Home / Programming / MicroPython / **MicroPython Basics**

MicroPython Basics

An introduction to MicroPython

Author • Francesca Sanfilippo & Karl Söderby

Last revision • 03/03/2023

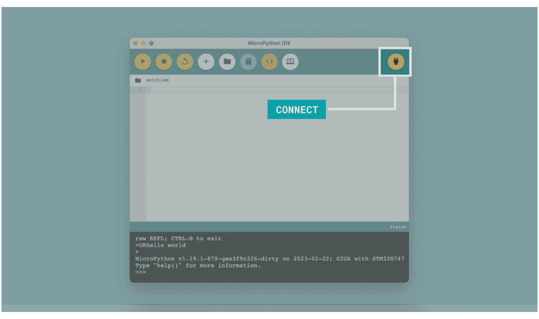
As you read in the Overview, MicroPython is an implementation of Python. In this page, you will find some basic and intermediate MicroPython scripts that can be used by any Arduino board. This includes some very common concepts such as variables, loops, delays, how to print and more.

> **Note:** There are some differences between MicroPython and Python, which mostly concern standard library and types, but also some language-level features.

After you download Arduino Lab for MicroPython, click the file for your system, extract and run the application. The interface is similar to Arduino IDE.

First of all, we need to connect our board to the computer via USB.

After connecting the board, click on the connect button, and select the port.



```
raw REPL; CTRL-B to exit
>OKhello world
>
MicroPython v1.19.1-879-gee3f9c326-dirty on 2023-02-22; GIGA with STM32H747
Type "help()" for more information.
>>>
```

Connect and select the port.

Now that you have connected your board, let's create a file that will contain the script that will run on your board. Click on New button to create your file. The editor automatically creates the main file in the board.

If you are not working with the editor, the file should be named main.py and should be saved to your board manually. The board will recognize this as the main program.

First MicroPython Script

Our first example is a basic script that will print

`Hello world!`

every second in the terminal. Paste the below script in the editor, and press the **"Play"** button.

COPY

```
print('Hello world!')
```

Program Constructs

There are three basic programming constructs which are Sequential, Looping and Branching.

- Sequential: a sequence of instructions.
- Looping: the program is executed according to the condition being used. There are two functions: while loop and for loop.
- Branching: it is a programming construct where a section of code is run only if a condition is met.

In this section we'll see some of program constructs.

Variables, While Loop and Sleep

In this second example we introduce the module time using import in order to pause the execution for a specific time. The module help us control the board with MicroPython.

COPY

```
import time
content = "Hello world!"

while True:
    print(content)
    time.sleep(1)
```

By definition a variable is a string of characters and numbers associated with a piece of information. We define the variable content `Hello World!`.

Then with the `while` loop we execute the statements as long as the condition is true. In the code we use the `sleep` function to pause the execution of the script for a second before it continues to print. This is imported from the `time` module.

Functions

A function is a block of code, a sequence of instructions composed by several statements, which runs only when it is called. You can pass the information as parameters into a function. A function can have input arguments, and can also have output parameters.

We can define our own functions, the most common way can be specified using the def keyword, inside the parentheses you can find the arguments if there are. Take a look to the example below:

COPY

```
def my_function():
    print("Hello world!")
```

Then you can call your function using the function name followed by parentheses:

COPY

```
my_function()
```

The function need two components: the header, starting with keyword def, followed by parentheses with inside the arguments and ending by colon (:) and the indented body is composed by descriptive string, function statements, return statements.

This script prints "Hello world!" every second. In addition, the function counter_function() increases the number every second and will be printed next to.

COPY

```
import time

content = "Hello world!"
```

```
count = 0

def counter_function():
    global count
    count = count + 1

while True:
    counter_function()
    print(content, count)
    time.sleep(1)
```

Conditionals and Loops

MicroPython supports logical conditions from mathematics, that can be used in several ways, the most common is an "if conditional" and “for loop”. The if statements is written by if keyword and it needs the indentation, otherwise you will get an error.

If/Else Statement

A if/else statement is used to handle conditions in your program. These statements guide the program while making decisions based on the conditions encountered by the program.

You can try the code below:

COPY

```
a = 42
b = 23
if a > b :
    print("a is greater than b
else :
    print("a is not greater th
```

The result in this case is always going to be:

`a is greater than b`, because `42` is larger than `23`.

For Loop

Simple use of a for loop and functions. This script counts to 10, and then back to 0.

COPY

```
import time

count = 0
```

```

def function_increase():
    global count
    count = count +1
    print(count)

def function_decrease():
    global count
    count = count -1
    print(count)

while True:
   for x in range(10):
    function_increase()
    time.sleep(1)

   for x in range(10):
    function_decrease()
    time.sleep(1)
```

Arrays

An array is one of the most known and used construct in programming. In MicroPython an array by definition is a collection of elements (values or variables), selected by one or more indices computed at run-time, you refer to an array element by referring to the index number.

COPY

```
myFruit = ['orange', 'pe

def printFruitNames():
   for fruit in myFruit
      print(fruit)

printFruitNames()
```

Suggest changes

The content on docs.arduino.cc is facilitated through a public GitHub repository. If you see anything wrong, you can edit this page here.

Need support?

Help Center
Ask the Arduino Forum
Discover Arduino Discord

License

The Arduino documentation is licensed under the [Creative Commons Attribution-Share Alike 4.0](#) license.

© 2024 Arduino