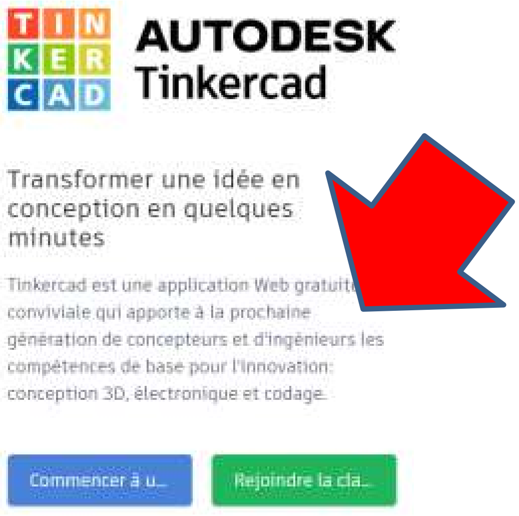
Lab4Schools

Lab Activity “TinkerCad”

Co-funded by the Erasmus+ Programme of the European Union

1- LAUNCH OF TINKERCAD

Visit the Tinkercad website | Create 3D digital designs with online CAD | Tinkercad
On the home screen, click on "join class".


AUTODESK Tinkercad
Transformer une idée en conception en quelques minutes
Tinkercad est une application Web gratuite conviviale qui apporte à la prochaine génération de concepteurs et d'ingénieurs les compétences de base pour l'innovation: conception 3D, électronique et codage.
Commencer à u...
Rejoindre la cla...

Note: I can see and edit all the circuits in my session.

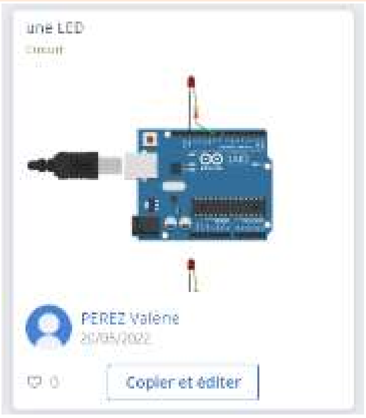
Enter the code and your nickname provided by the teacher.
Code = DKH7TN48DAA3
Or click on the link and enter your User ID.
https://www.tinkercad.com/joinclass/DKH7TN48DAA3

In the proposed menu, click on "my classes" then on your **2TSCRSA class**.
All you have to do is follow the proposed activities.

2- LED PROGRAMMING

Select the activity "I discover programming".

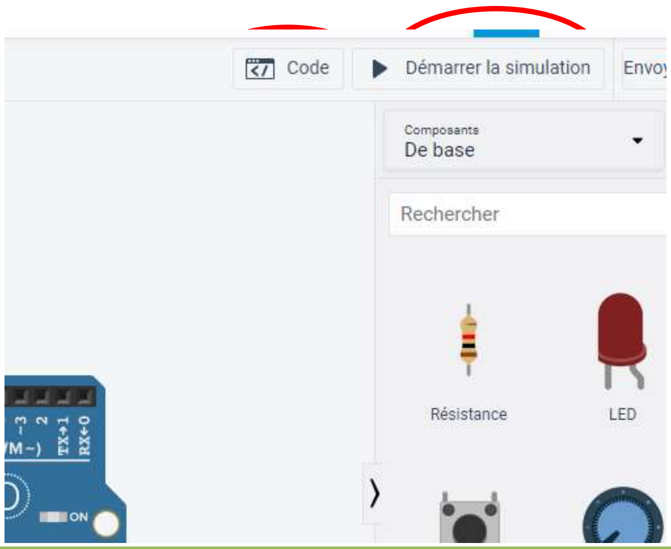
Exercise n°1: understanding the 1 Led program



Select **copy and edit** the circuit "one LED".
Your circuit appears on the screen.

Starting the simulation, what do you observe as functioning?

Complete the Answers document

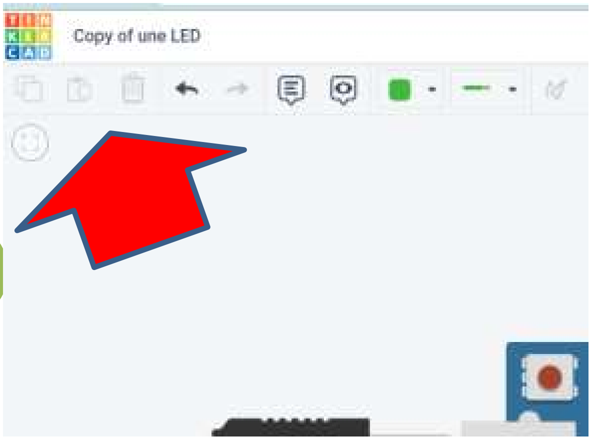


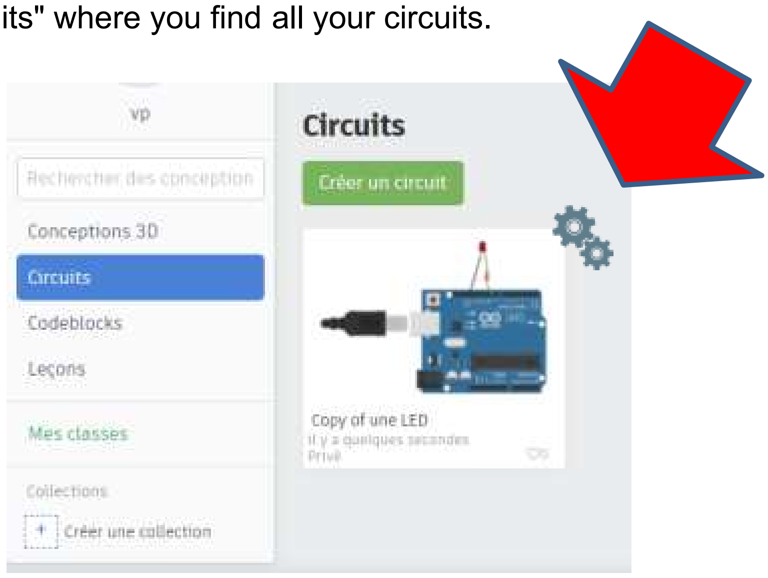
Open the window containing the code.

What is the name of the only variable used?
Its type? its value? why?
What do pinMode, digitalWrite and delay do?

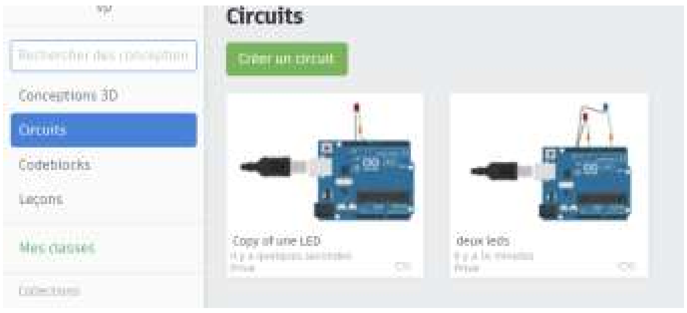
Exercise n°2: programming of 2 LEDs

Click on the dashboard and then on "circuits" where you find all your circuits.

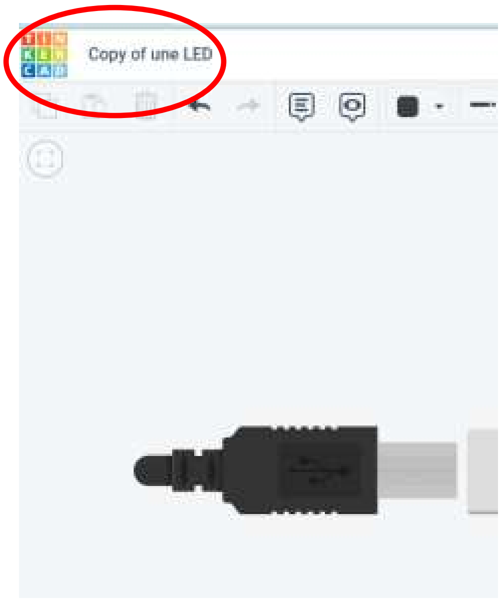




Duplicate the circuit "one led". You now have two circuits available whose name you will change the name of the new circuit appeared.



Edit this copy and modify the name of the circuit.
Call the "two leds".



In this circuit, add an identical 2nd blue LED on a new pin. Complete the program in order to turn on the second LED with the following constraints: when the 1st is on, the 2nd is off and vice versa.
Test the new program.

Complete the Answers document

Have your teacher validate the function.
Download the code for further experimentation. You can also copy the code into a WORD document.

3- Programming a ribbon of LEDs

Select the activity "Iprogram a ribbon of LEDs".
See the appendices: NeoPixels, the Arduino library

Exercise n°3: understanding the program

Select **copy and edit** the circuit "ribbon of random leds".
Starting the simulation, what do you observe as functioning?
In the program, highlight in red the names of the variables chosen by the program developer, which are then used in the body of the main program and which could be changed by us.
Complete the Answers document

What needs to be changed so that the LEDs are lit faster? Slow down?

Duplicate this circuit, change its name and then modify it.
Make the changes and test the new program.

Have your teacher validate the function**.**

What does RBG mean?
If we set the 3 colors to 255, what color do we get?
If you want to light only in YELLOW, how should you do it?
Make the changes and test the new program.

Have your teacher validate the function**.**

Modify the program in order to observe the lighting of one led after the other, i.e.:

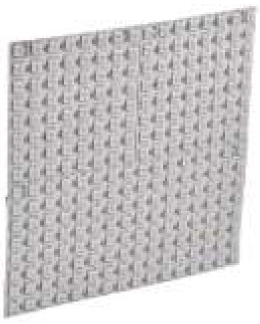
- Only 1 is lit each time
- The previous one is extinguished
- Ignition in order

How do I turn off a LED ?
What is the command to turn off all the LEDs at the same time?
How is the **for()** loop used in the code defined? What is it used for?

Modify the program and test it.
Have your teacher validate the function**.**
Download the code for further experimentation. You can also copy the code into a WORD document.

Annexe : LED NeoPixel

The NeoPixel are ultra-bright intelligent LEDS RV B . They can be combined to form bands, rectangles etc.



For example, this small module in the form of a stick, is equipped with 8 NeoPixels.
+

Each NeoPixel has inside:

- an LED RVB
- an integrated electronic circuit, the [WS2812B intelligent control LED integrated light source]
- a Data in paw and a Data out leg.

The NeoPixel are chainable between them, that is to say that the incoming data of one will be the outgoing data of the next and so on.
This information is transmitted very quickly from one NeoPixel to another and each NeoPixel takes the information about it.
The 1st NeoPixel will have the address 0, the next 1 etc.....
We are talking about addressable LEDs.
The data comes from a single pin of a microcontroller such as Microbit, ESP or Arduino.

The microcontroller must send information regarding:

- **brightness from 0 to 255 for each channel in Red Green Blue,**
- **the address of the NeoPixel concerned.**

Appendix: ARDUINO Library – Adafruit.NeoPixel.h

Controlling NeoPixels "from scratch" is quite a challenge. AdaFruit offers a library/bookstore allowing above all to have fun and focus on interesting points. The library works with most Arduino boards and derivatives: Uno, Mega, Leonardo, Micro, Adafruit Flora, etc.

Now let's take a look at the code...

1- All programs using a NeoPixel ribbon start with the inclusion of the specific (declaration) library. This library contains the commands necessary to use and order a ribbon of NeoPixels type LEDs.

```
#include <Adafruit_NeoPixel.h>
```

2- The following code block is used to declare our LED ribbon:

```
#define PIN 6
Adafruit_NeoPixel ruban1 = Adafruit_NeoPixel(60, PIN, NEO_GRB + NEO_KHZ800);
```

The first line assigns a number to the constant called "PIN" to use as a reference later. Here the ribbon is connected to pin 6.
It's not absolutely necessary to do it like this, but it does make changing code easier if you want to use a different control pin... otherwise, the entire code would have to be revised.

The second line declares a NeoPixel *object*.
In our case, our tape used is called "ribbon1". We will refer to the latter later in the code to control the ribbon/strip of pixels.
There are 3 arguments/parameters in the parentheses of the function called **Adafruit_neoPixel(parameter1, parameter2, parameter3):**

Parameter 1 = The number of chained NeoPixels
Parameter 2 = Data pin number n
Parameter 3 = Pixel type (flags), to be combined together as needed

NEO_KHZ800 data stream at 800 KHz (most NeoPixel based on w/WS2812 LEDs)
NEO_KHZ400 data stream at 400 KHz (For classic 'v1' FLORA pixels (not V2) driven by WS2811)
NEO_GRB Pixels are connected in GRB data stream (GRB=Green,Red,Blue=Green,Red,Blue - most NeoPixel products)
NEO_RGB Pixels are connected in RGB data stream (RGB=Red,Green,Blue=Red,Green,Blue - FLORA v1 pixels, not v2)

In the example given above, the 3 parameters are:
The number of NeoPixels chained in the ribbon/strip is set to 60, which corresponds to one meter of medium density tape. Change this setting to match the number of pixels you are using.
The pin number to which the NeoPixel ribbon (or other device) is connected. This should be a number but we can also use the symbol "PIN" precitedfrom mment declared and refer to it.
A value identifying the type of NeoPixel connected. In most cases, you don't have to worry about it... And you can just define two arguments. This code example is just very meticulous to offer a complete descriptive description. If you use classic Flora Pixels in "V1" version, they require the use of NEO_KHZ400 + NEO_RGB as a parameter.

3- in the setup() function, you must call *begin()* to prepare the data pin for NeoPixels:

```
void setup() {
ribbon1.begin(); // Initializes all pixels to 'off'
}
```

The second line, strip.show(), is not absolutely necessary, it is only there to be meticulous. The function pushes the data to the pixels... since no color is yet given to the different pixels. Here, however, they are initialized to the "off"/off state... to avoid having any luminous state initialized by a previous program.

4- In the loop(), all that remains is to control and control the color of a Pixel.

```
ruban1.setPixelColor(n, red, green, blue);
```

n: The first argument — n in this example — is the pixel number along the ribbon, starting at 0 for the closest to Arduino. If you have a ribbon/chain of 30 pixels, they are numbered from 0 to 29 (as is commonly the case in computing). Many locations in the code use a for loop, using the loop's counter variable as an argument identifying the pixel number... This makes it possible to give a value to several pixels.

The following 3 arguments define the pixel color by specifying the brightness level in red, green, and blue.
red: Red brightness level, 0 is the most held (off) and 255 is the maximum brightness.
green: Green brightness level, 0 is the most held (off) and 255 is the maximum brightness.
blue: Blue brightness level, 0 is the most held (off) and 255 is the maximum brightness.

To initialize the 12th pixel (so the number 11, counting from 0) to the magenta color (red + blue), you must write the code:

```
ruban1.setPixelColor(11, 255, 0, 255);
```

Important !!

```
Ribbon1.show();  // transmet pixel data in RAM at NeoPixels
```

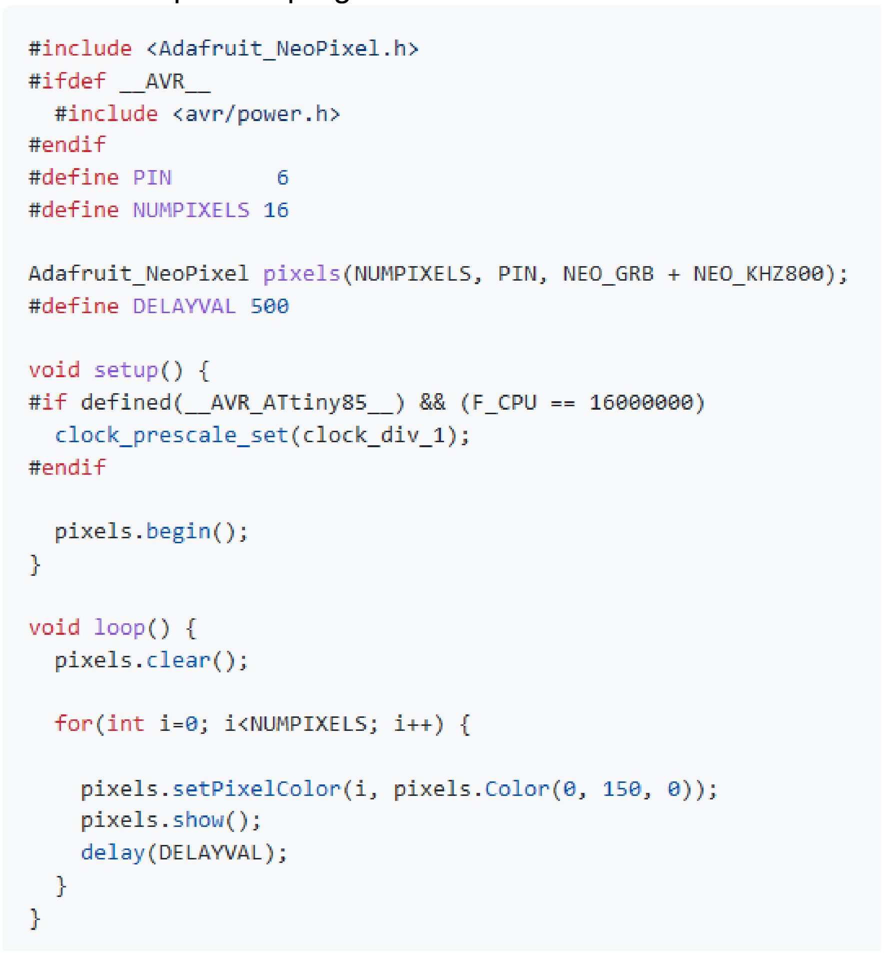
The show() function updates the entire ribbon/chain at once. While this extra step may seem boring, it's actually a good thing. If every call to setPixelColor() had an immediate effect, your animation would have bursts rather than updating the ribbon seamlessly.

5- Can I have different NeoPixel objects on different pins?
Obviously! Each order pin requiring its own declaration with a unique name:

```
Adafruit_NeoPixel ruban_a = Adafruit_NeoPixel(16, 5);
Adafruit_NeoPixel ruban_b = Adafruit_NeoPixel(16, 6);
```

The above declaration creates two NeoPixel objects, one on pin 5 and one on pin 6, each containing 16 pixels and using the implicit NeoPixel configuration (NEO_KHZ800 + NEO_GRB).

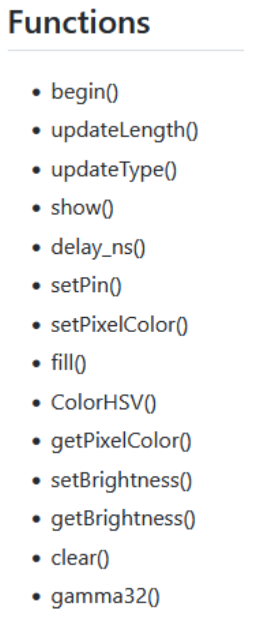
6- An example of a program:



```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
  #include <avr/power.h>
#endif
#define PIN        6
#define NUMPIXELS 16

Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
#define DELAYVAL 500

void setup() {
#if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
  clock_prescale_set(clock_div_1);
#endif

  pixels.begin();
}

void loop() {
  pixels.clear();

  for(int i=0; i<NUMPIXELS; i++) {

    pixels.setPixelColor(i, pixels.Color(0, 150, 0));
    pixels.show();
    delay(DELAYVAL);
  }
}
```

Ribbon1. clear(); // Fills the entire NeoPixel strip with 0 / black / off

The functions included in the library



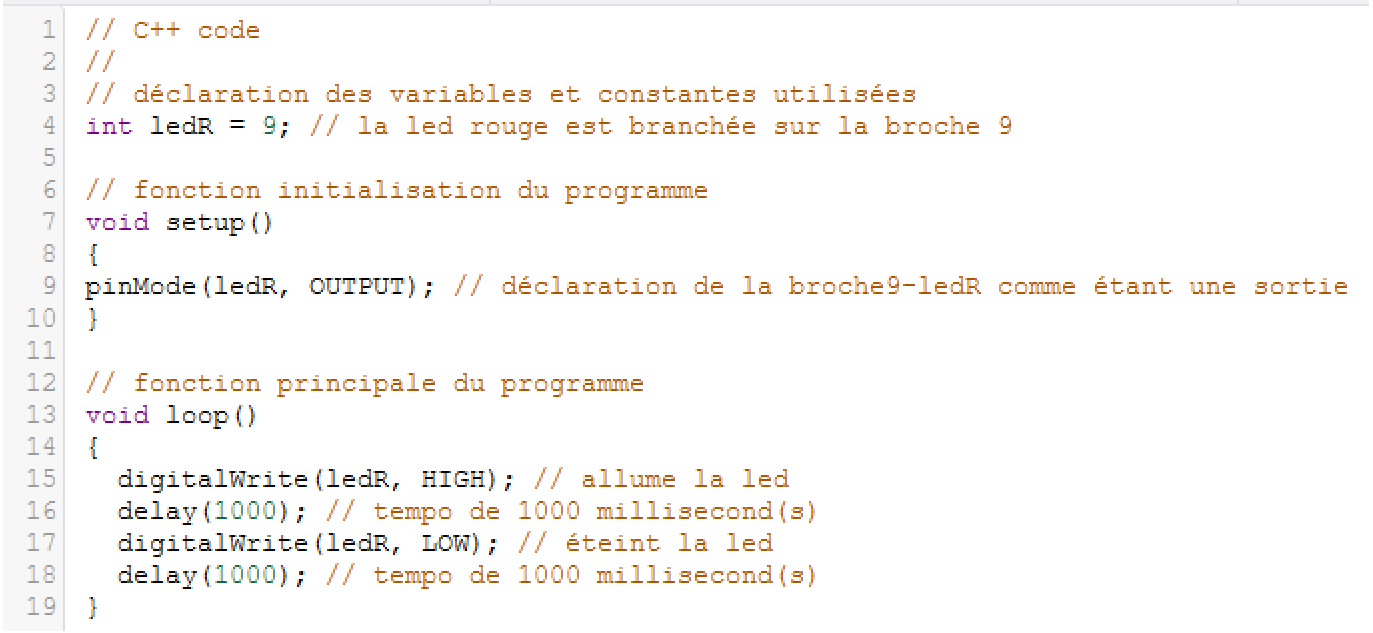
Functions

- begin()
- updateLength()
- updateType()
- show()
- delay_ns()
- setPin()
- setPixelColor()
- fill()
- ColorHSV()
- getPixelColor()
- setBrightness()
- getBrightness()
- clear()
- gamma32()

Note : to write a comment, we use // followed by the comment

2 – LED PROGRAMMING

Exercise n°1: understanding the 1 Led program



```
// C++ code
//
// déclaration des variables et constantes utilisées
int ledR = 9; // la led rouge est branchée sur la broche 9

// fonction initialisation du programme
void setup()
{
pinMode(ledR, OUTPUT); // déclaration de la broche9-ledR comme étant une sortie
}

// fonction principale du programme
void loop()
{
  digitalWrite(ledR, HIGH); // allume la led
  delay(1000); // tempo de 1000 millisecond(s)
  digitalWrite(ledR, LOW); // éteint la led
  delay(1000); // tempo de 1000 millisecond(s)
}
```

Note: // comment

Observed operation:

Name of the only variable :

Data type:

Assigned value:

What is it associated with? What is it used for?

What is **pinMode (parameter1, parameter2)** used for ?

What is the **digitalWrite function (parameter1, parameter2)** used for?

What is the **delay** function **used for**?

Exercise n°2 : programming of 2 LEDs

Lines of code added in red

```
// C++ code
//
Declaring variables and constants used
int ledR = 9; The red LED is plugged into pin 9

Program initialization function
void setup()
{
pinMode(ledR, OUTPUT); definition of the 9-ledR spindle as an output
}

Main function of the program
void loop()
{
  digitalWrite(ledR, HIGH); Lights the LED
  delay(1000); // tempo de 1000 millisecond(s)
  digitalWrite(ledR, LOW); Turns off the LED
  delay(1000); // tempo de 1000 millisecond(s)
}
```

3 – PROGRAMMING A RIBBON OF LEDS

Exercise n°3 : understanding the program

Observed operation:

Highlight in red the names of the chosen variables

```
#include <Adafruit_NeoPixel.h> // inclusion of specific library

#define PIN 2 // declaration of the PIN constant at value 2
the PIN constant contains the number of the pin where the LED tape is connected

#define NUMPIXELS 12 // declaring the constant NUMPIXELS at the value 12
NUMPIXELS contains the number of pixels on the ribbon

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

Declaring variables used in the program
int delayval = 100; Tempo duration of 100 milliseconds
int redColor = 0; declaring the variable called redColor and initialized to 0
int greenColor = 0;
int blueColor = 0;

void setup() {
  // Initialize the NeoPixel library.
  pixels.begin();
}

void loop() {
 setColor(); calling the function


  for (int i=0; i < NUMPIXELS; i++) {
    // pixels. Color takes RGB values, from 0,0,0 up to 255,255,255
    pixels.setPixelColor(i,pixels. Color(redColor, greenColor, blueColor));

    // This sends the updated pixel color to the hardware.
    pixels.show();

    // Delay for a period of time (in milliseconds).
    delay(delayval);
  }
}

declaring the function called setColor()
assigning a random value between 0 and 255 to the 3 RGB variables
void setColor(){
  redColor = random(0, 255);
  greenColor = random(0,255);
  blueColor = random(0, 255);
}
```

Faster ignition: what? How many?

Slower ignition: what? How many?

RGB ?

Yellow : what? how much?

If we set the 3 colors to 255, what color do we get?

How do I turn off a LED ?

What is the command to turn off all the LEDs at the same time?

How is the **for()** loop defined ? What is it used for?