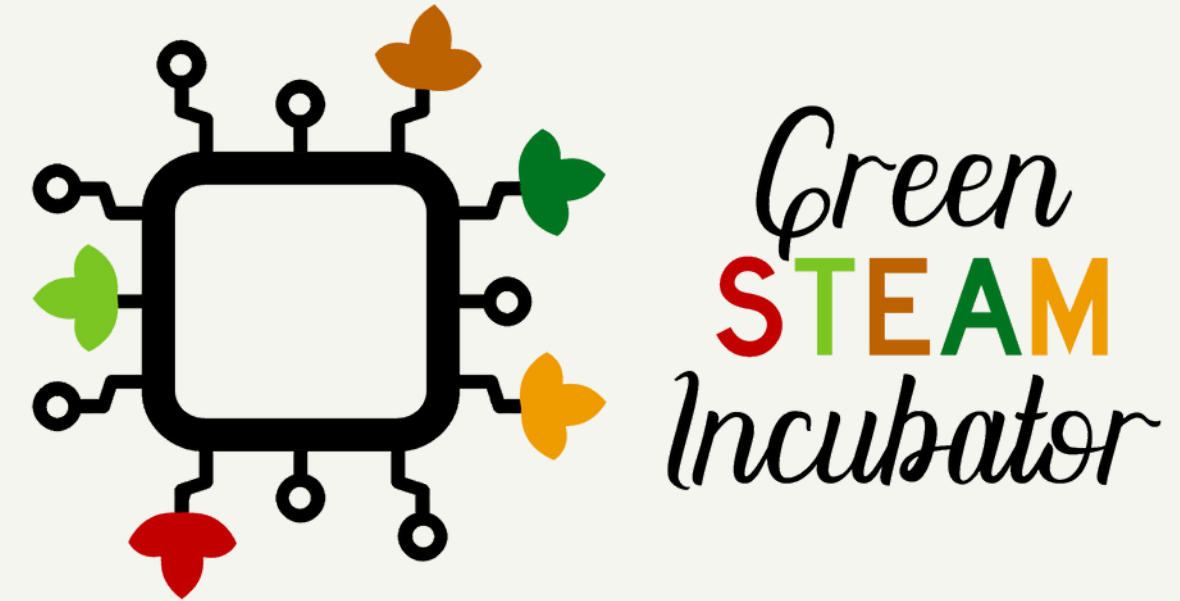


THE ARDUINO FUNDAMENTALS

HOW TO PROGRAMME IN ARDUINO, UPLOAD PROGRAMS TO ARDUINO BOARD

THE GREEN STEAM INCUBATOR



Partners

THE CONSORTIUM



The European Commission support for the production of this publication does not constitute endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein; Project Number: 2019-3-CY02-KA205-001692

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



Educational Objectives



- ✓ Learn the basics on how to program Arduino
- ✓ Learn how to upload programs (sketches) to Arduino board



Required Material & Resources

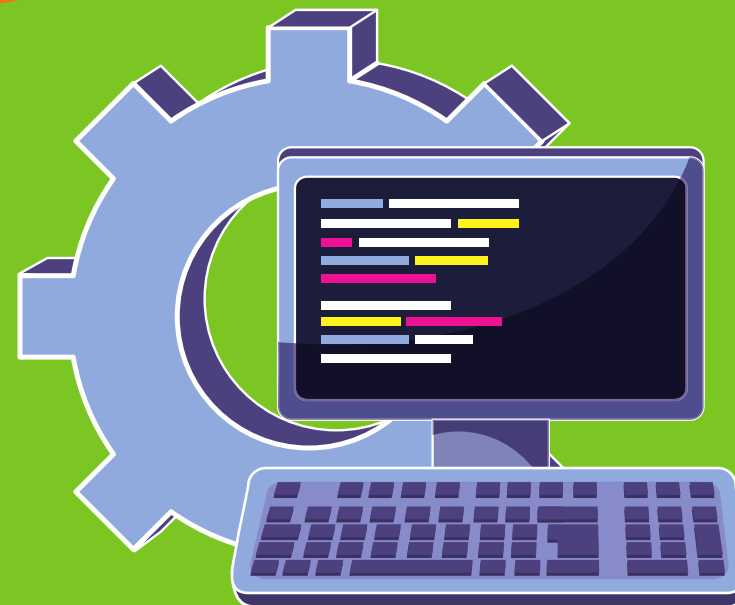
- Arduino UNO
- Software Arduino IDE
 - Computer
- Arduino USB cable

Arduino programs are called "sketches"

WHAT IS A SKETCH IN ARDUINO?

An Arduino Sketch is a set of instructions that you create to accomplish a particular task

Arduino uses the C programming language



How to program Arduino

Arduino programs can be divided into three main parts:

▶ Structure


▶ Values (variables and constants)

▶ Functions

In this session, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.



Software structure consists of two main functions:



Setup() function

```
Void setup ( ) {  
}
```

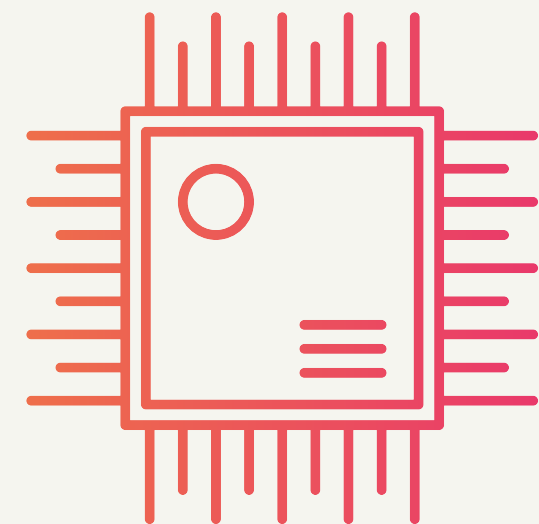
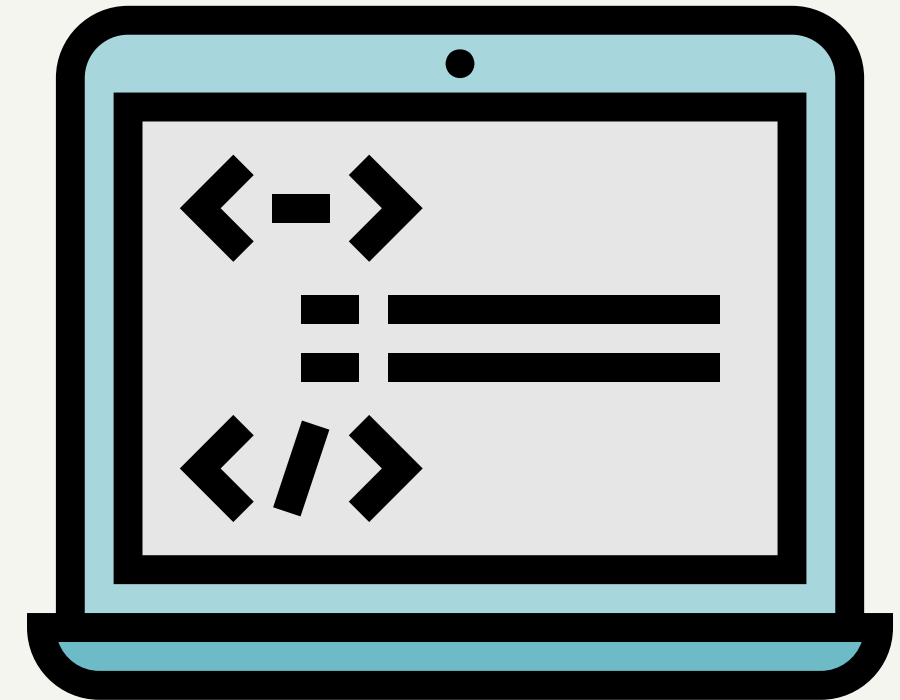
The setup() function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once after each power-up or reset of the Arduino board.



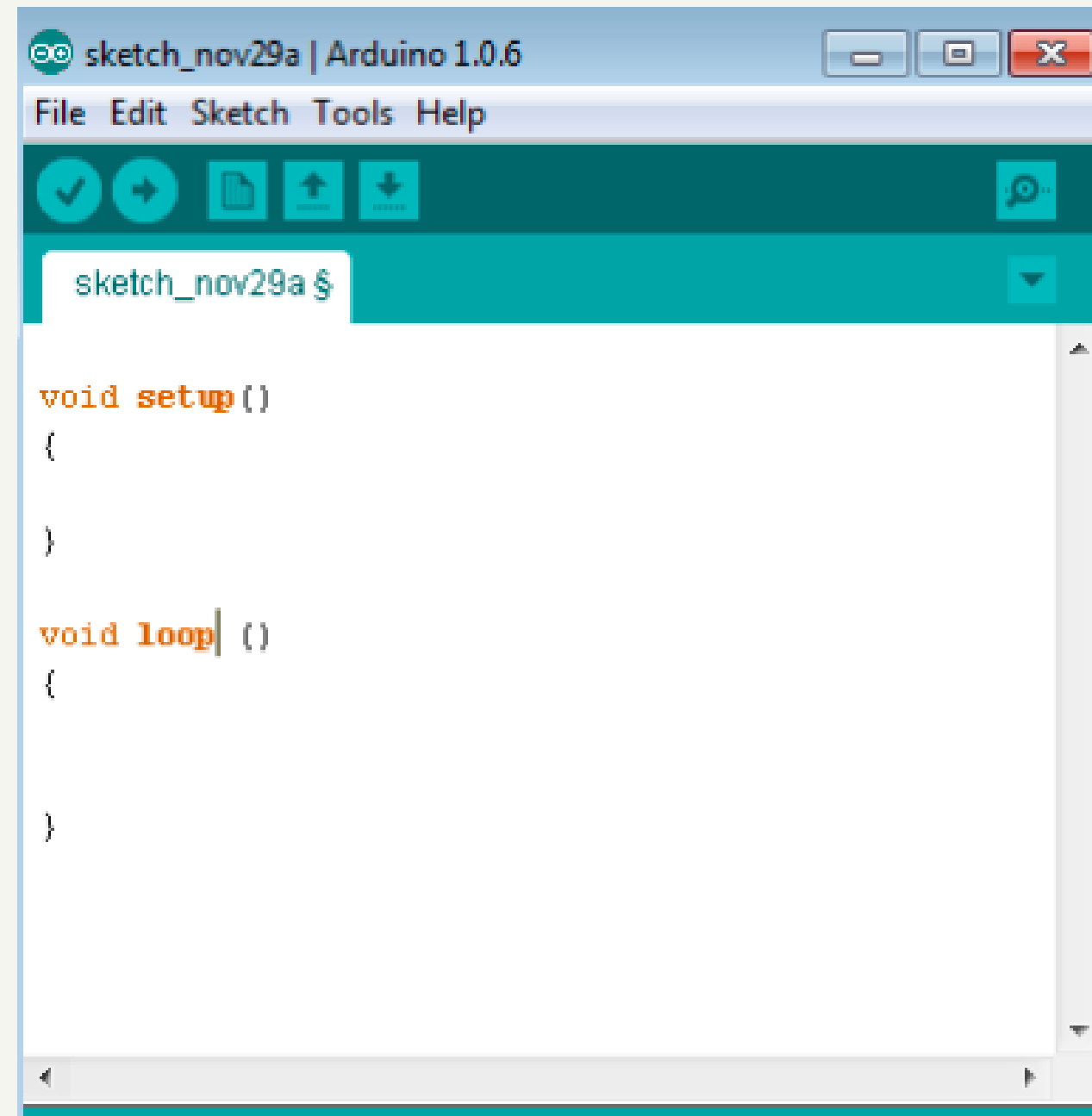
Loop() function

```
Void Loop ( ) {  
}
```

After creating a setup() function, the loop() function does precisely what its name suggests and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

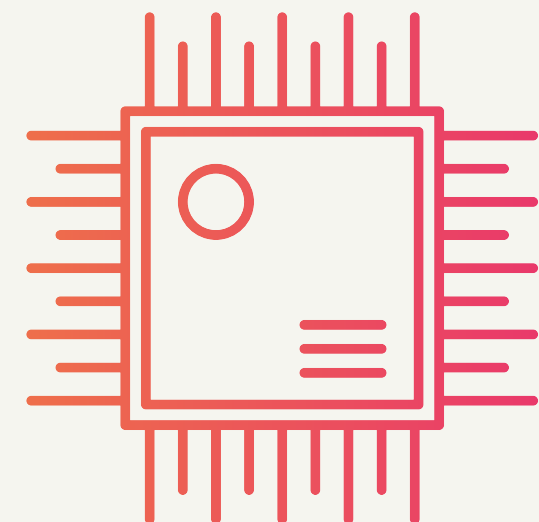
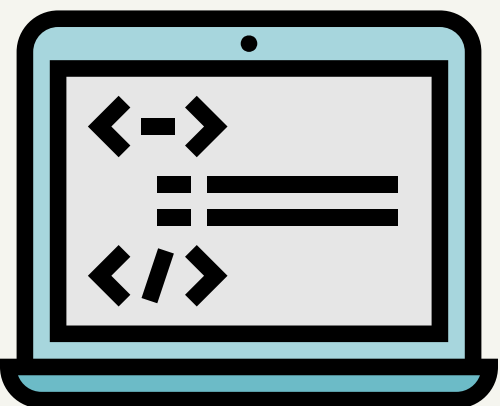


PROGRAMME STRUCTURE

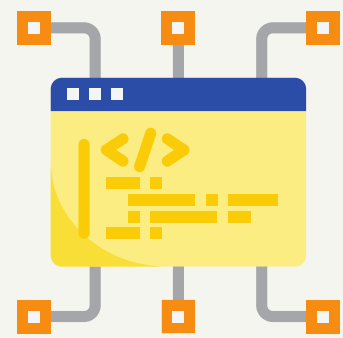


```
sketch_nov29a | Arduino 1.0.6
File Edit Sketch Tools Help
sketch_nov29a $
void setup()
{
}
void loop() ()
{
}
```

Source: https://www.tutorialspoint.com/arduino/arduino_program_structure.htm



Arduino - Data Types

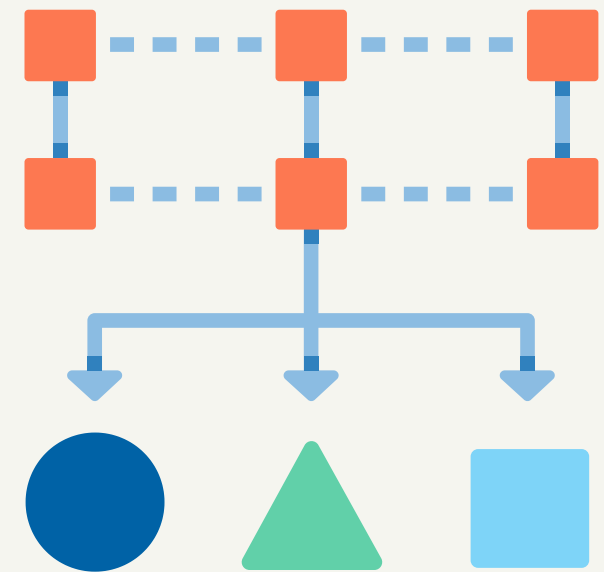


Data types in C refers to an extensive system used for declaring variables or functions of different types.

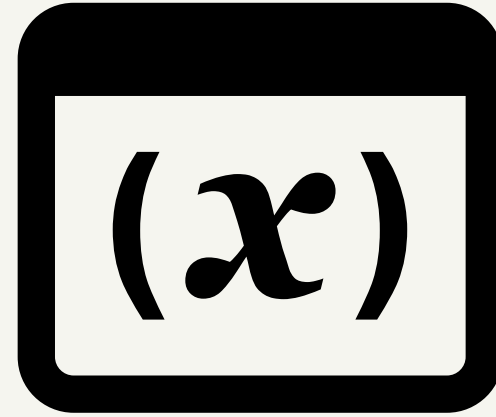
The type of a variable determines how much space it occupies in the storage and how the bit pattern stored is interpreted.

void	Boolean	char	Unsigned char	byte	int	Unsigned int	word
long	Unsigned long	short	float	double	array	String-char array	String-object

Data Types in Arduino



Arduino - Variables (I)



Variables in the C programming language, which Arduino uses, have a property called scope.

A scope is a region of the program, and there are three places where variables can be declared.

1

Local variables:

inside a function or a block

2

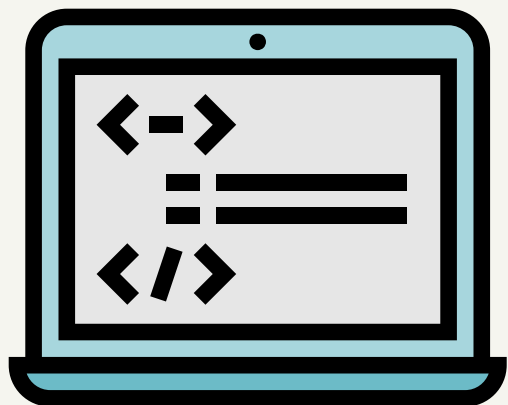
Formal parameters:

in the definition of function parameters

3

Global parameters:

outside of all functions



Arduino - Variables (II)

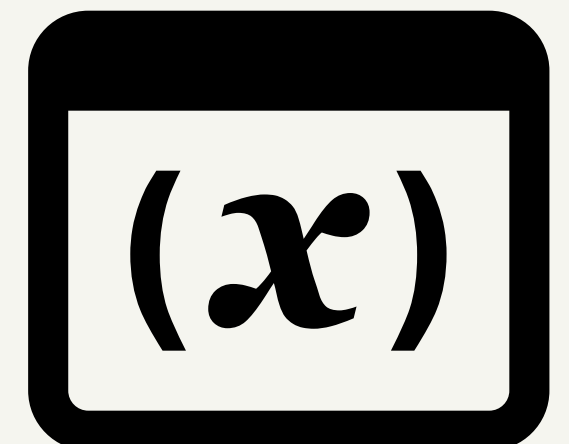
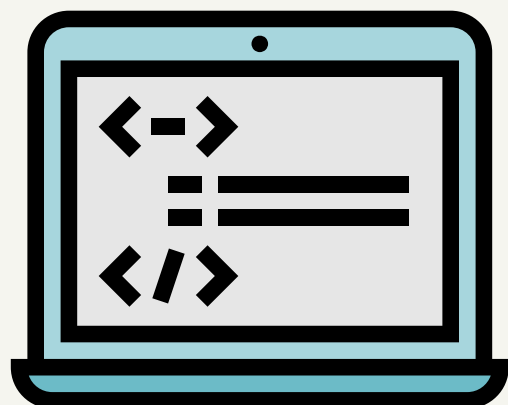
Local VS Global Variables

```
Void setup () {  
  
}  
  
Void loop () {  
  int x , y ;  
  int z ; Local variable declaration  
  x = 0 ;  
  y = 0 ; actual initialization  
  z = 10 ;
```

Example of Local Variables

```
Int T , S ;  
float c = 0 ; Global variable declaration  
  
Void setup () {  
  
}  
  
Void loop () {  
  int x , y ;  
  int z ; Local variable declaration  
  x = 0 ;  
  y = 0 ; actual initialization  
  z = 10 ;  
}
```

Example of Global Variables

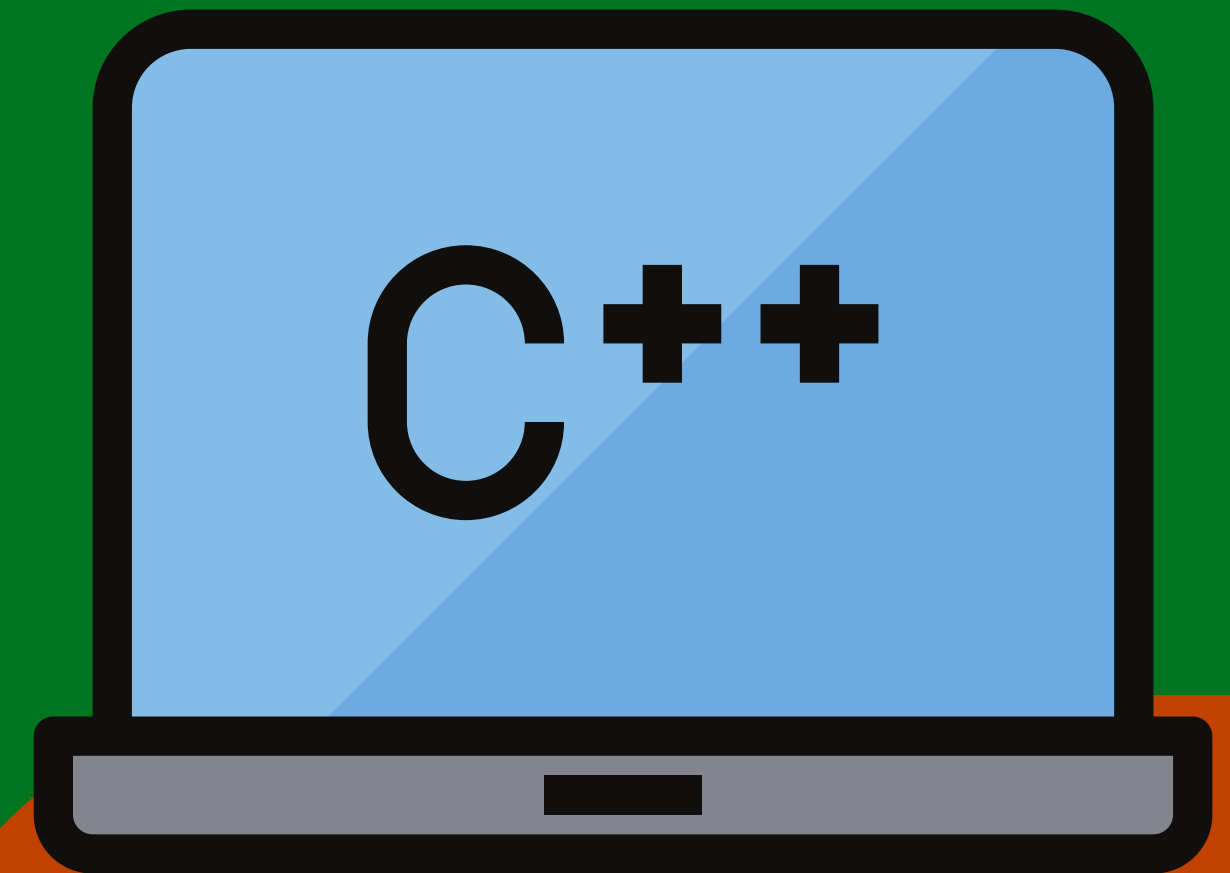


What is an operator?

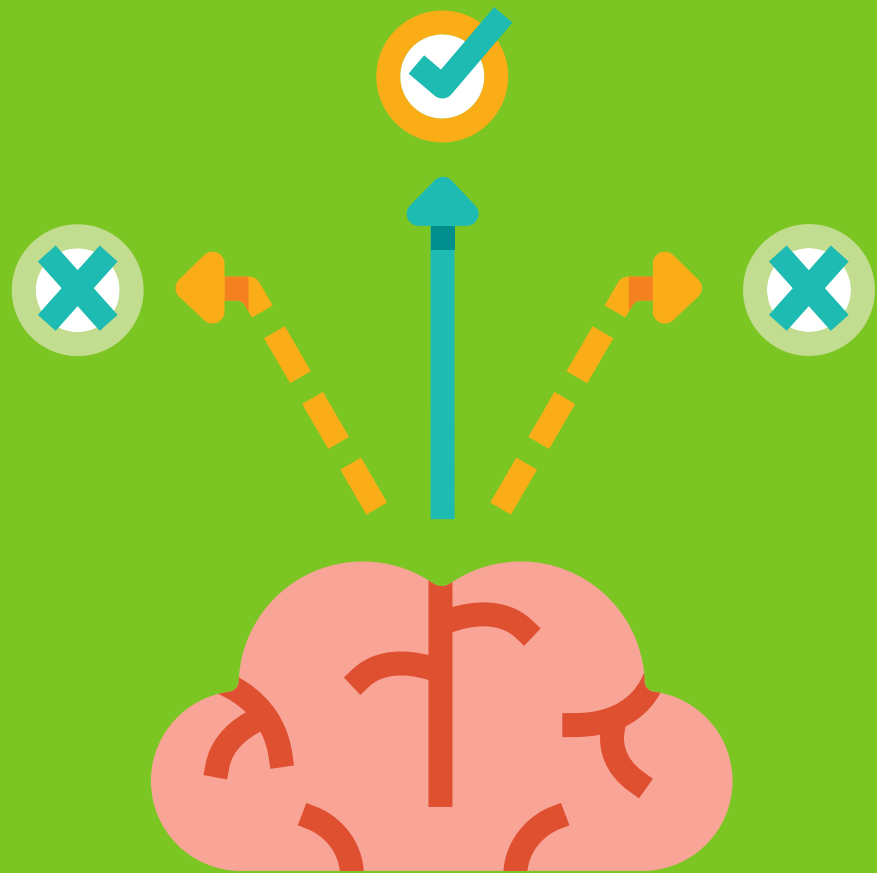
An operator is a symbol that tells the compiler to perform specific mathematical or logical functions.

C language is rich in built - in operators, including the following types:

- Arithmetic Operators
- Comparison Operators
- Boolean Operators
- Bitwise Operators
- Compound Operators



Arduino - Control Statements



Decision-making structures require that the programmer specify one or more conditions to be evaluated or tested by the program.

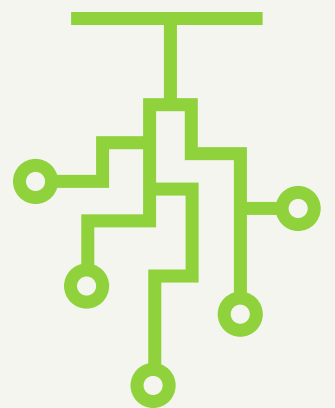
It should be along with a statement or statements to be executed if the condition is determined to be true.

Optionally, other statements can be executed if the condition is determined to be false.



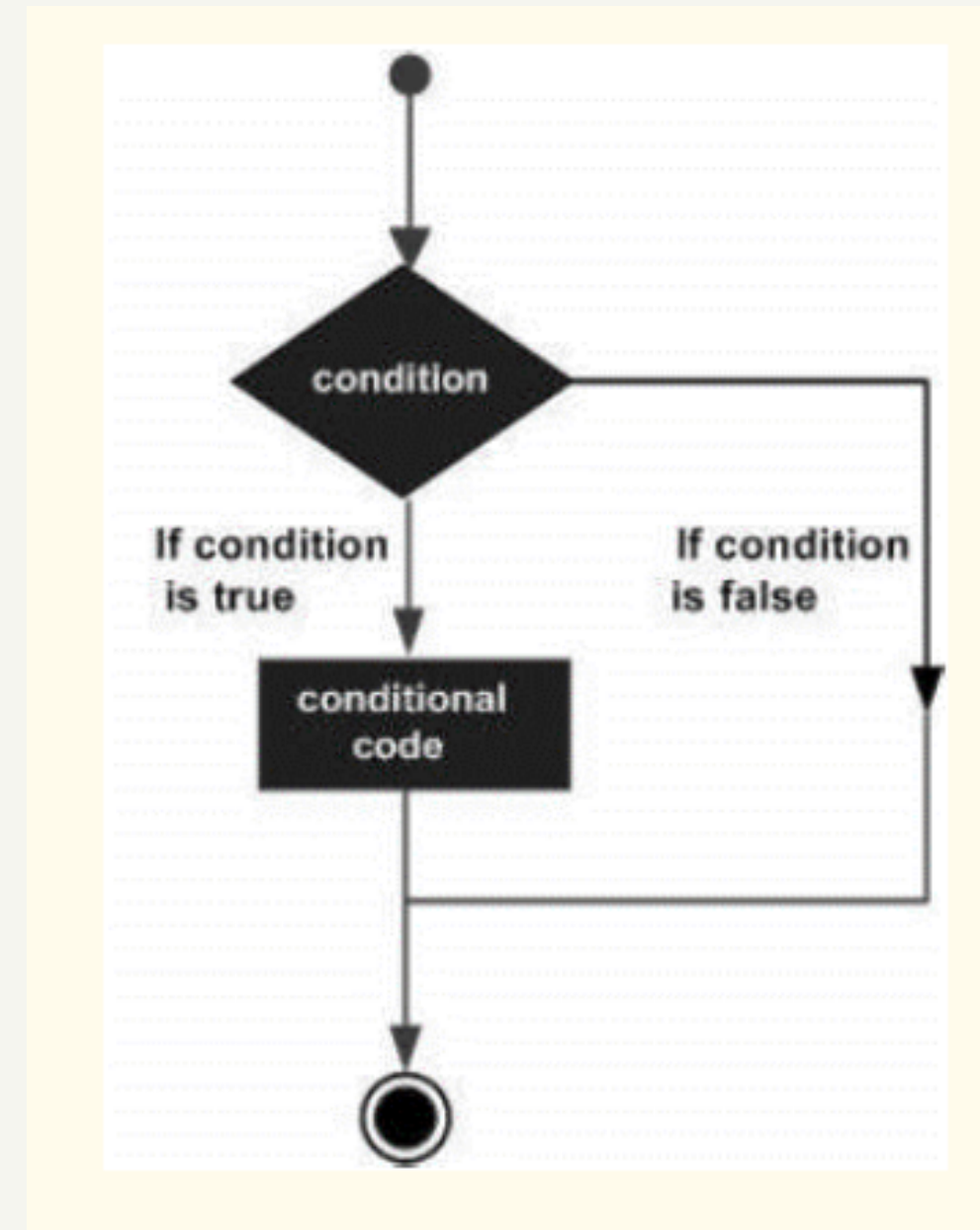
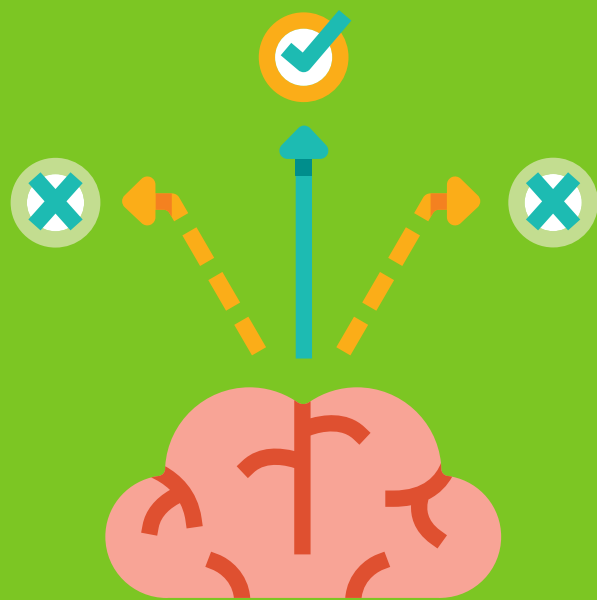
Control Statements are elements in the Source Code that control the flow of program execution

- if statement
- if ... else statement
- if ... else if ... statement
- switch case statement
- Conditional Operator



Arduino - Control Statements

Decision - making structure

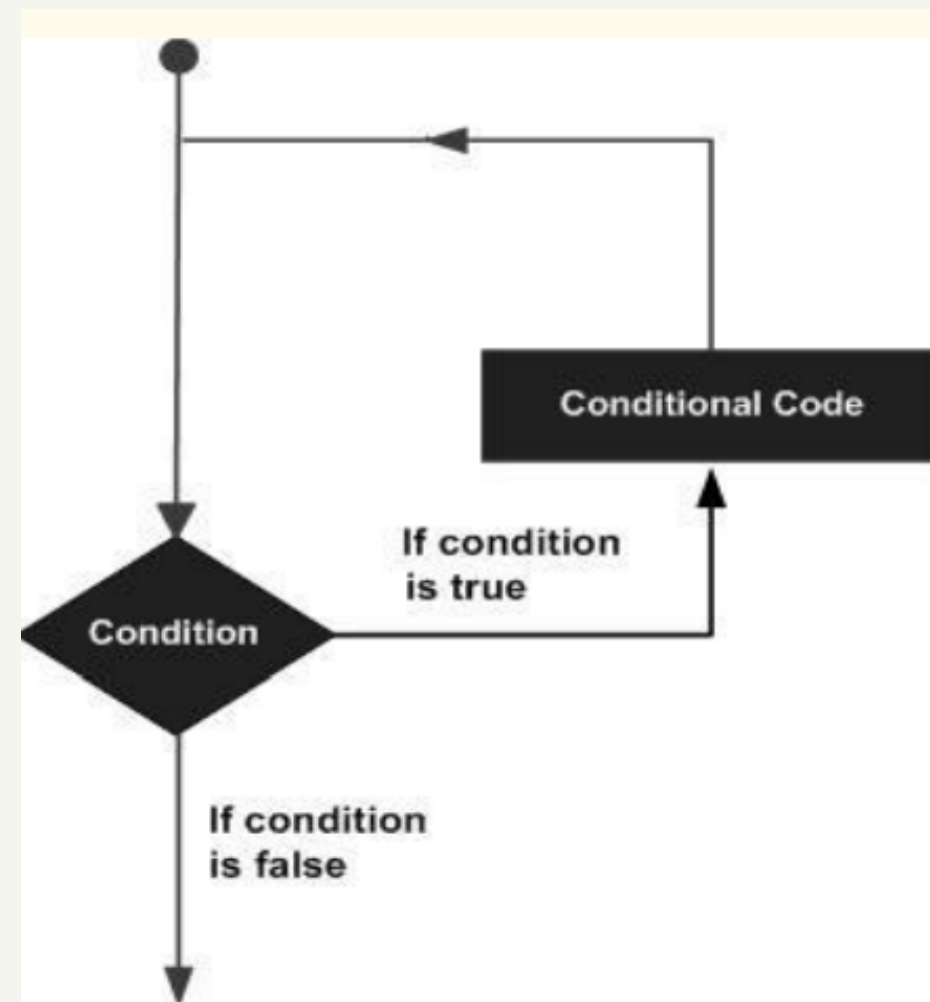


The general form of a typical decision-making structure found in programming languages

What is a loop?

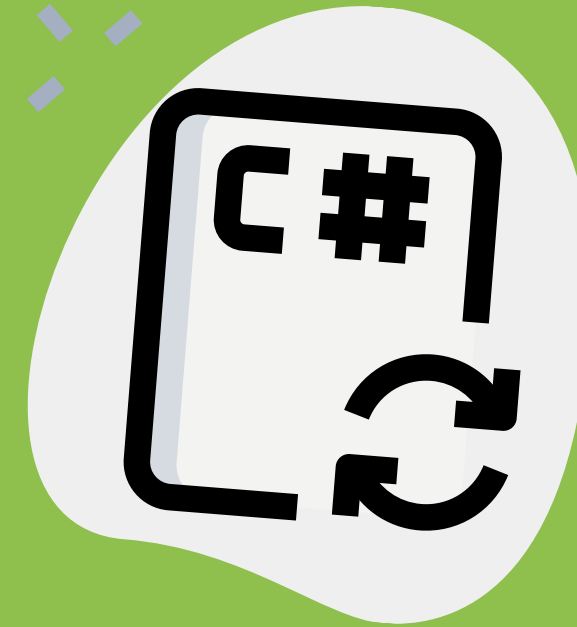


A loop statement allows us to execute a statement (or a group of statements) multiple times - a block of code that will repeat over and over again.








The general form of a loop statement in the most of the programming languages

Arduino Loops

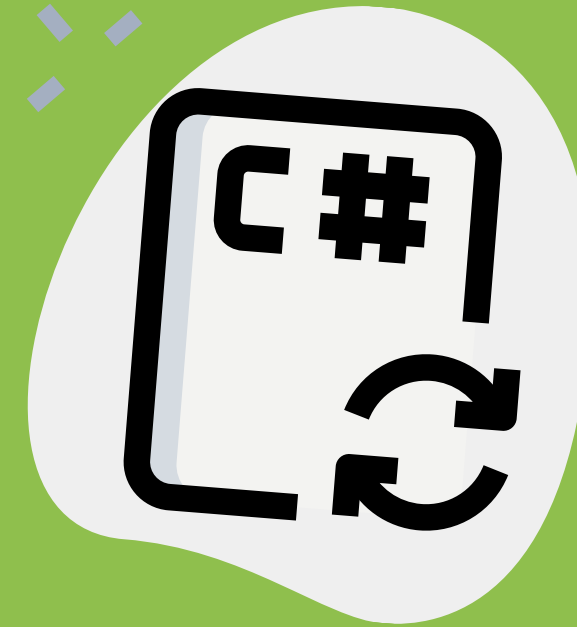


Loops in C

C programming language provides the following types of loops to handle looping requirements:


-  While loop: will repeat continuously and infinitely until the expression inside the parenthesis () becomes false
-  do ... while loop: the loop - continuation condition is tested at the beginning of the loop before performing the loop's body
-  for loop: executes statements a predetermined number of times
 -  Nested Loop: a loop inside another loop
 -  Infinite loop: no terminating condition

Arduino Loops



Create your first sketch in the Arduino IDE



 You will create and upload a simple sketch that will cause the Arduino's LED to blink repeatedly by turning it on and then off for 1-second intervals.



To begin, connect your Arduino to the computer with the USB cable

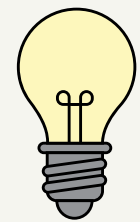
Open the IDE & Choose Tools4Serial Port

***Make sure the USB port is selected (ensures that the Arduino board is properly connected)

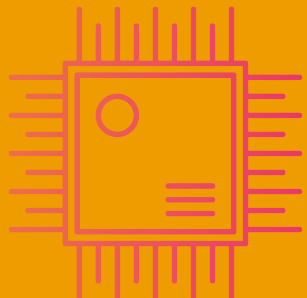


The first stage in creating any sketch is to add the void setup () function: this function contains instructions to execute only once, each time it is reset or turned on

```
void setup()  
{  
  
}
```




Our program will blink the LED (connected to Arduino's **digital pin 13**).
A **digital pin** can detect or generate an electrical signal. In our case, we will generate an electrical signal.

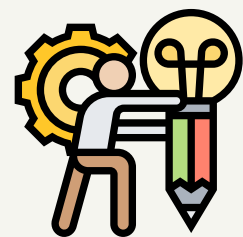


Enter the following into your sketch between the braces { }

```
pinMode(13, OUTPUT); // set digital pin 13 to output
```



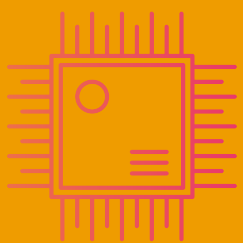
every function in
Arduino sketches end
with a semicolon



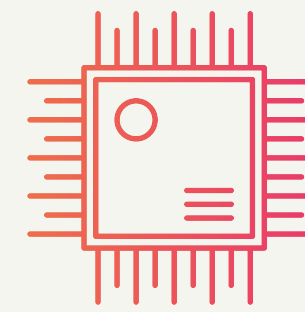
Pin to **OUTPUT**, which will **generate** an electrical signal.



If you want it to **detect** an incoming electrical signal, then use **INPUT** instead.



First Steps (III)

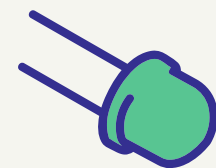


Our code until now

General Info /
Name

```
/*  
Arduino Blink LED Sketch  
by Mary Smith, created 09/09/12  
*/  
  
void setup()  
{  
  pinMode(13, OUTPUT); // set digital pin 13 to output  
}
```

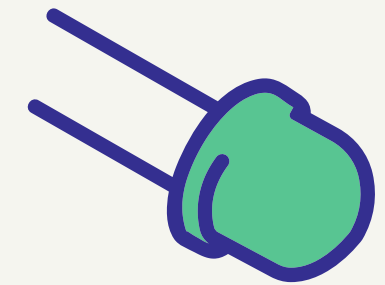
First Step /
Void setup



Our goal is to make the LED blink repeatedly.

We will create a loop function to execute an instruction repeatedly, until the power is shut off or someone presses the RESET button

How to make our LED blink




- Enter the code after the void setup () section to
- create an **empty loop**

```
void loop()  
{  
  // place your main loop code here:  
}
```

NEXT ➔

Enter the actual function in the void loop () for the Arduino to execute

```
digitalWrite(13, HIGH); // turn on digital pin 13  
delay(1000); // pause for one second  
digitalWrite(13, LOW); // turn off digital pin 13  
delay(1000); // pause for one second
```

 Click Verify to make sure you have entered everything correctly & Don't Forget to Save your Sketches!



Code Analysis

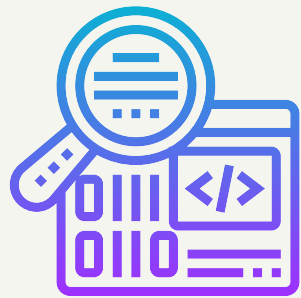


```
digitalWrite(13, HIGH); // turn on digital pin 13
delay(1000); // pause for one second
digitalWrite(13, LOW); // turn off digital pin 13
delay(1000); // pause for one second
```

digitalWrite() function controls the voltage from a digital pin
✓ 13 - pin 13 of the LED
✓ HIGH - current will flow from the pin, and the LED will turn on

The delay() function causes the sketch to do nothing for a period of time —
1.000 milliseconds, or 1 second

We turn off the voltage to the LED with digitalWrite(13,LOW);
We pause again for 1 second while the LED is off, with delay(1000);



Final Sketch



```
/*  
Arduino Blink LED Sketch  
by nickname, created 28/12/20  
*/  
  
void setup()  
{  
  pinMode(13, OUTPUT); // set digital pin 13 to output  
}  
  
void loop()  
{  
  digitalWrite(13, HIGH); // turn on digital pin 13  
  delay(1000); // pause for one second  
  digitalWrite(13, LOW); // turn off digital pin 13  
  delay(1000); // pause for one second  
}
```



Verify & Save

SAVE

Next, you should verify your sketch to ensure that it has been written correctly so Arduino can understand.

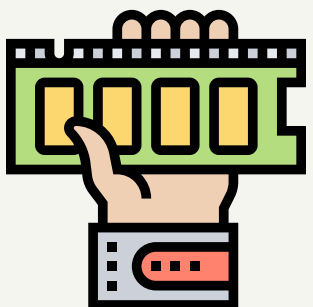
To verify your complete sketch, click [Verify](#) in the IDE and wait a moment.

```
Done compiling.  
  
Binary sketch size: 1,076 bytes (of a 32,256 byte maximum)  
  
17 Arduino Uno on COM14
```

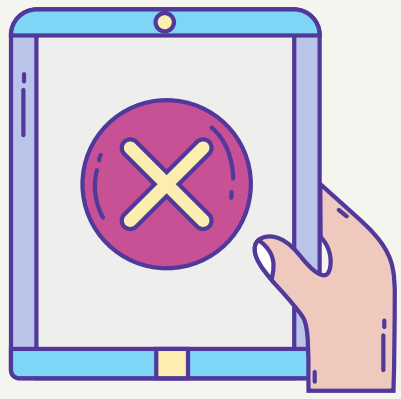
The sketch has been verified.



This "Done compiling" message tells you that the sketch is OK to upload to your Arduino.



It also shows how much memory it will use (1,076 bytes in this case) of the total available on the Arduino (32,256 bytes).



Verification error

In case that your sketch is not OK, when you click Verify the message window should display a **verification error message**

You forgot to add a semicolon at the end of the second delay(1000) function)

blinky:16 where in the sketch the IDE thinks the error is located

```
digitalWrite(13, LOW); // turn off digital pin 13
delay(1000) // pause for one second
}
```

expected ';' before '}' token

blinky.cpp: In function 'void loop()':
blinky:16: error: expected ';' before '}' token

17 Arduino Uno on COM14

The message window with a verification error



Final Step: Uploading & Running your Sketch



Once having saved your sketch, ensure that your Arduino board is connected, and click Upload in the IDE.

The IDE may verify your sketch again and then upload it to your Arduino board.



The TX/RX LEDs on your board should blink during this process, indicating your program functions properly



Thank you!

QUESTIONS?



The European Commission support for the production of this publication does not constitute endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein; Project Number: 2019-3-CY02-KA205-001692

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

