with the enter button. As you can see, we kept the same coordinations, but unlike the other buttons, we used this one twice - for pressing and releasing. So, as the command said, when you press enter, the screen will display "Enter press", and when you release enter, the screen will display "Enter release".

Check the photos below to see what is supposed to happen:







## Buzzzzzz

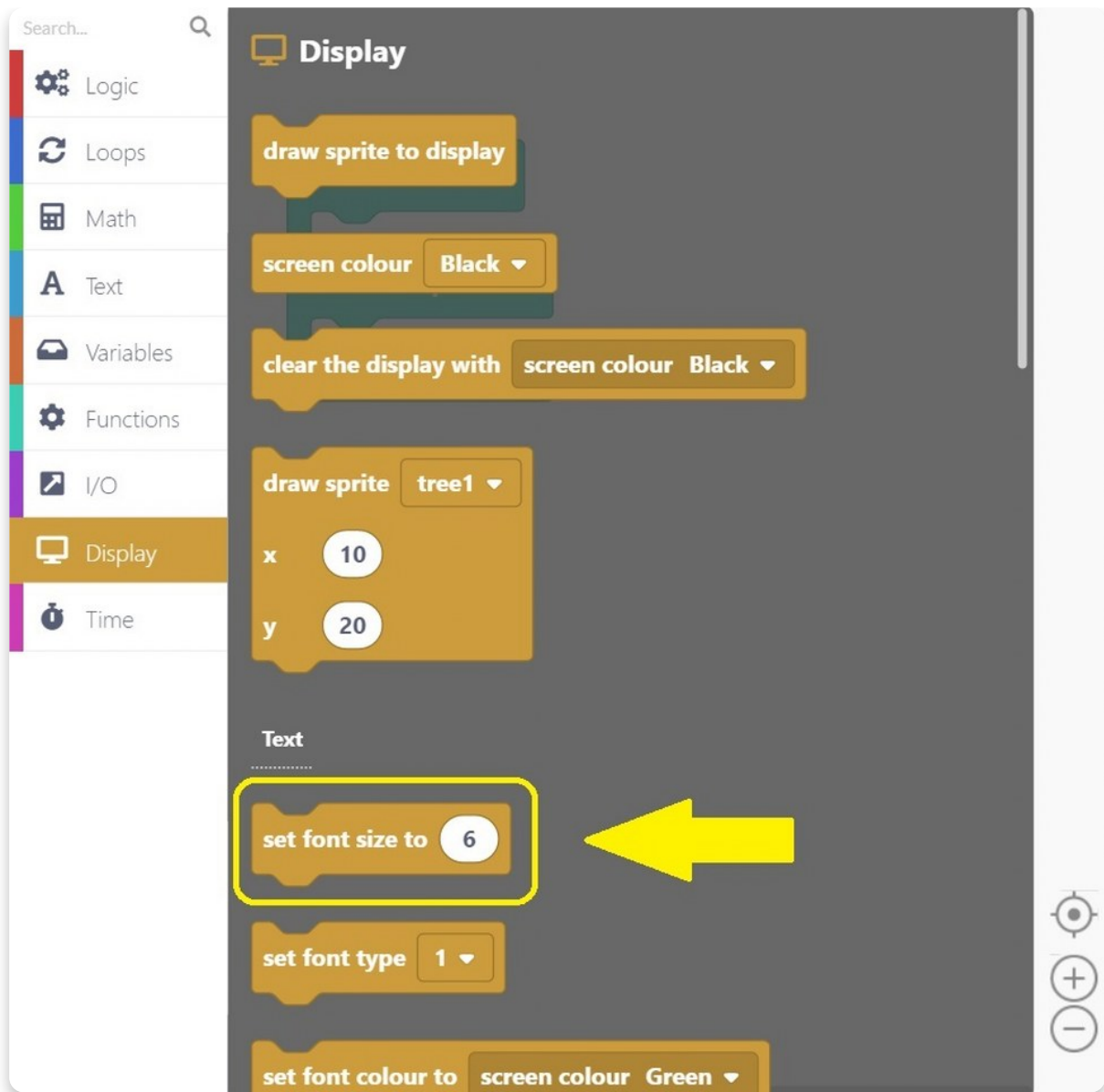
Let's learn what to do with the **Piezo buzzer** you soldered onto your Chatter.

As the word itself says, the buzzer is used for making buzzing sounds.

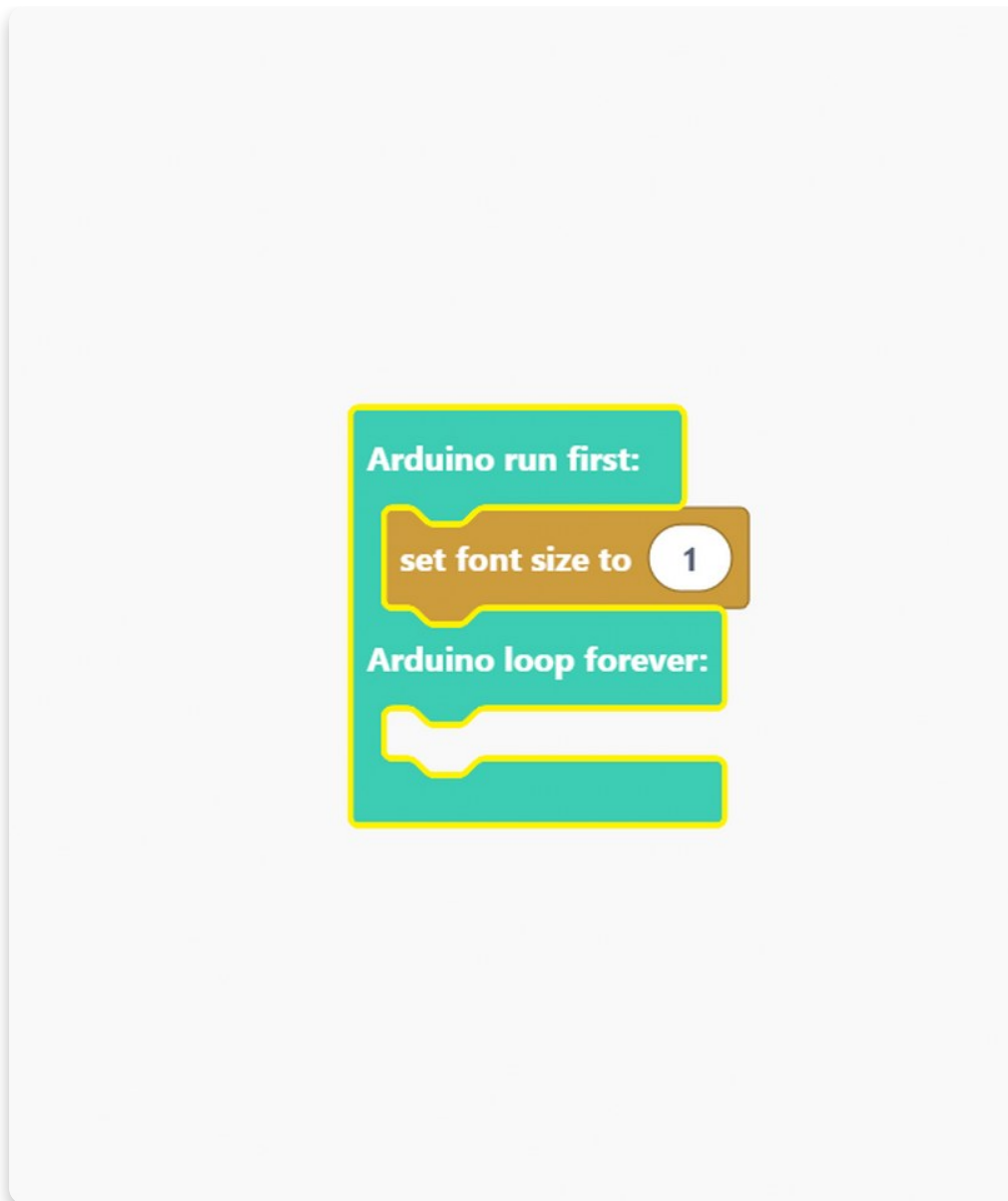
We'll make a very similar sketch to the last one, but this time, **pressing the buttons will trigger a particular sound to come out of the buzzer.**

## Let's start!

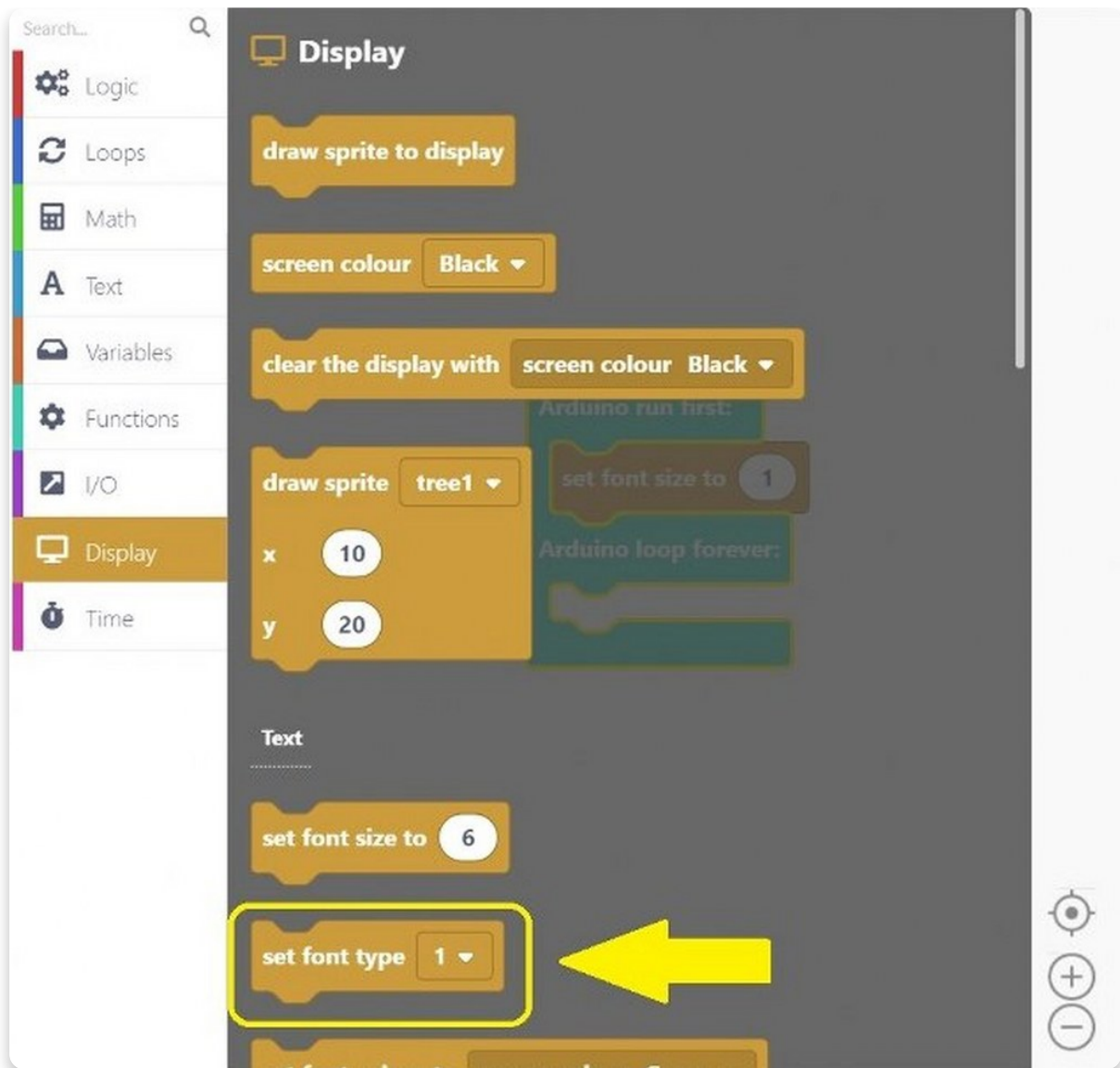
For the beginning, go to the **Display section**, and click on the "**set font size to**" block.



We'll also change the font size to 1 here, as we did for the last two sketches.



The next thing to set is a **font type**, another block that you can find in the **Display section** as well.

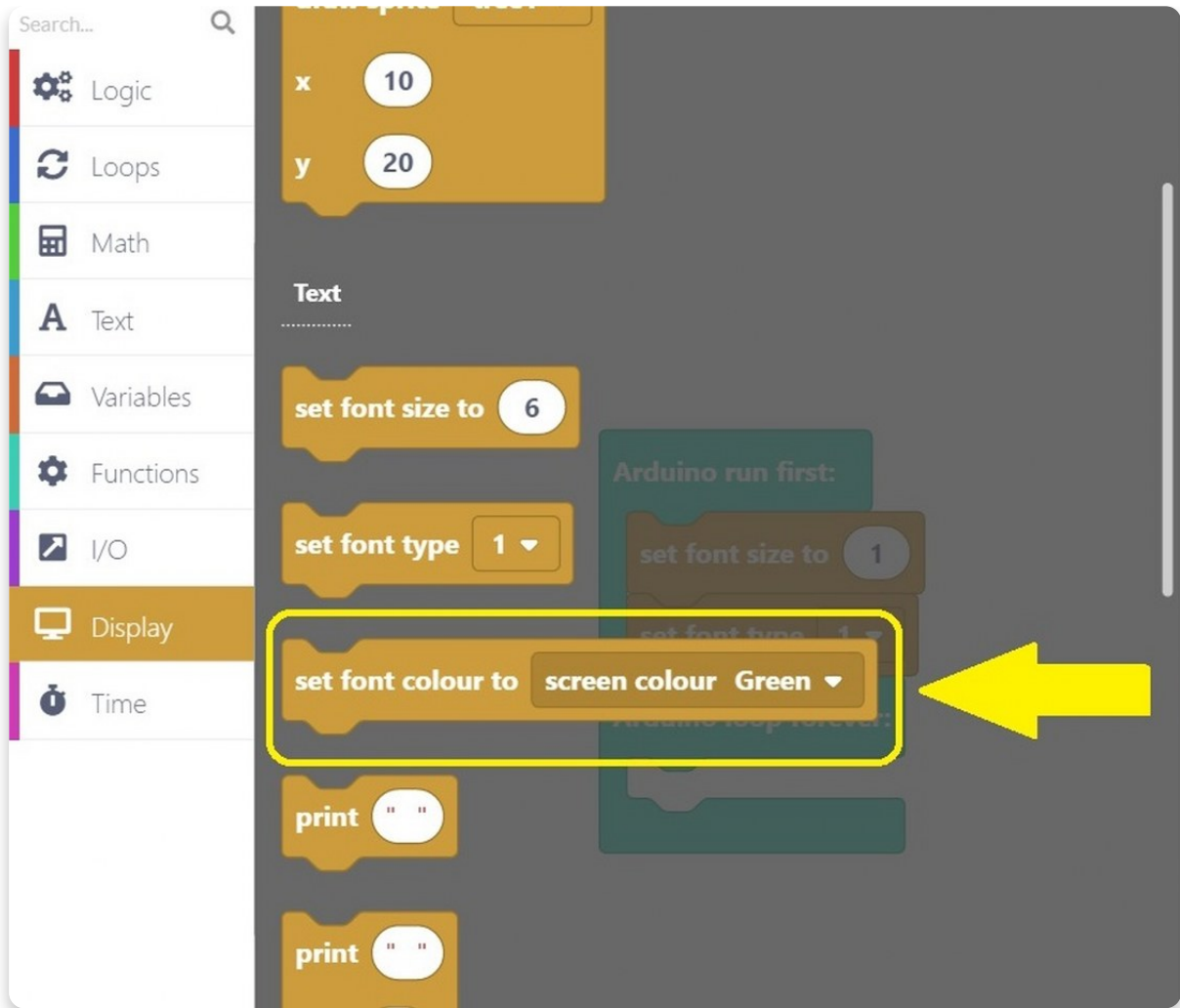


Click on the block and drag it into the drawing area.

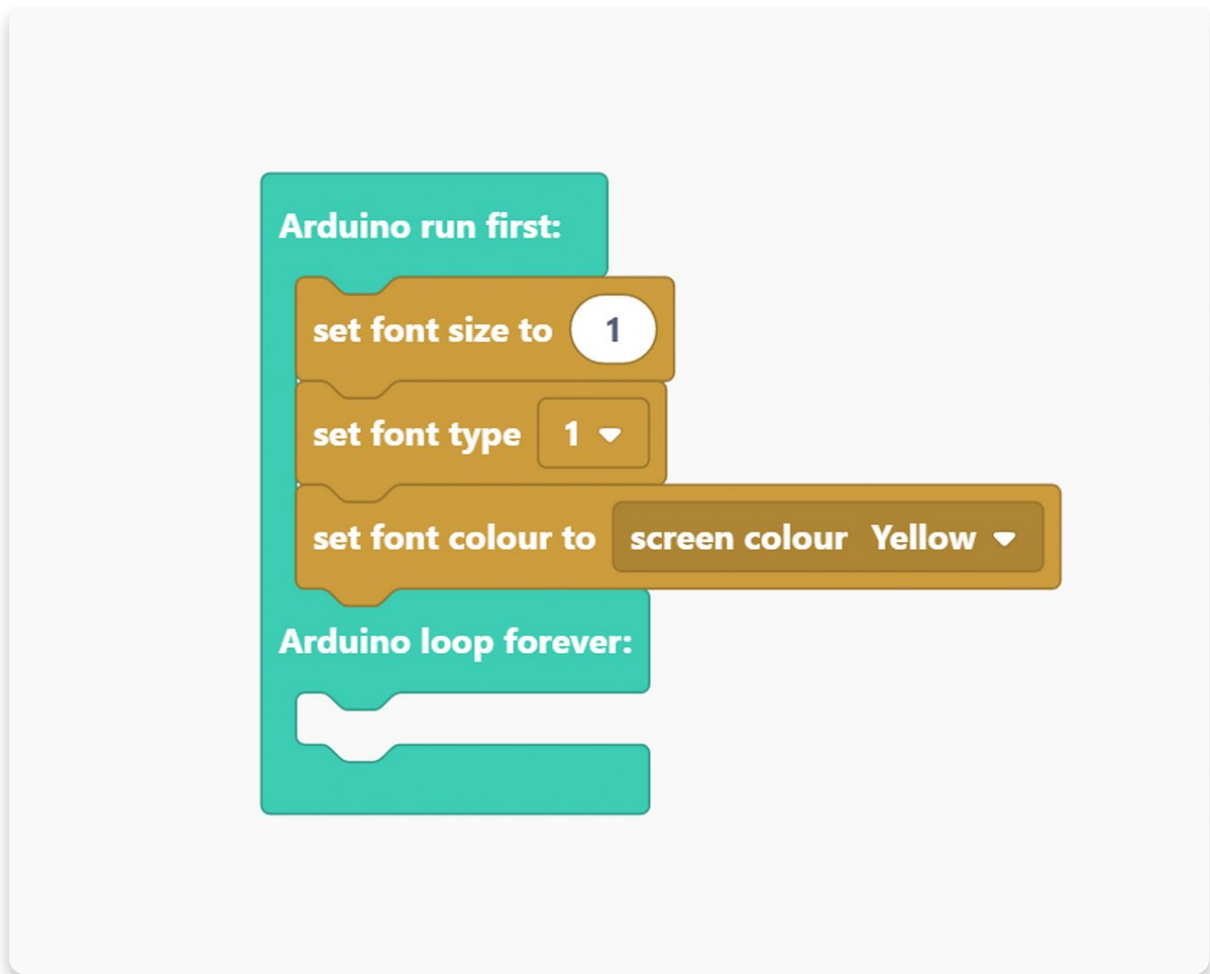




And now, let's set the **font color**.



This time we choose our font to be **yellow**.



Now that we have a font created let's change **what happens with the buzzer when a certain button gets pressed or released.**

Like in the previous sketch, we'll use blocks from the **I/O** section to determine what'll happen when specific buttons get pressed.

Search...



Logic

Loops

Math

Text

Variables

Functions

I/O

Display

Time

## I/O

### Piezo

Play tone with frequency **1000** Hz for **500** milliseconds

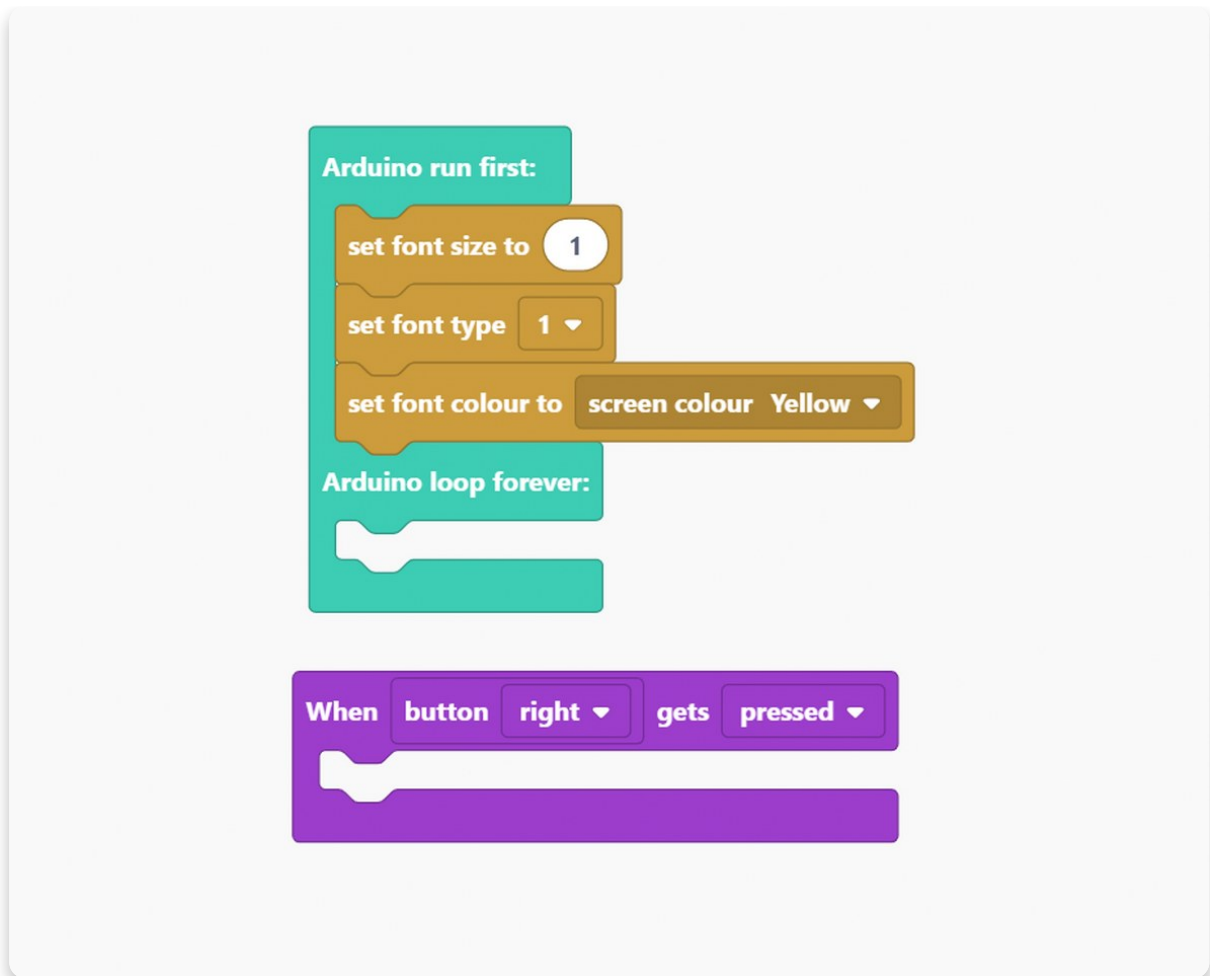
Stop playing tone



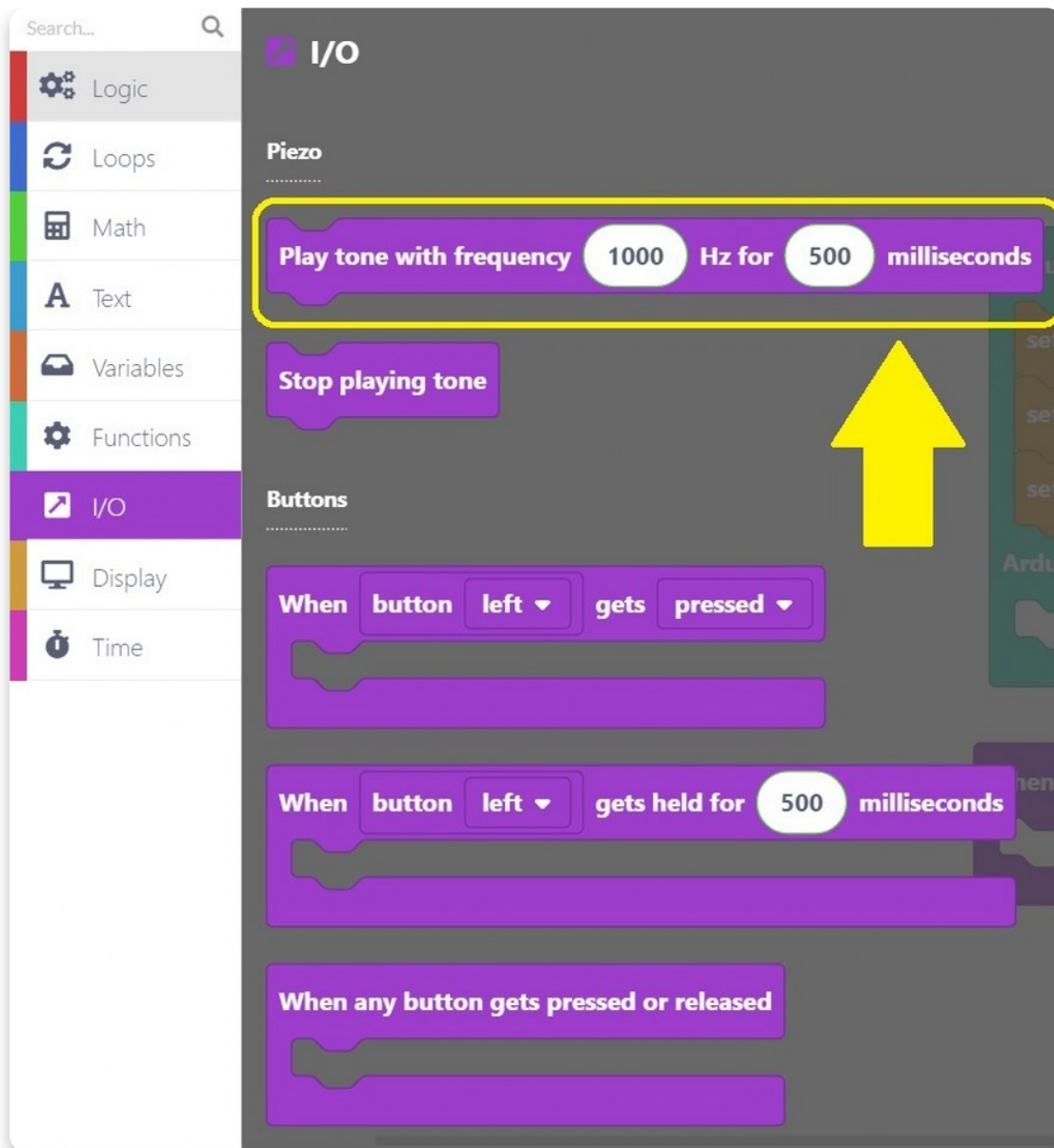
### Buttons

When **button left** gets **pressed**

When **button left** gets held for **500** milliseconds



Let's introduce you to a new block called **"Play tone with frequency 1000 Hz for 500 milliseconds"**.



Click on the block, and drag it into the I/O block on the drawing area.

We changed the **milliseconds** to be **200**, but you can make it however long or short you want to.

```
Arduino run first:  
set font size to 1  
set font type 1  
set font colour to screen colour Yellow  
Arduino loop forever:  
  

```

```
When button right gets pressed  
Play tone with frequency 1000 Hz for 200 milliseconds
```

Let's **clear** the **display** in **purple** once we press the right button.

Search...

- Logic
- Loops
- Math
- Text
- Variables
- Functions
- I/O
- Display**
- Time

### Display

draw sprite to display

screen colour Black ▾

**clear the display with screen colour Black ▾**

draw sprite tree1 ▾

x 10

y 20

Text

.....

set font size to 6

set font type 1 ▾

set font colour to screen colour Green ▾

Arduino run fir

set font size


set font type

set font colour

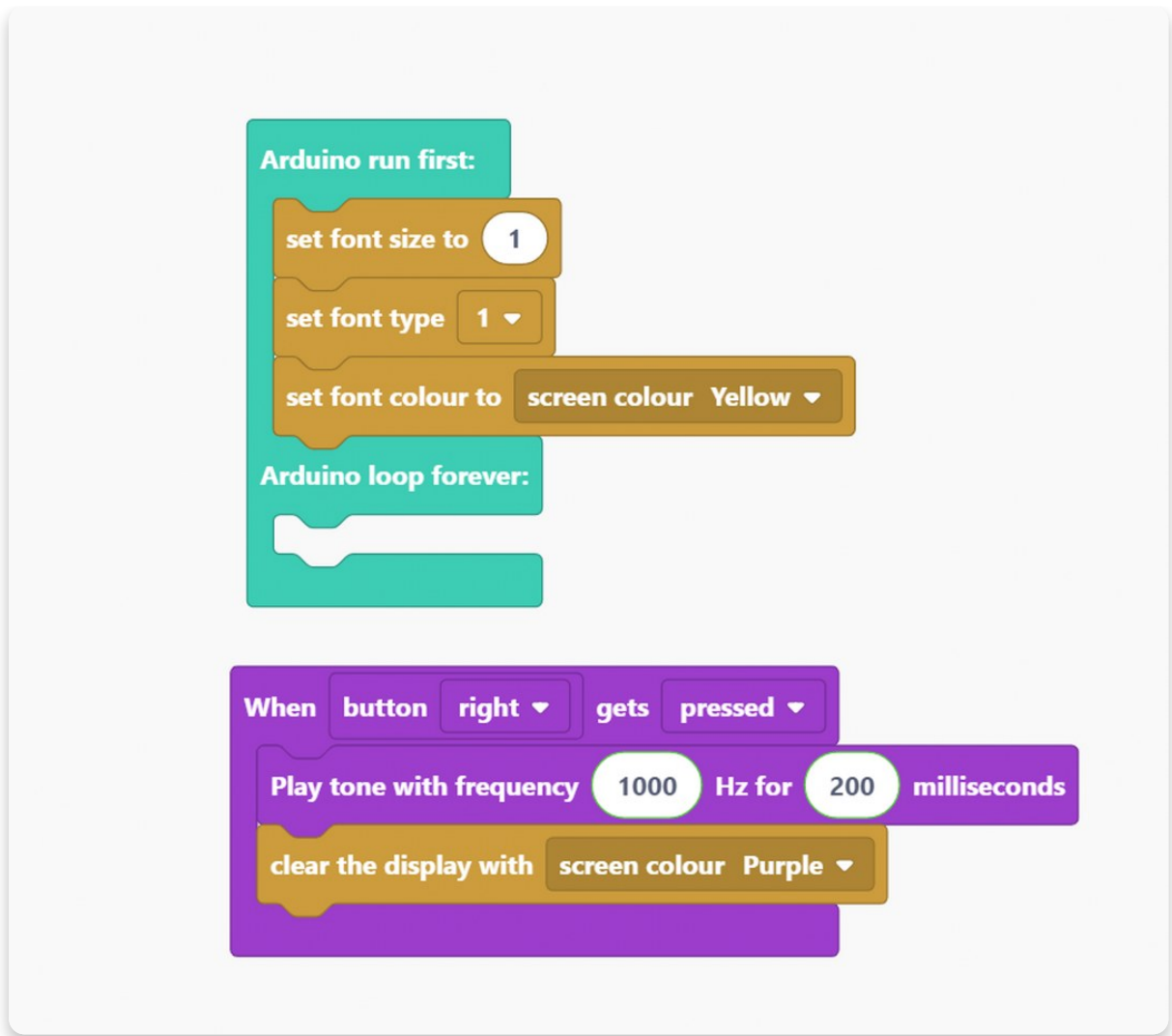
Arduino loop fo

When button

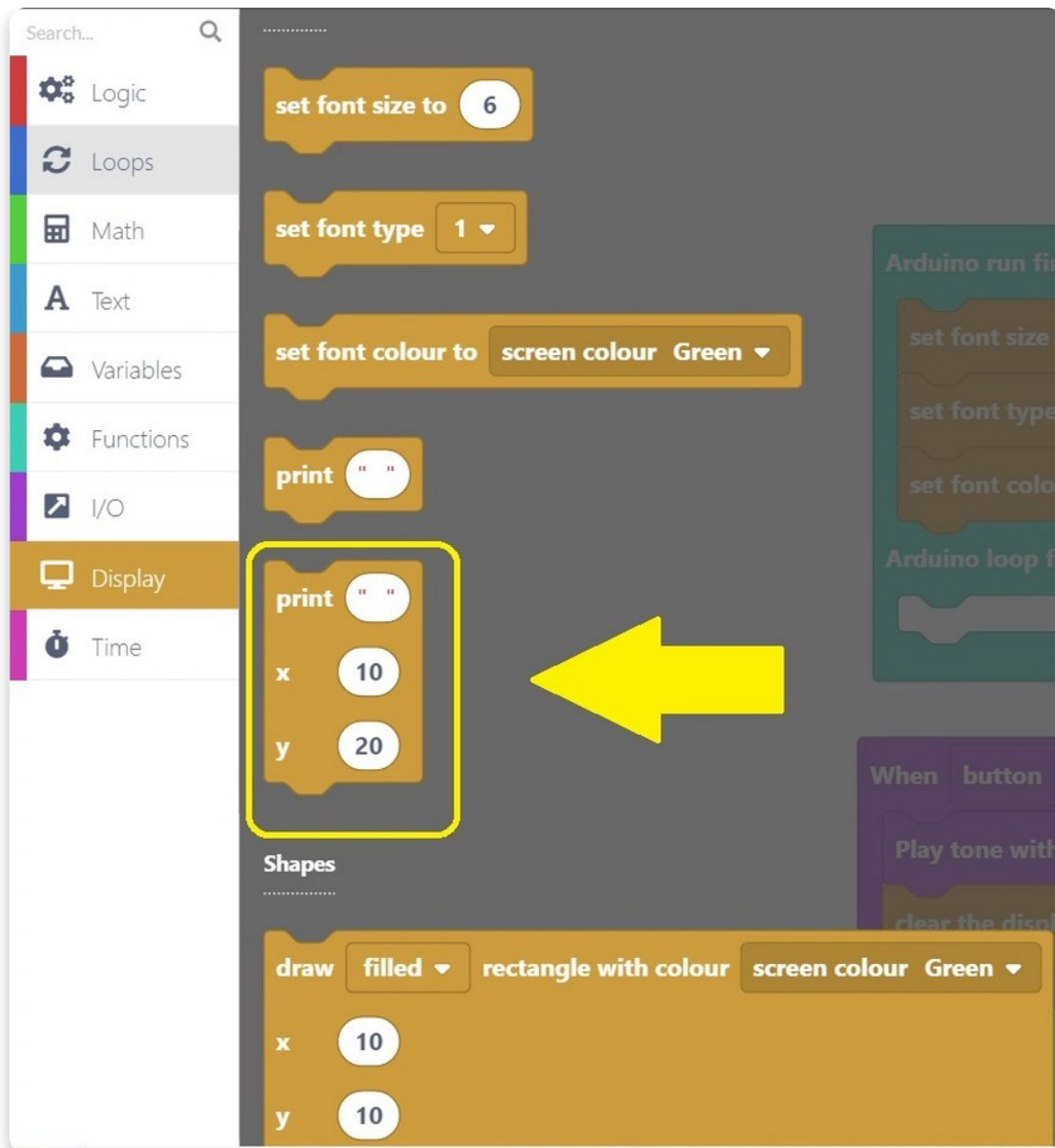
Play tone with



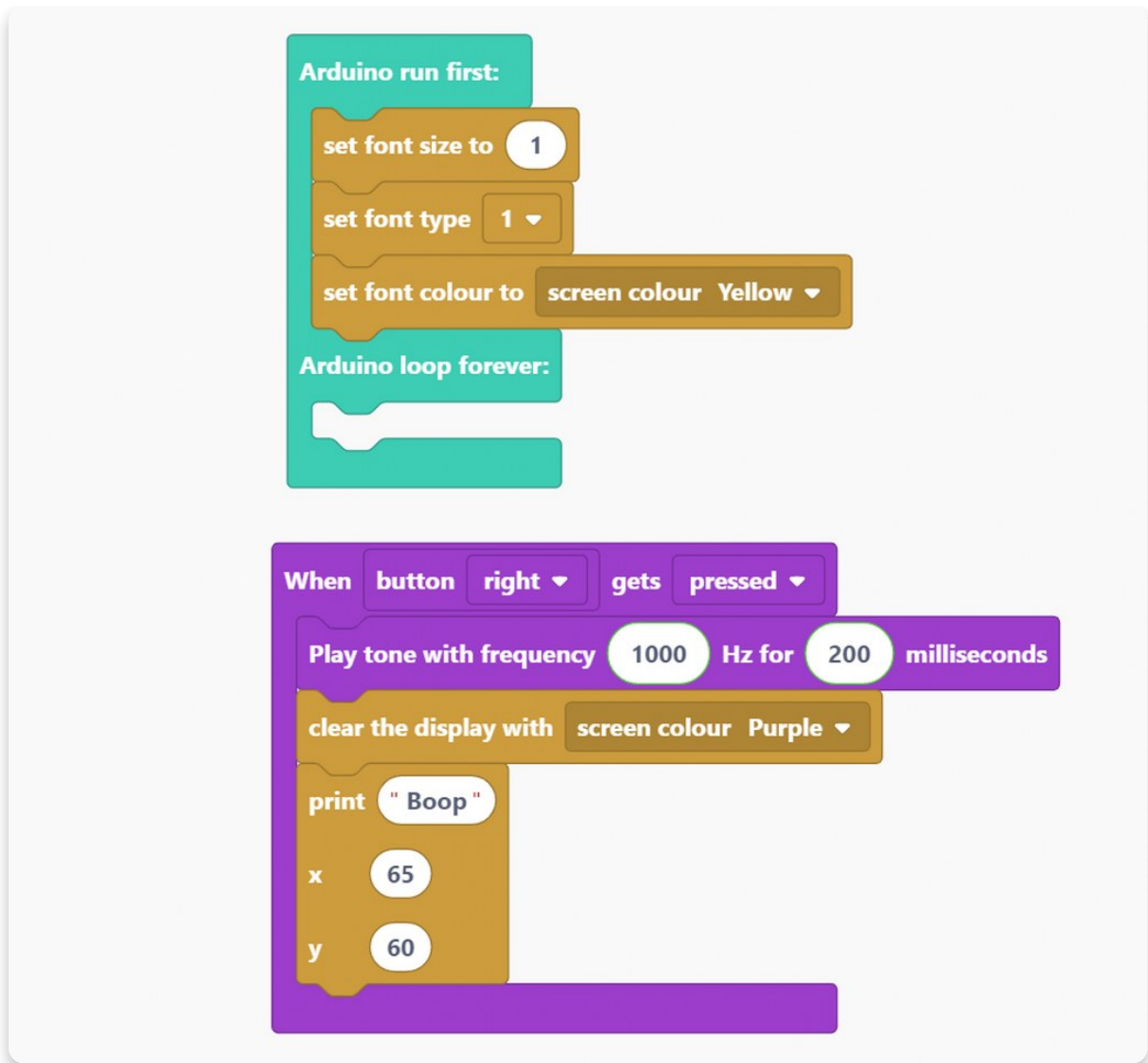




You can choose what will be written on the display once the button is pressed.



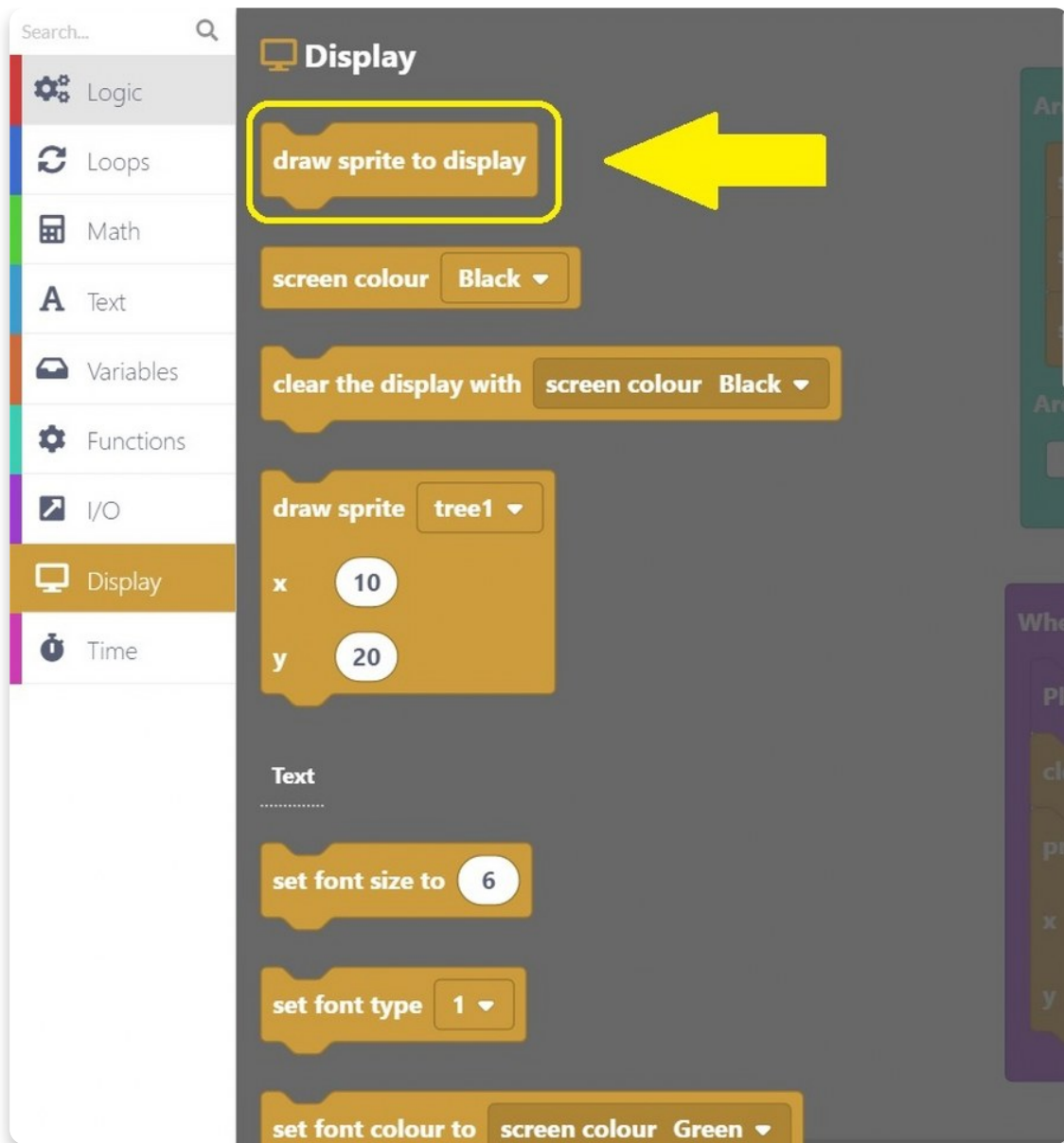
For example, the screen can say **"boop"** when we press the **right** button.



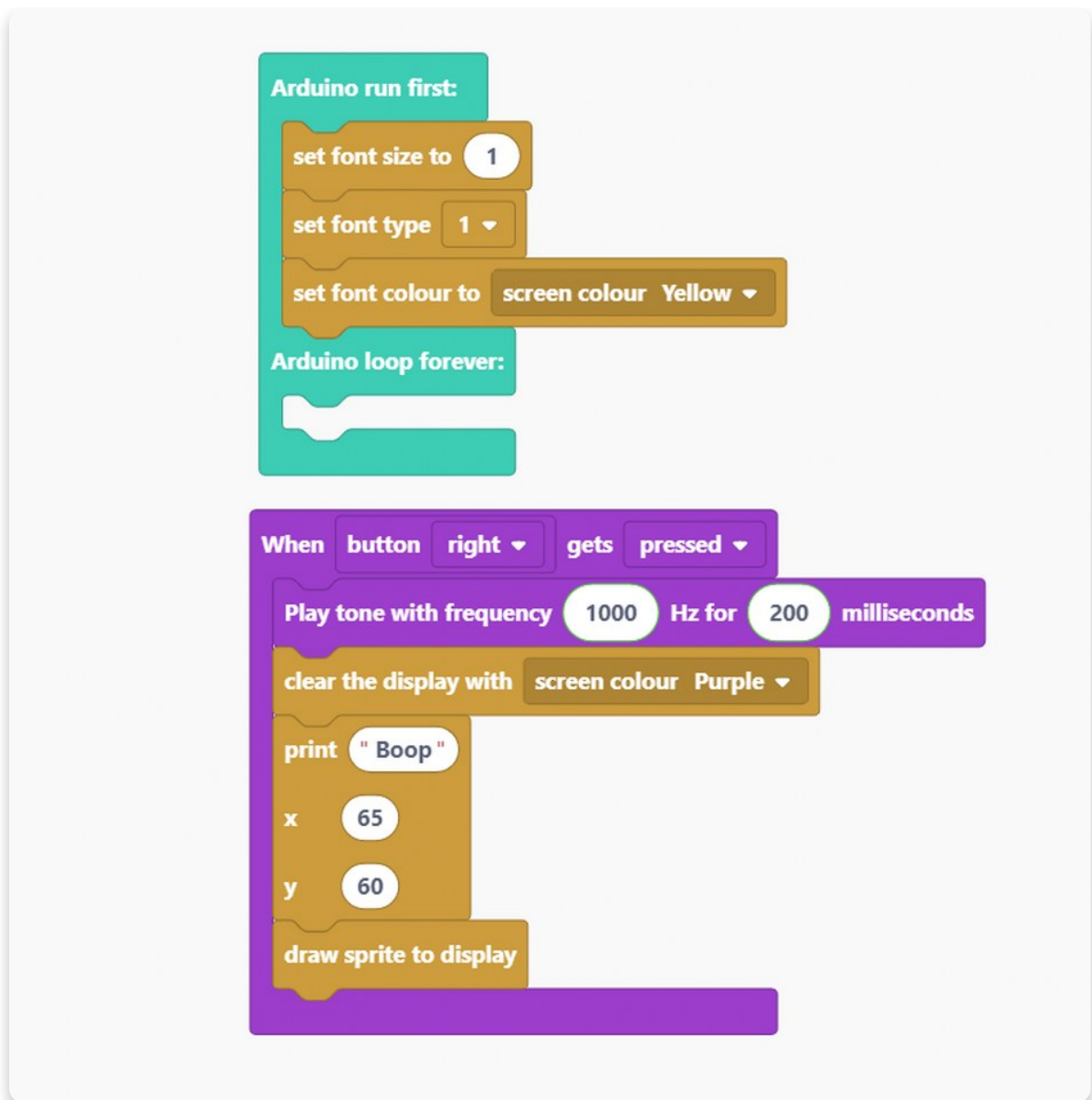
**Don't forget about the coordinates!**

Let's make x be 65 and y 60.

And for the end, drag the "**draw sprite to display**" block for a sketch to work.

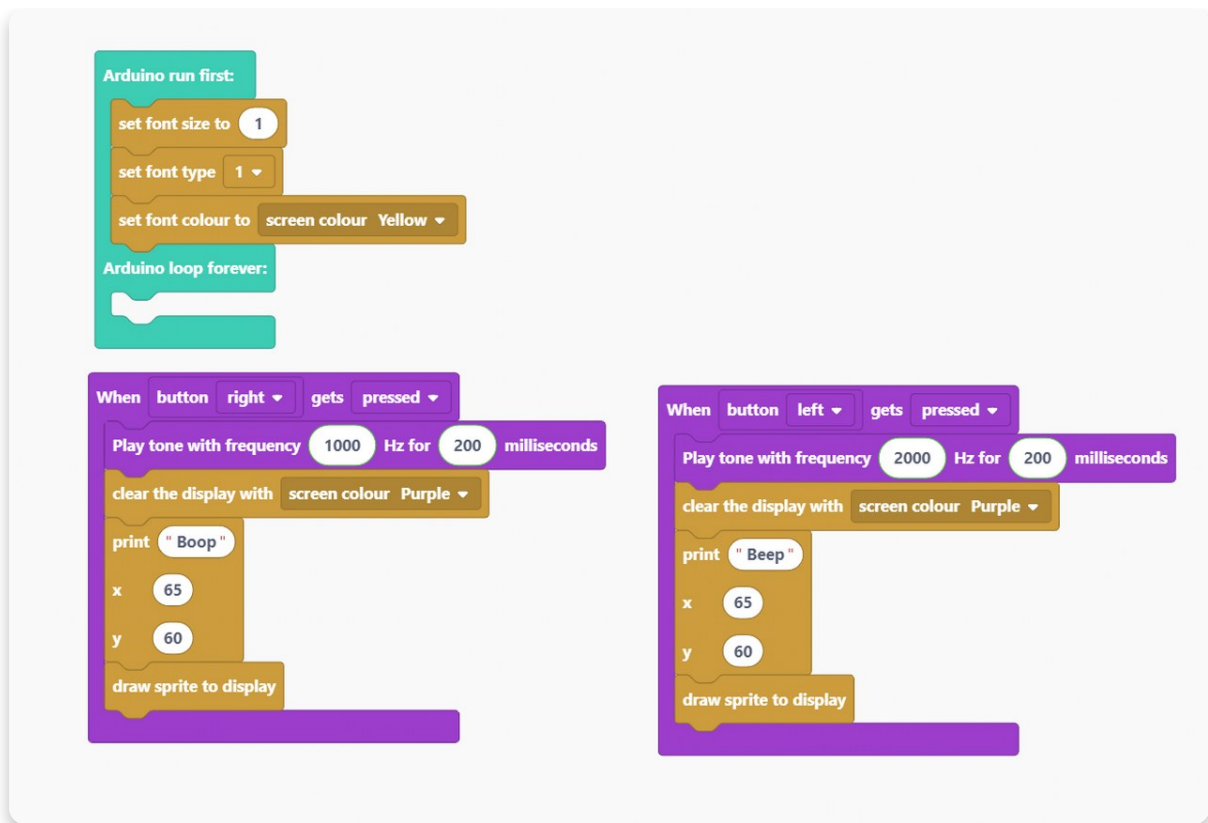


First button - DONE!



Since we'll need the same blocks, we can easily **duplicate** them.

Let's do one button at a time.



The first one we'll do is for when a left button gets pressed.

We decided to play tone with a **frequency** of **2000** Hz this time, but we kept the same amount of time we'll play that tone for.

The screen's color stays purple, but we'll print "**Beep**" this time.

**Let's duplicate the block again.**

Now, we'll use a button used for going back and playing tone with a frequency of **3000 Hz** for **50 milliseconds**. In the meantime, on the screen will be written "**Ding**".

Note that we kept coordinations the same for all prints.



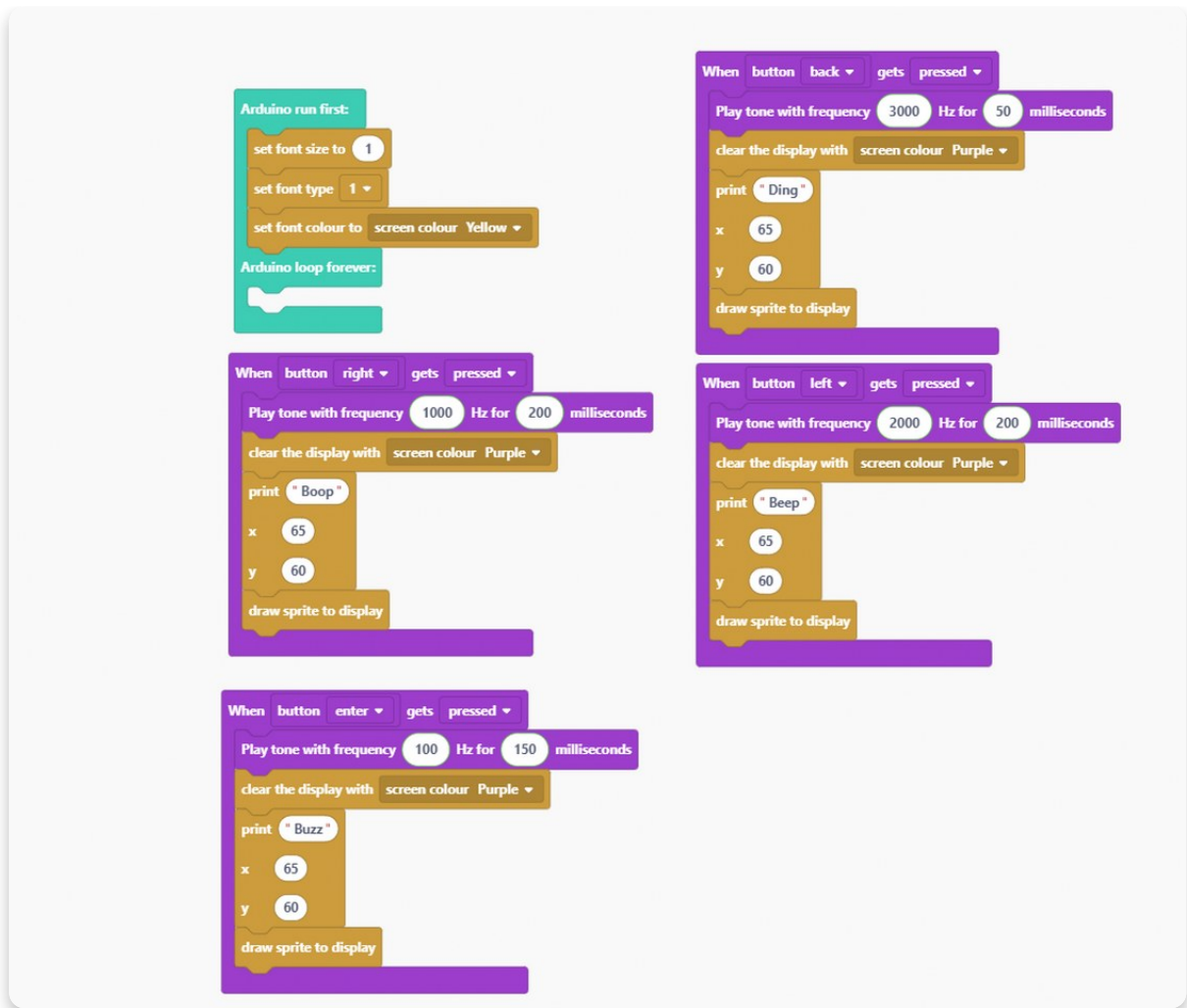
Yes, you guessed it right!

It's time to duplicate some more blocks.

The last button we'll use is the **enter** one.

In this sketch, we won't do anything with releasing this button.

If you press the enter button, the buzzer will play tone with a frequency of **100 Hz** for **150 milliseconds**. While the tone is playing, the screen will say "**Buzz**".



**Great! We used all of the four buttons from the first row on Chatter.**

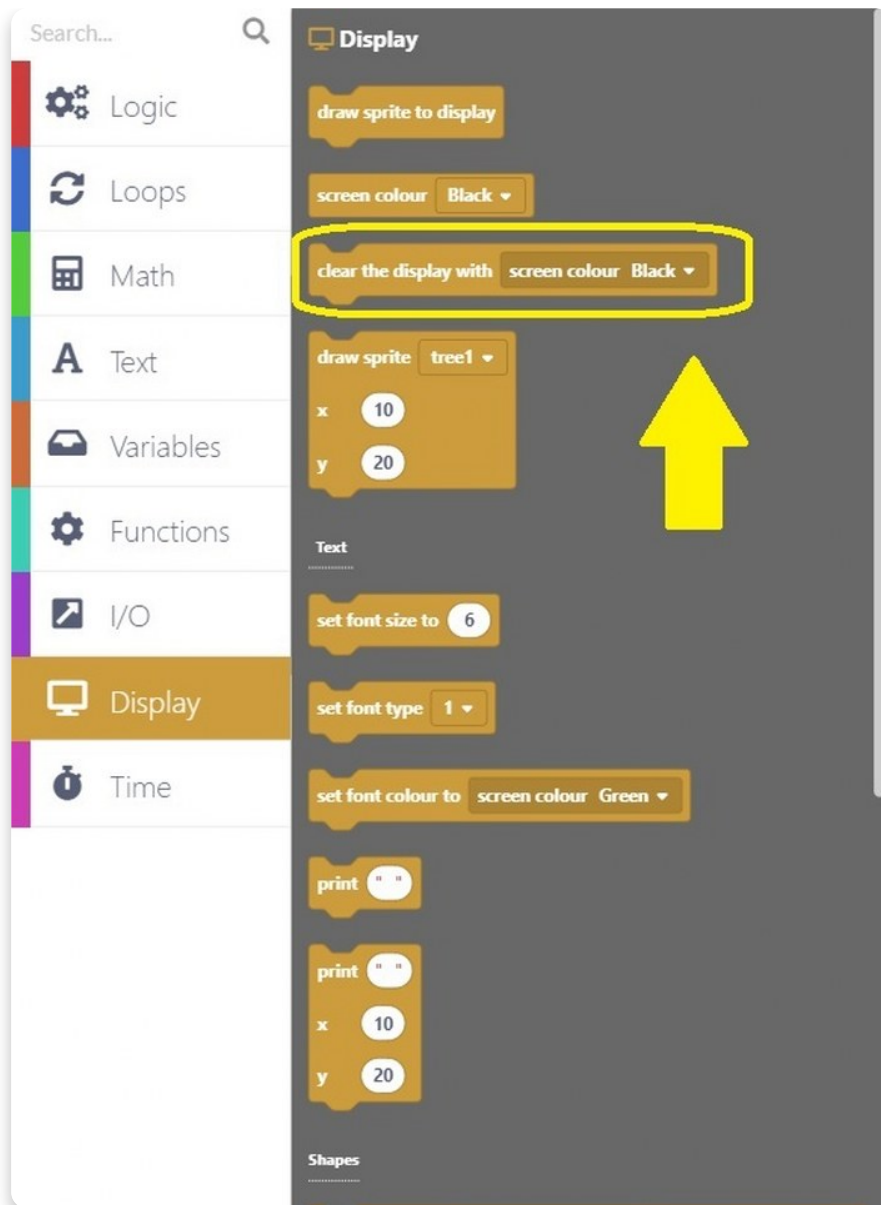
We should do something with the screen once Chatter turns on. The best thing we can do is write what will happen next on the screen.

To do so, we'll need to jump to the **Display section** and use a couple of blocks.

The first thing we want to do is clear the display so that the text can be transparent.

To do so, use this block:





This block needs to be dragged to the **Arduino run first** block section.

```
Arduino run first:  
set font size to 1  
set font type 1  
set font colour to screen colour Yellow  
clear the display with screen colour Purple  
Arduino loop forever:
```

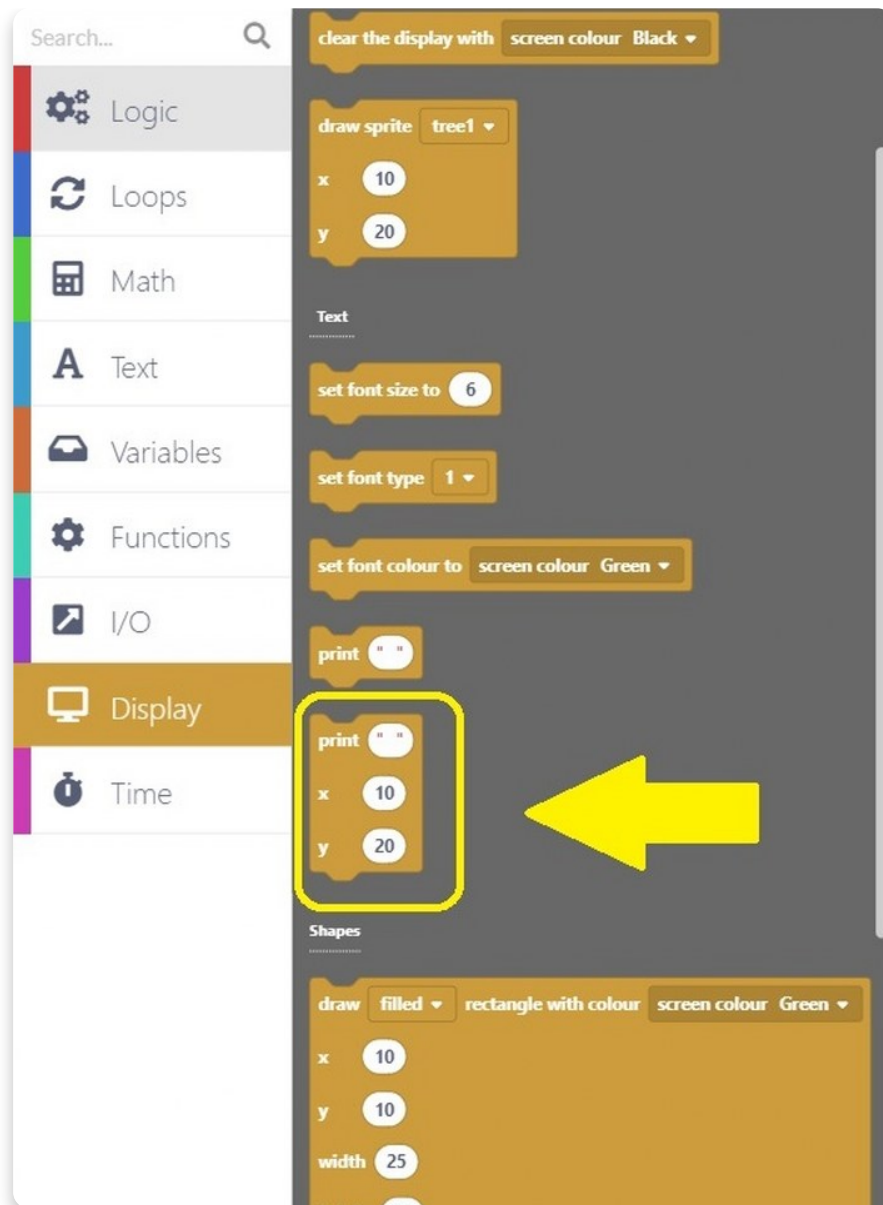
```
When button back gets pressed  
Play tone with frequency 3000 Hz for 50 milliseconds  
clear the display with screen colour Purple  
print "Ding"  
x 65  
y 60  
draw sprite to display
```

```
When button right gets pressed  
Play tone with frequency 1000 Hz for 200 milliseconds  
clear the display with screen colour Purple  
print "Boop"  
x 65  
y 60  
draw sprite to display
```

```
When button left gets pressed  
Play tone with frequency 2000 Hz for 200 milliseconds  
clear the display with screen colour Purple  
print "Beep"  
x 65  
y 60  
draw sprite to display
```

```
When button enter gets pressed  
Play tone with frequency 100 Hz for 150 milliseconds  
clear the display with screen colour Purple  
print "Buzz"  
x 65  
y 60  
draw sprite to display
```

Since we cleared the display, it's time to write something on it.



Drag a circled block in the **Arduino run first** section.

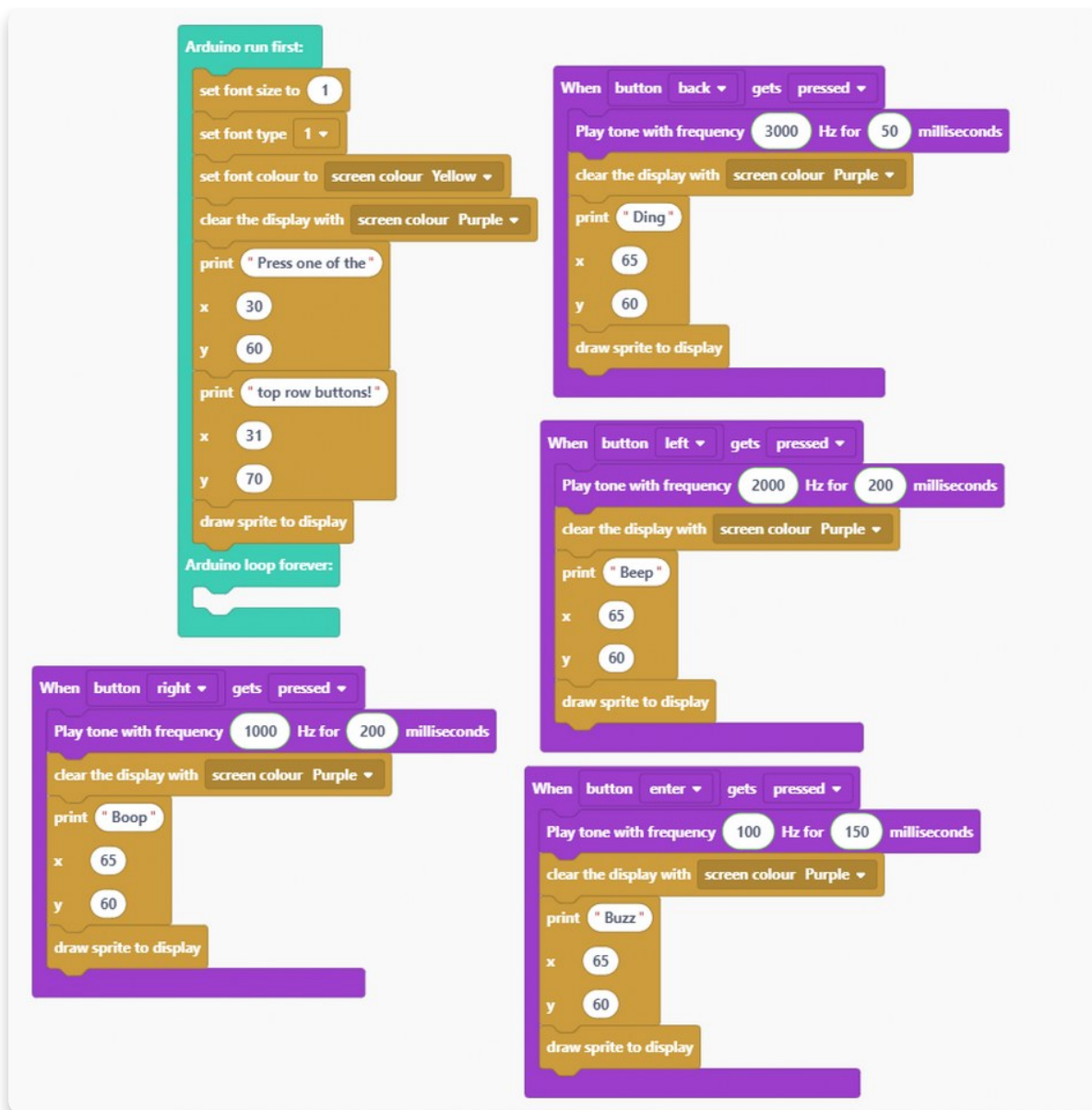
Also, you'll need two of these blocks, so the text is nicely placed in the middle of the screen.



So, the text that will be written on display once you turn your Chatter on will say, **"Press one of the top row buttons!"**. Basically, this is like an on-screen guide.

And, we hope you know what must be placed at the end of every sketch.

The **"draw sprite to display"** block!



**You did it!**  
**Congrats!**

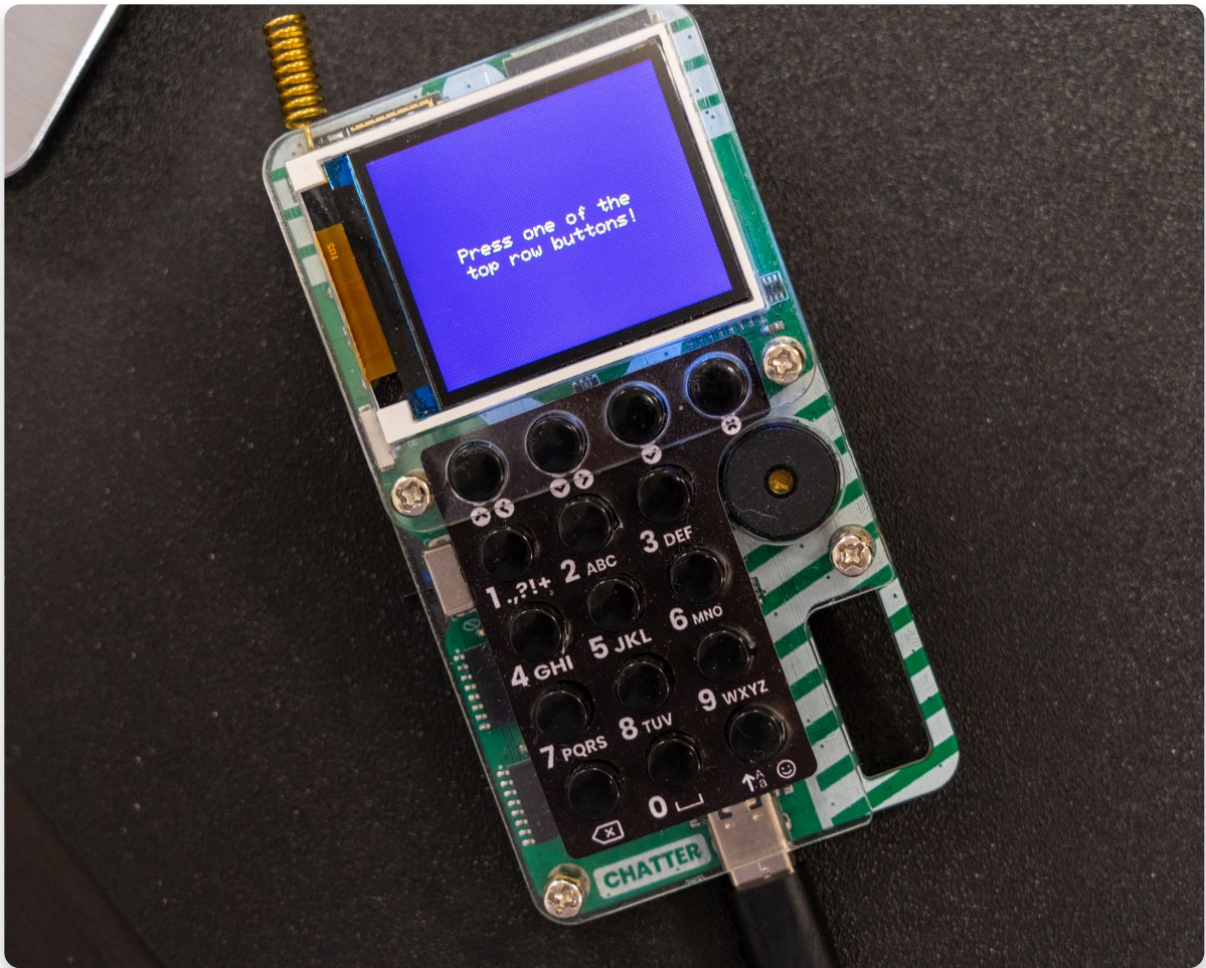
Hit the big red **Run button**, wait for a code to compile, and check it out.

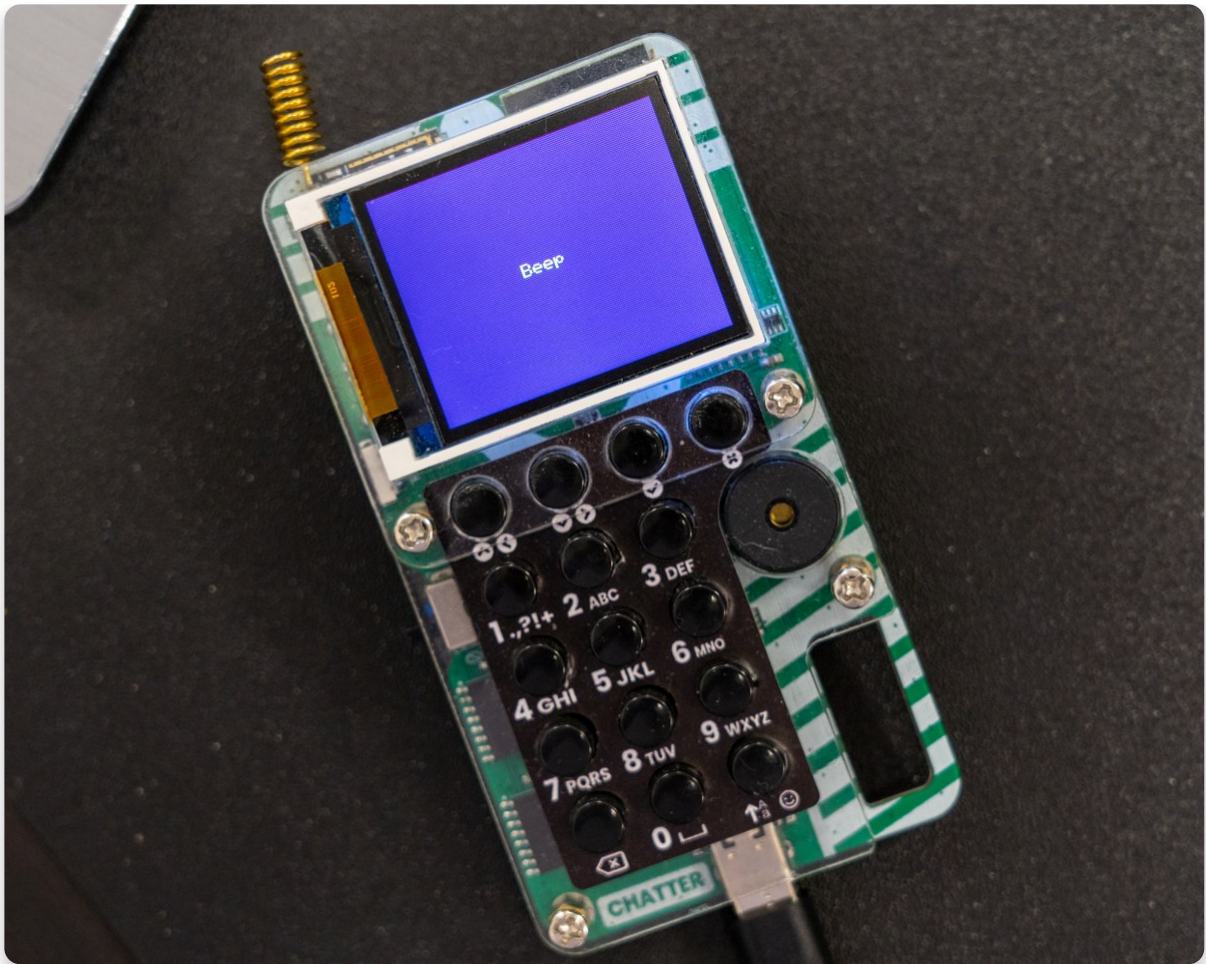
Once the code is compiled, your Chatter should restart, and the screen will turn yellow with the purple sign saying "Press one of the top row buttons!".

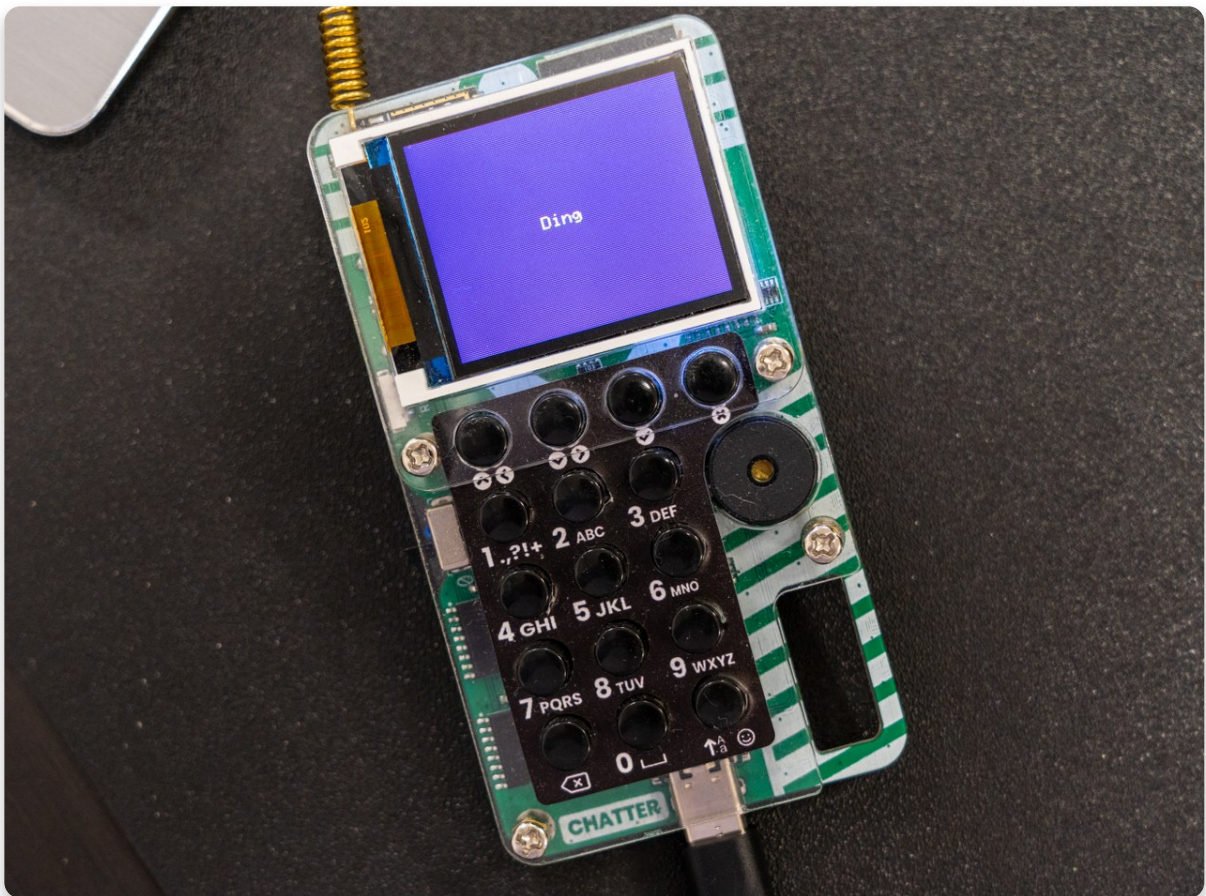
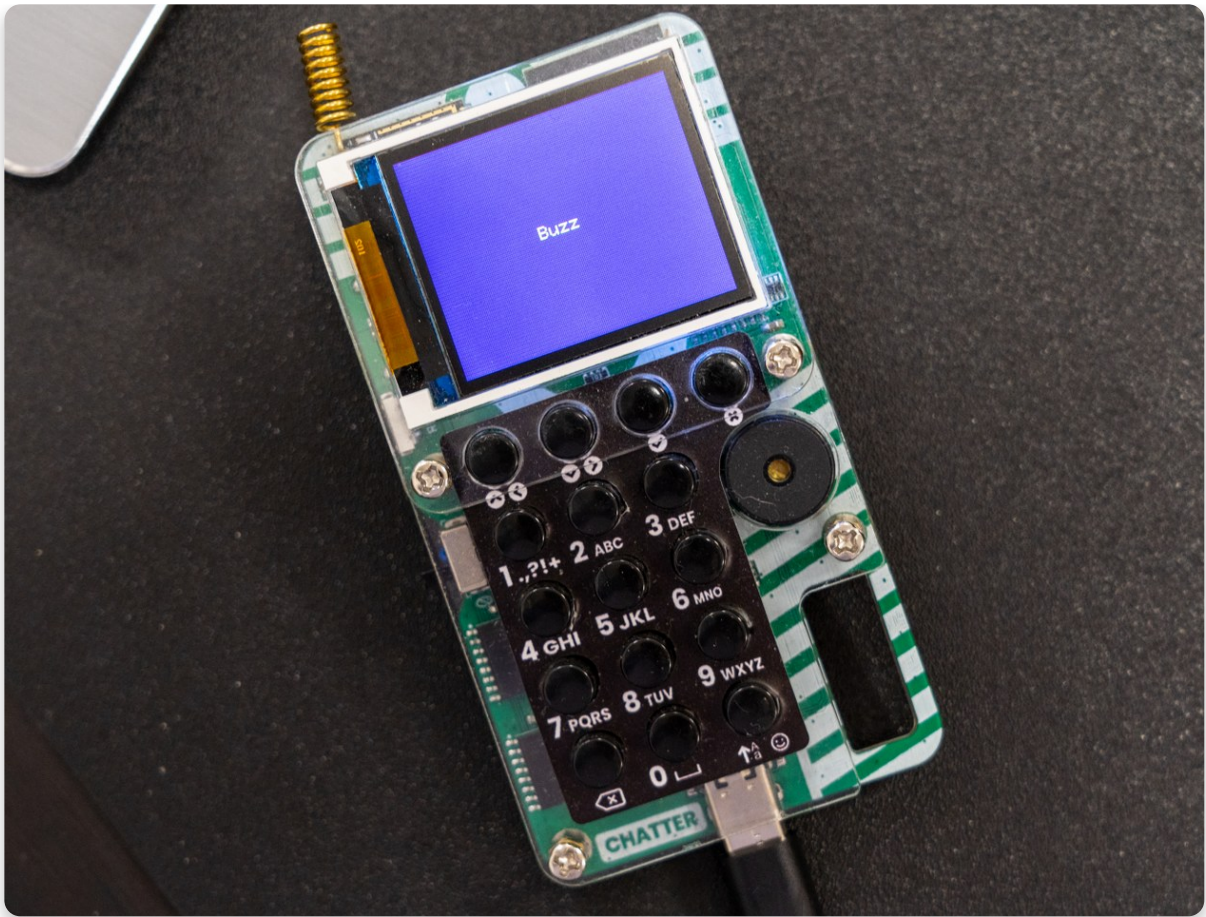
If you proceed with the command you got and click on one of the buttons, the sounds will start playing from the buzzer. Depending on the button you clicked on, the tone frequency will be different, and the duration of the tone will change.

Also, each button will trigger another text written over the display.

You can check what the screens should look like in the photos below:









# Draw images

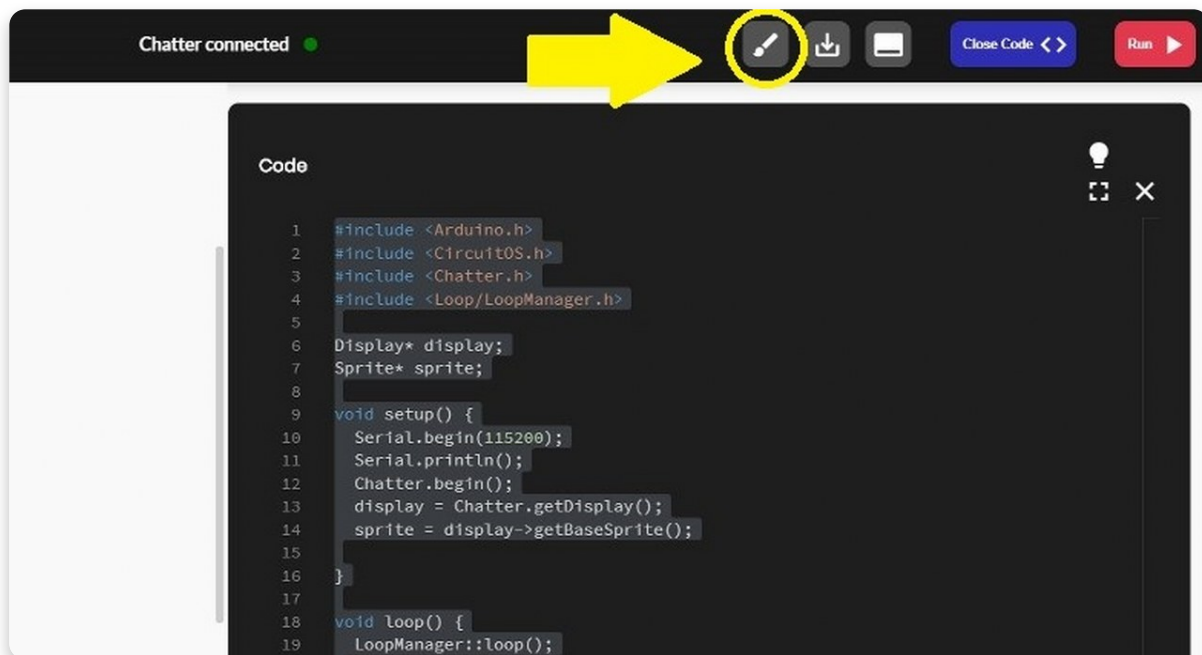
At the beginning of this guide, we mentioned a cool little tool called **Sprite editor** (icon: brush).

Now, we'll learn how to use it.

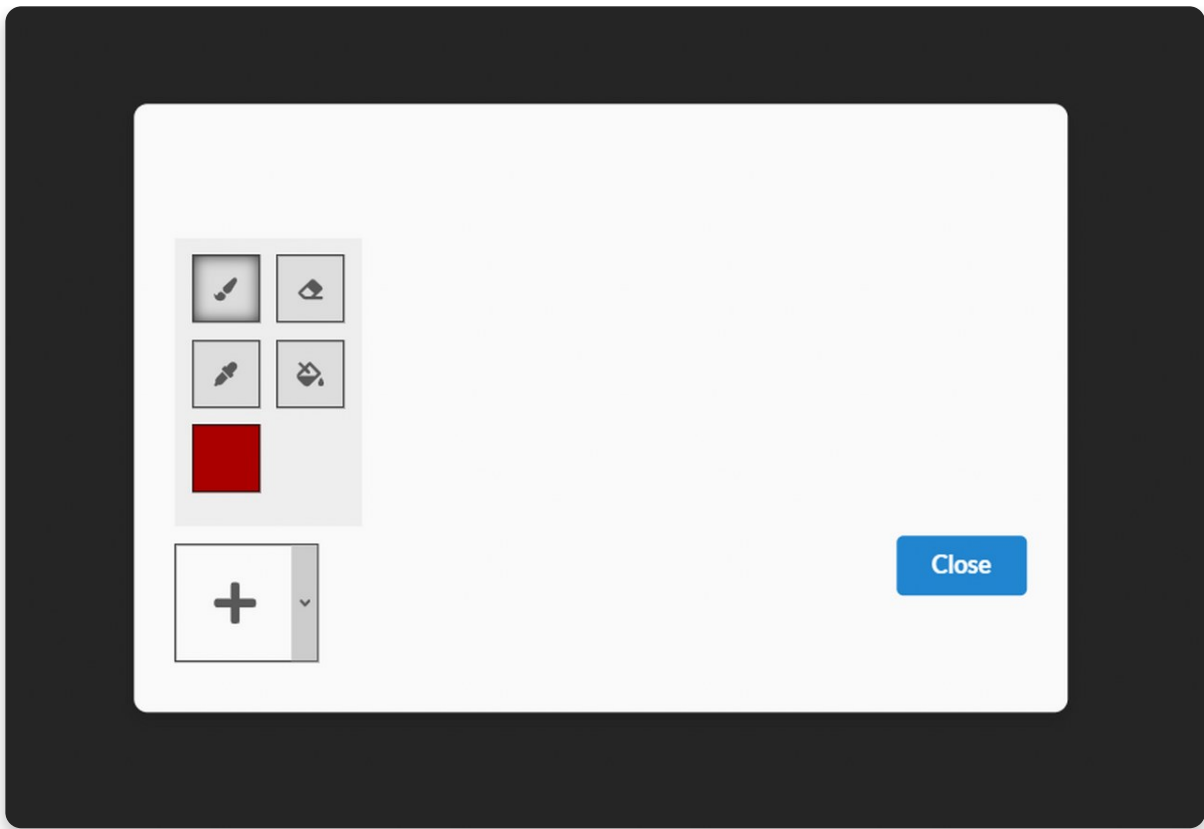
If you have ByteBoi, nothing's changed. You can draw and display the images on the screen the same way. But, if you're new here, here comes drawing lessons.

In computer graphics, a **sprite** is a two-dimensional image or animation that is integrated into a larger scene.

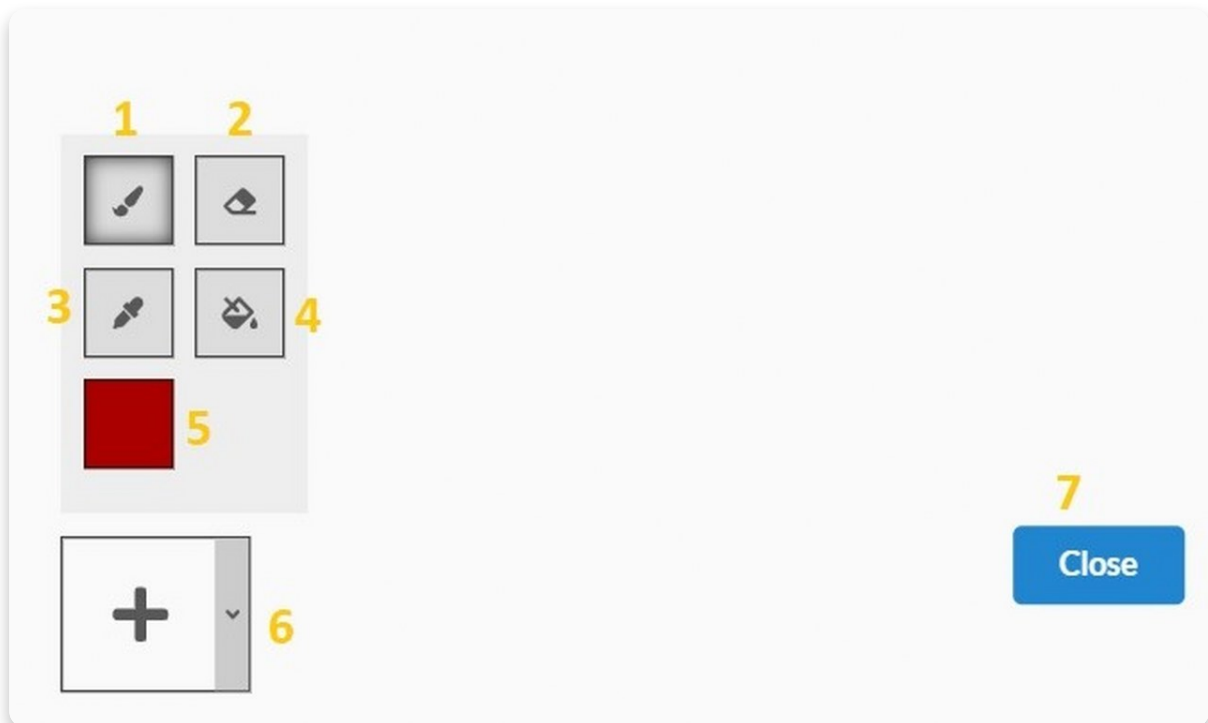
Find a **brush icon** on your **Toolbar** first.



A window will open, as shown in the photo below:



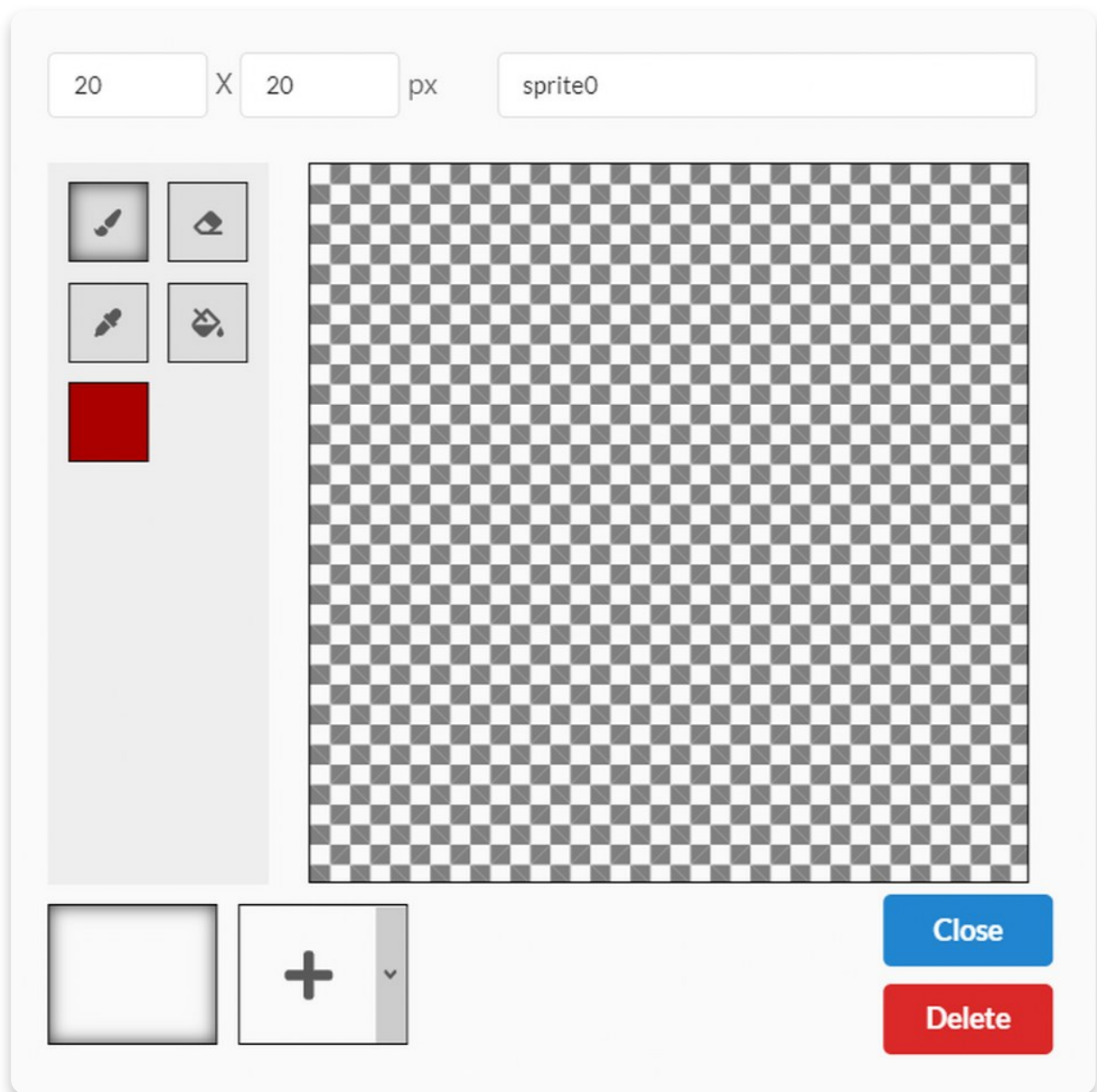
Let's see what each icon means.



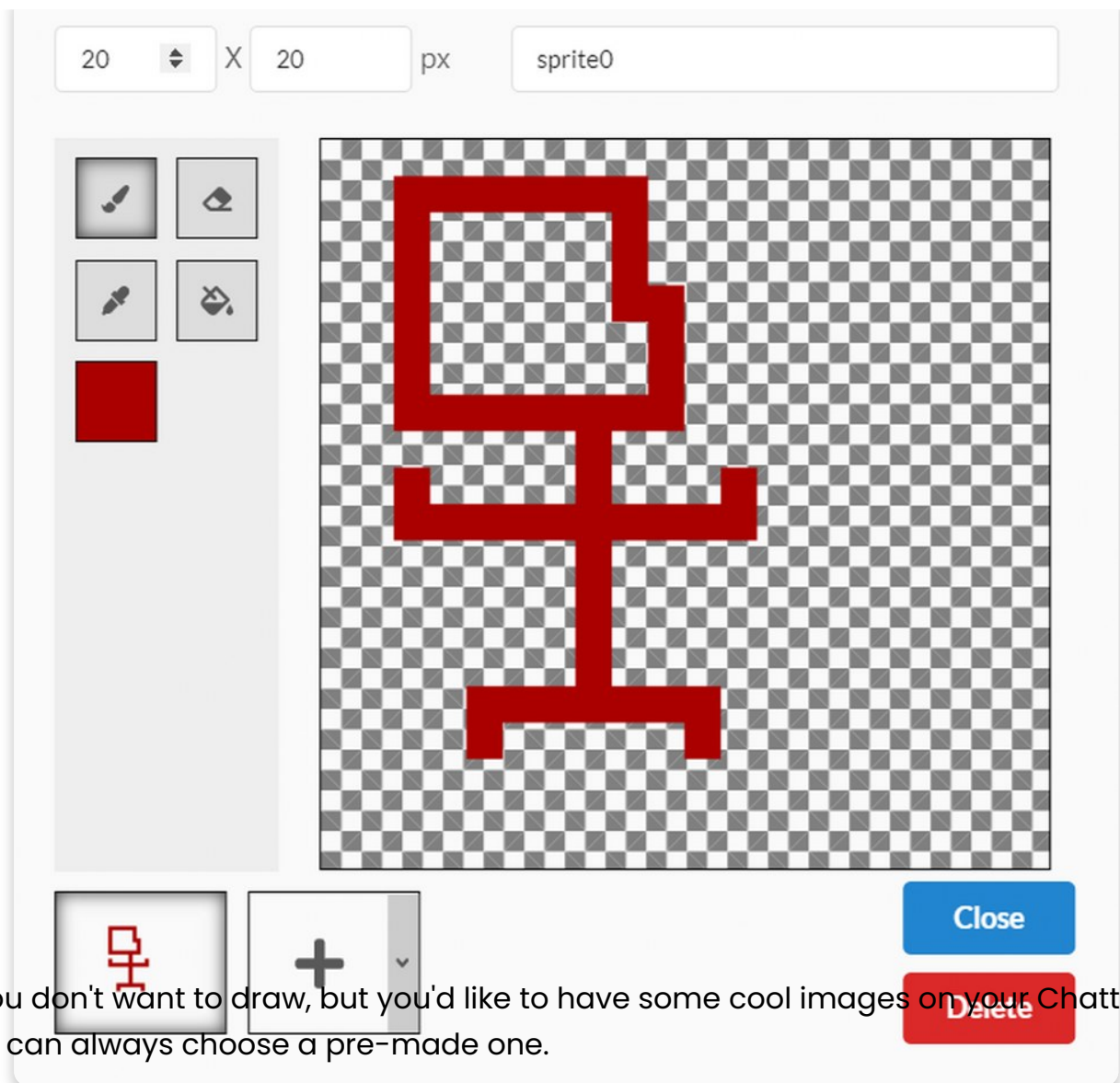
1. **Paintbrush** - you'll use it for painting whatever you want.
2. **Eraser** - you'll use it for erasing mistakes you make.
3. **Color picker** - pick a color from the picture and draw.
4. **Fill in with color** - fill the image or part of an image with a different color.

5. **Color palette** - here, you can choose which color you will use for drawing.
6. **New picture** - if you click on a plus (+) sign, you will get a blank window for your new picture. But if you press an arrow pointing down, you will see all the pre-made images.
7. **Close** - once you're done with the drawing, press the big blue button saying "Close".

This is the blank window you'll see at the beginning of the drawing.

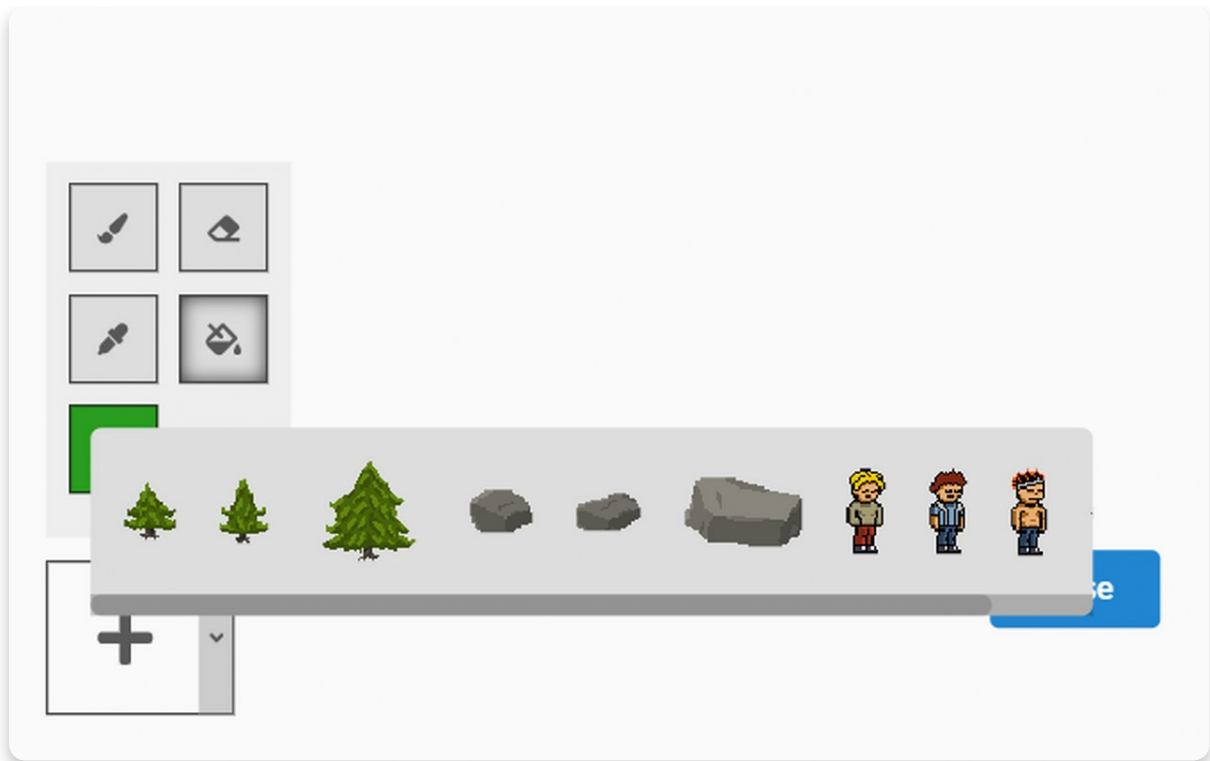


You can paint whatever you want, and later on, you can put it on your Chatter's display.



If you don't want to draw, but you'd like to have some cool images on your Chatter, you can always choose a pre-made one.

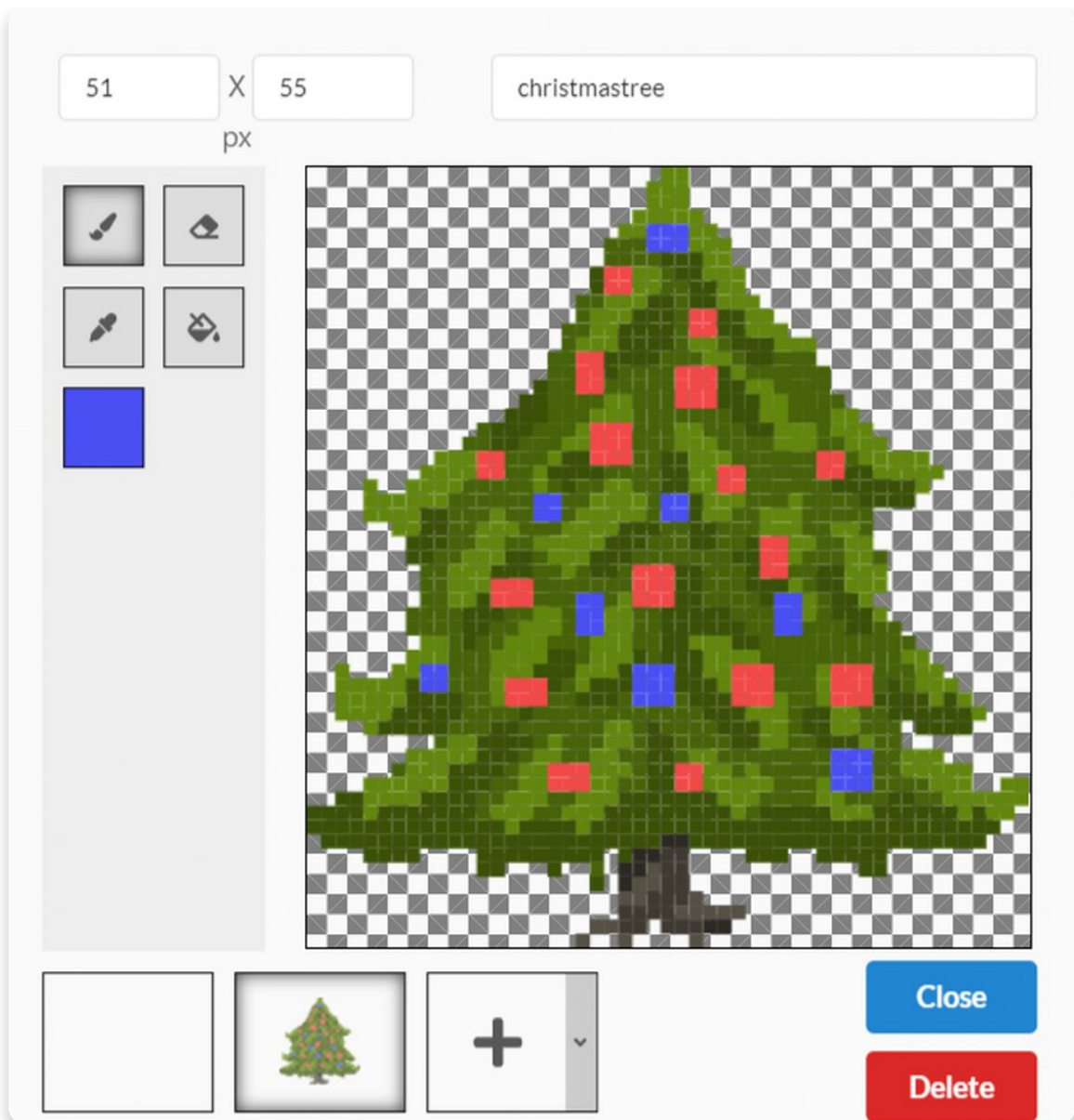
Maybe you want to draw on the pre-made icons and make them better. No problem, just click on the one you want to upgrade, and start drawing!



For example, I chose to draw a Christmas tree but didn't want to draw a tree from scratch. So, I took a premade tree and drew Christmas ornaments.

Also, I didn't want my tree to have a generic name, so I renamed it to christmastree.

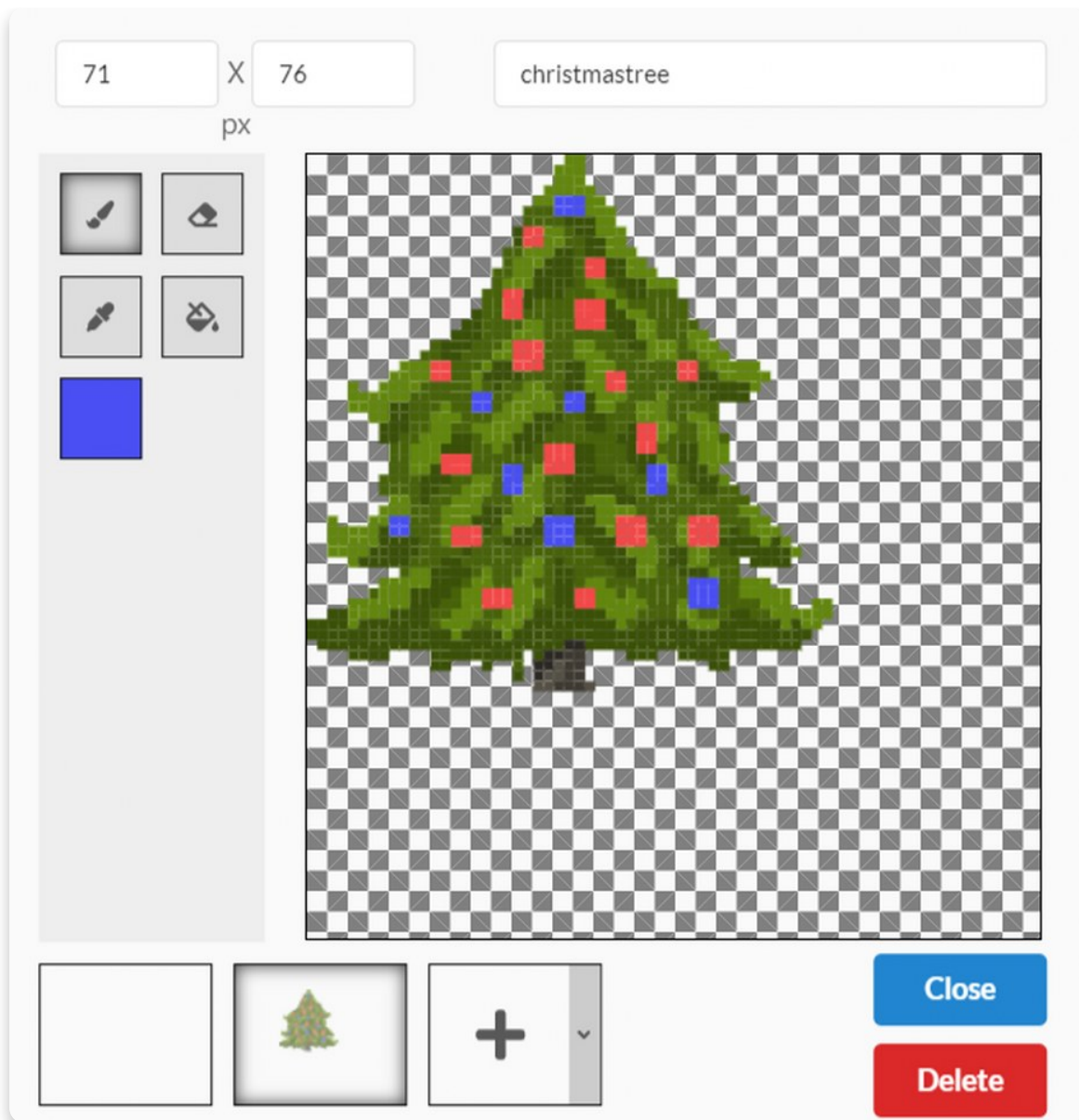
If you want to rename your drawing, you should know that its name cannot begin with a number, and there can't be any spaces between words.



One more tip is that you can change your sprite's resolution in the upper left corner.

**The resolution** is the size of a sprite, and it's measured in pixels.

**A pixel** is the smallest controllable element of a picture represented on the screen.



Great job!

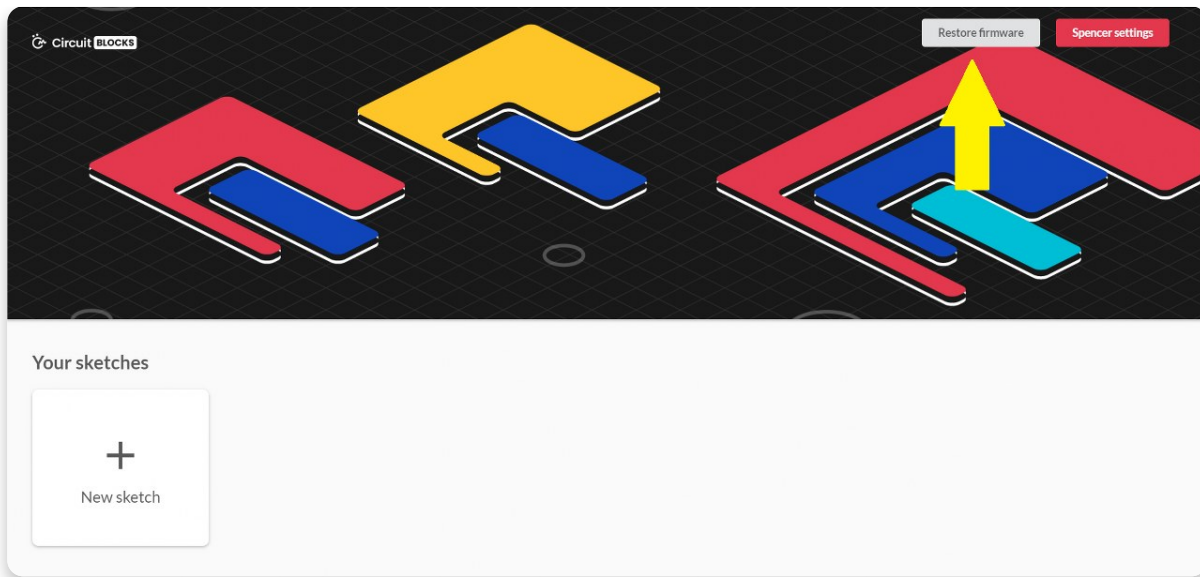
You can try drawing from scratch and making your icons.

## Restore Chatter's base firmware

# Restore Chatter's firmware

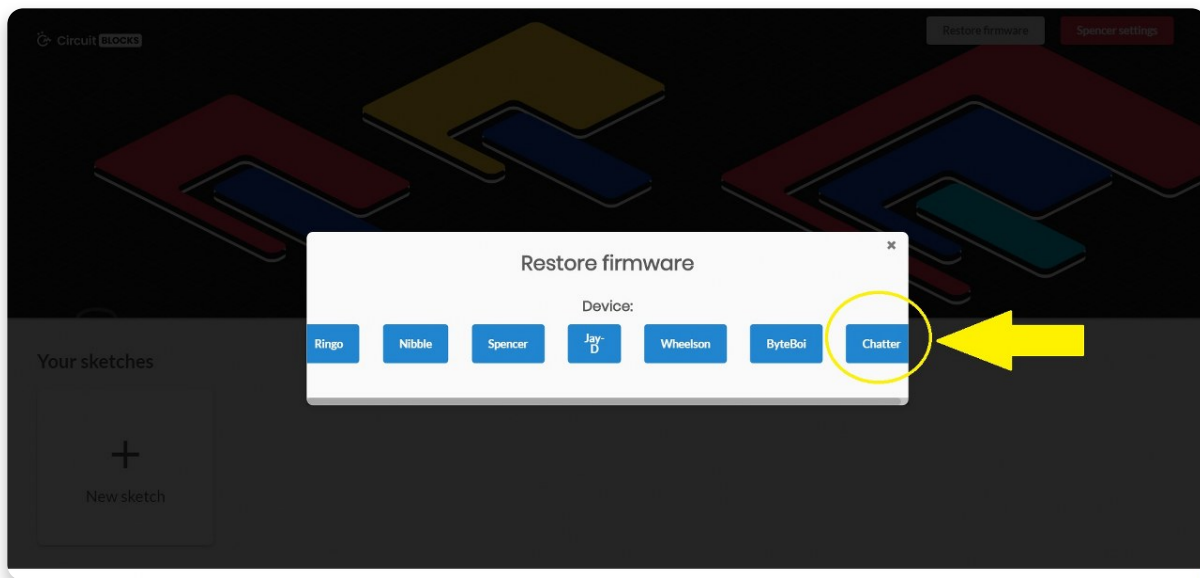
Once you're done coding and just want your Chatter to be "normal" again, you need to restore his base firmware.

This is quite simple, just connect your Chatter to the USB port of your computer and press the "Restore firmware" button on the top right.



You will be prompted with a window to choose the device you are restoring the firmware for.

Choose Chatter, of course.



Wait for a few seconds, and your Chatter will be back and running like usual.

You need to do this whenever you're done coding your Chatter if you want him to revert to his initial out-of-the-box functionality.

## What's next?

You've reached the end of our first Chatter coding tutorial, congratulations!



I hope you're as excited as we are about Chatter's future since there are so many cool things we want to do with it in the future firmware and CircuitBlocks updates.

In the meantime, continue exploring on your own and show us what you've done with your Chatter by sharing it on the [CircuitMess community forum](#) or via our [Discord channel](#).

If you need any help with your device, as always, reach out to us via [contact@circuitmess.com](mailto:contact@circuitmess.com), and we'll help as soon as we can.

Thank you, and keep making!