Arduino Programming Basics - Recap (*Arduino For Beginners* Course)

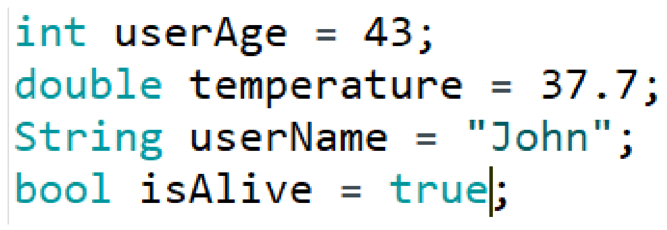
Here is a quick recap of the programming concepts you've seen in the course. If you have any doubt, please refer to this PDF or re-watch the corresponding lessons.

Variables

A variable:

- Stores an information that can be reused later in the program
- Can store different kinds of values (data type)
- Needs to be declared before initialized and used
- Exists only in the scope where it is declared
- Can be modified. You can also create non-modifiable variables, or constants.

Example:



```
int userAge = 43;
double temperature = 37.7;
String userName = "John";
bool isAlive = true;
```

Data Types

When declaring a variable, you have to choose a data type. Here are some of the most common ones:

- Round numbers:
    - byte: value between 0 … 255
    - int: value between -32,768 … 32,767
    - long: value between -2,147,483,648 ... 2,147,483,647
- Float numbers:
    - double
    - float
- Text:
    - String: this data type is specific to Arduino and doesn't exist in C/C++
- Binary value:
    - bool: true or false

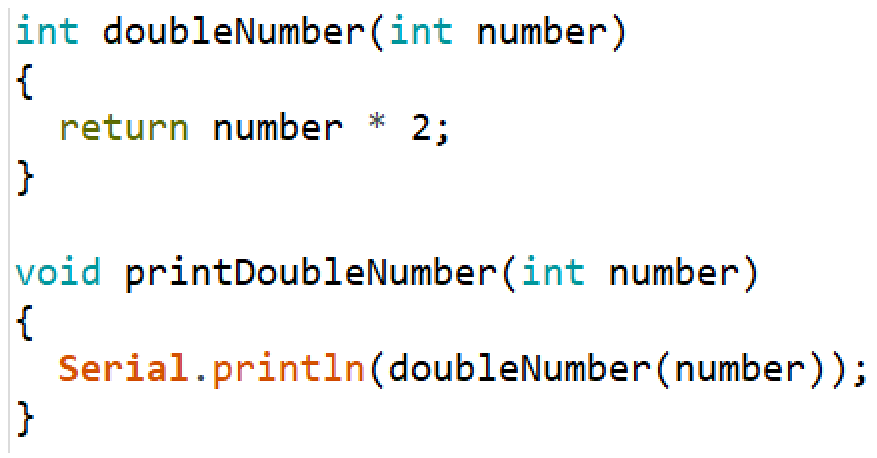
Note:
- With Round numbers, when you reach the maximum or minimum of the range, the value will “overflow”, which means that it will go to the opposite of the range. Make sure you use a data type with a big enough range so your variable doesn’t overflow.
- You can add “unsigned” in front of a round number (int, long), to only handle positive values. In that case, you also double the range of the variable in the positive side.

Functions

A function:
- Is a reusable block of code
- Avoids repetition of code in your program
- Makes your program more modular and more scalable
- Takes zero or more parameters as input
- Needs to return a value with a “return” statement (not needed if return type is “void”)

Notes:
- “setup” and “loop” functions are predefined Arduino functions and are called during the program execution
- You can call a function from another function.

Example:



```
int doubleNumber(int number)
{
  return number * 2;
}

void printDoubleNumber(int number)
{
  Serial.println(doubleNumber(number));
}
```

Scope

The scope of a variable also refers to the variable’s visibility.
- A variable’s scope depends on where the variable has been declared
- You can use a variable in a more nested scope, but not in a broader scope

- 2 variables can have the same name if they are on 2 unrelated scopes (and thus those are 2 different variables)
- For each parameter of a function, a new local variable is created inside the function and exists only in this scope (inside the curly braces of the function)

Conditions

With a condition you can decide to execute or not a block of code, depending on a conditional statement.

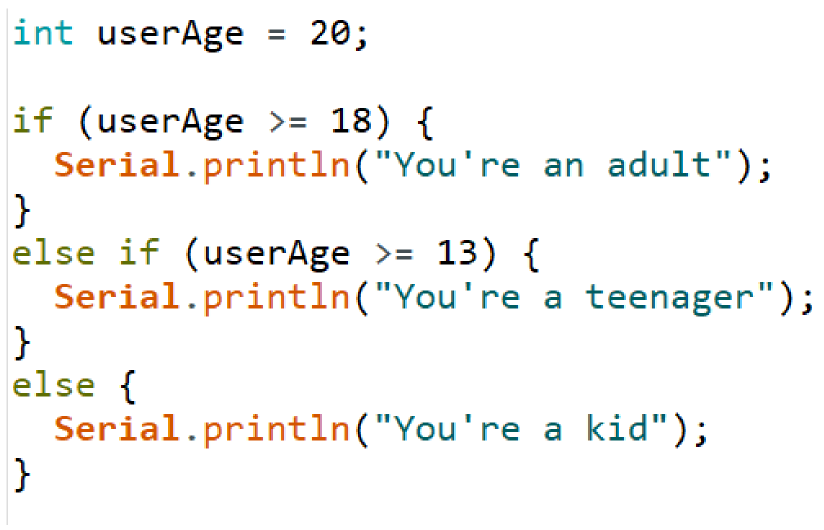
You can use multiple comparison operators :

- == : equal to (!!! don't forget to write 2 '=' or else it means that you are assigning a value, not doing a comparison !!!)
- != : not equal to
- < : strictly lower than
- > : strictly higher than
- <= : lower or equal than
- >= : higher or equal than

You can combine multiple conditional statements with logical operators :

- || : or
- && : and

Example:



```
int userAge = 20;

if (userAge >= 18) {
  Serial.println("You're an adult");
}
else if (userAge >= 13) {
  Serial.println("You're a teenager");
}
else {
  Serial.println("You're a kid");
}
```

Loops

A loop:
- Executes a block of code multiple times
- Allows you to reduce code repetitions

You can use 2 kinds of loops:
- while loop :
  - executes a block of code until a condition is false
  - Useful when you don’t know when exactly you need to stop the loop
- for loop :
  - executes a block of code for a given amount of times
  - Useful when you know how many times you’ll need to execute the code, or to go through an array

Example:

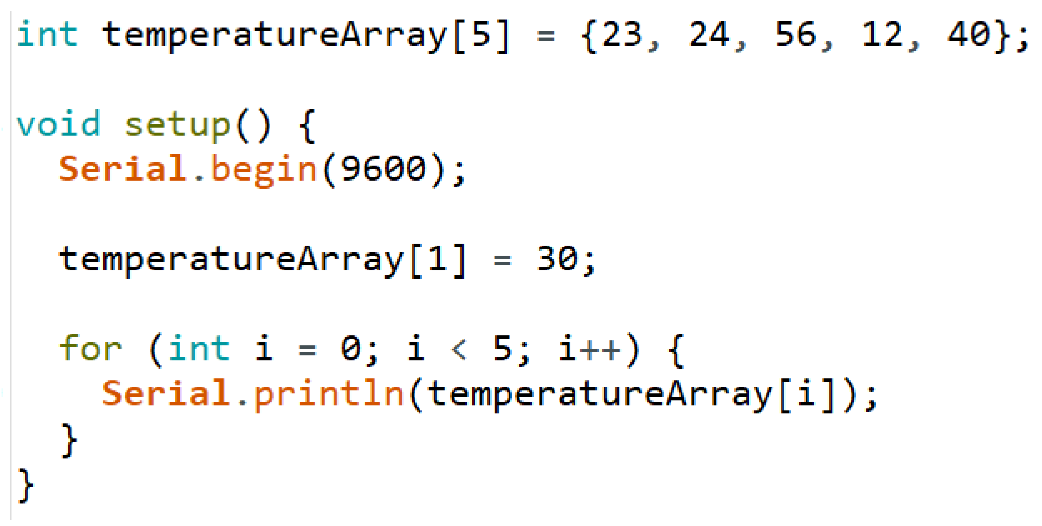
```
for (int i = 0; i < 10; i++) {
  Serial.println("Hello for");
}

int i = 0;
while (i < 10) {
  Serial.println("Hello while");
  i++;
}
```

Arrays

An array stores a collection of variables:
- All the values inside an array have the same data type
- The size of an array is set when it is declared (and cannot be changed afterwards)
- You access/modify the values with an index (starting at 0, not 1)

Note:
- Pay attention not to access/modify values outside of the array !

Example:



```
int temperatureArray[5] = {23, 24, 56, 12, 40};

void setup() {
  Serial.begin(9600);

  temperatureArray[1] = 30;

  for (int i = 0; i < 5; i++) {
    Serial.println(temperatureArray[i]);
  }
}
```