Creative Code Lessons For Ikamva Youth

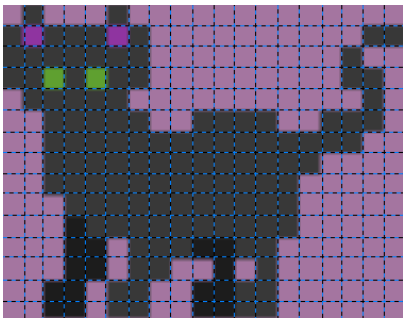
By Marion Walton & Lyndon Daniels

Based on Marion Walton's Training Resources at
https://ikamvacodes.wordpress.com/creative-code/
CREATIVE CODE is a Dr Marion Walton initiative

Shapes and Coordinates

Pixels

Whether the screen you are using is on a mobile phone, a desktop computer, TV or any other digital device that screen is made up of pixels. Pixels are tiny dots that cover the screen, and store color information. As a result when many pixels are placed next to each other they can appear to form an image.



Every digital image is made up of pixels. They appear here as the blocks making up this image of a cat. This image is 19 pixels wide and 15 pixels high. As you can see it's not a very detailed picture of a cat, that is because there are not enough pixels to make up the missing details in the image. We call this a **Low Resolution Image**.

A more detailed picture of a cat would be made up of many more pixels, if the image was a digital photograph it would be a **high resolution image** consisting of thousands of pixels.

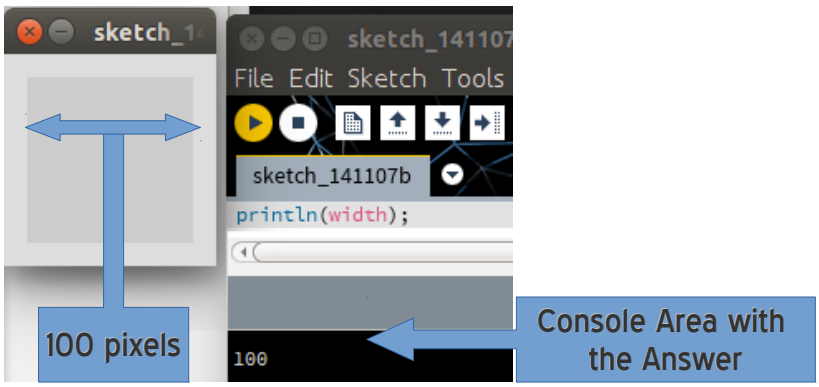
When we create code we need to be aware of how many pixels are available to us. Knowing this information will help us to draw shapes and other objects to a screen.

Processing makes it really easy for us to find out how many pixels we can use in our program by providing us with two special keywords width and height.

For example if we wanted to know how many pixels made up the width of our app we could ask Processing with a command such as the following,

```
println(width);
```

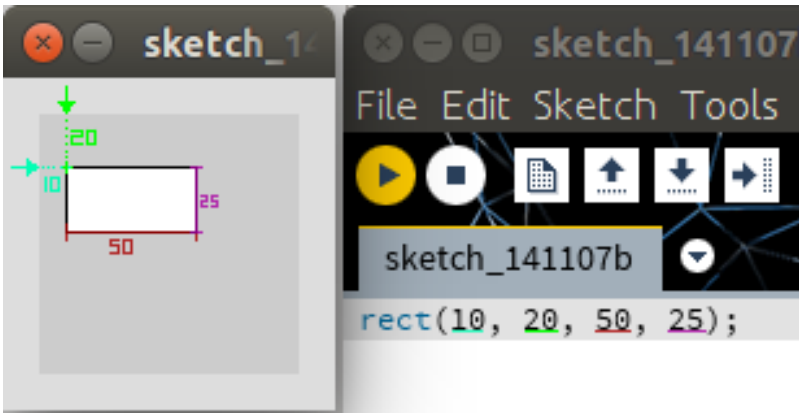
You can copy and paste this code into an empty processing document. Then hit the Run button to get the answer.
You'll notice that the answer appears in the bottom of the Processing window, we call the Console Area.



When we know how many pixels are available to us, we can use this information to set coordinates for drawing shapes with code.

For example if we want to draw a rectangle in Processing we would need to first tell Processing where to place the rectangle, then how wide and tall the rectangle is. For example the following code will draw a rectangle,

```
rect(10, 20, 50, 25);
```



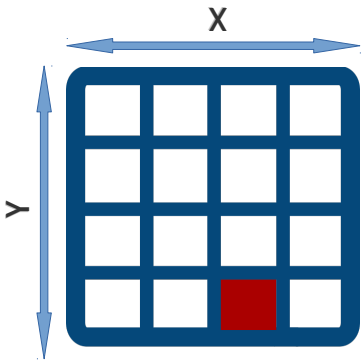
To draw a rectangle we use the keyword rect(). This type of command is known as a **function**. We can control the rectangle function with four numbers that we call arguments.

- The first argument "10" is how many pixels the rectangle is from the left edge of the app screen.
- The second argument "20" is how far the rectangle is from the top of the app screen.
- The third argument "50" is how wide the rectangle is.
- The fourth argument "25" is how high the rectangle is.

As you can see arguments are really important for modifying a function's behavior.

Coordinates

As we already know the pixels that make up our screen are placed next to each other, the placement of these pixels therefore forms a grid.



In this image of a grid we have a screen that is made up of 16 pixels. We call this a 4 x 4 (four by four) grid, because it is 4 pixels wide and 4 pixels high.

In programming we call the width of a grid the X axis and we call the height of a grid the Y axis, therefore the coordinates of the red pixel is (3, 4) that is 3 X and 4 Y.