

# Brush up MicroPython

**Brush up** = repasar, refrescar. Con esta página NO se quiere realizar un curso de MicroPython pues excede de los propósitos del curso, pero sí una visión rápida de las diferentes instrucciones que se dan en el curso.

Este repaso es inspirado en el [tutorial MicroPython Basics](#) autora Francesca Sanfilippo & Karl Söderby

## Lo básico

Hemos visto la función **print** visualiza un mensaje en la consola :

```
print('Hola mundo !')
```

Podemos introducir una **variable**, frase que contenga el texto, la función **time.sleep(segundos)** que hace una pausa, (para utilizar esta función se necesita importar la librería time con **import time** ) y dentro de un **bucle while** que se ejecuta mientras sea verdadero lo que le sigue, en este caso while True se ejecutará siempre:

```
import time

frase = "Hola mundo !!"

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

Aquí se utiliza

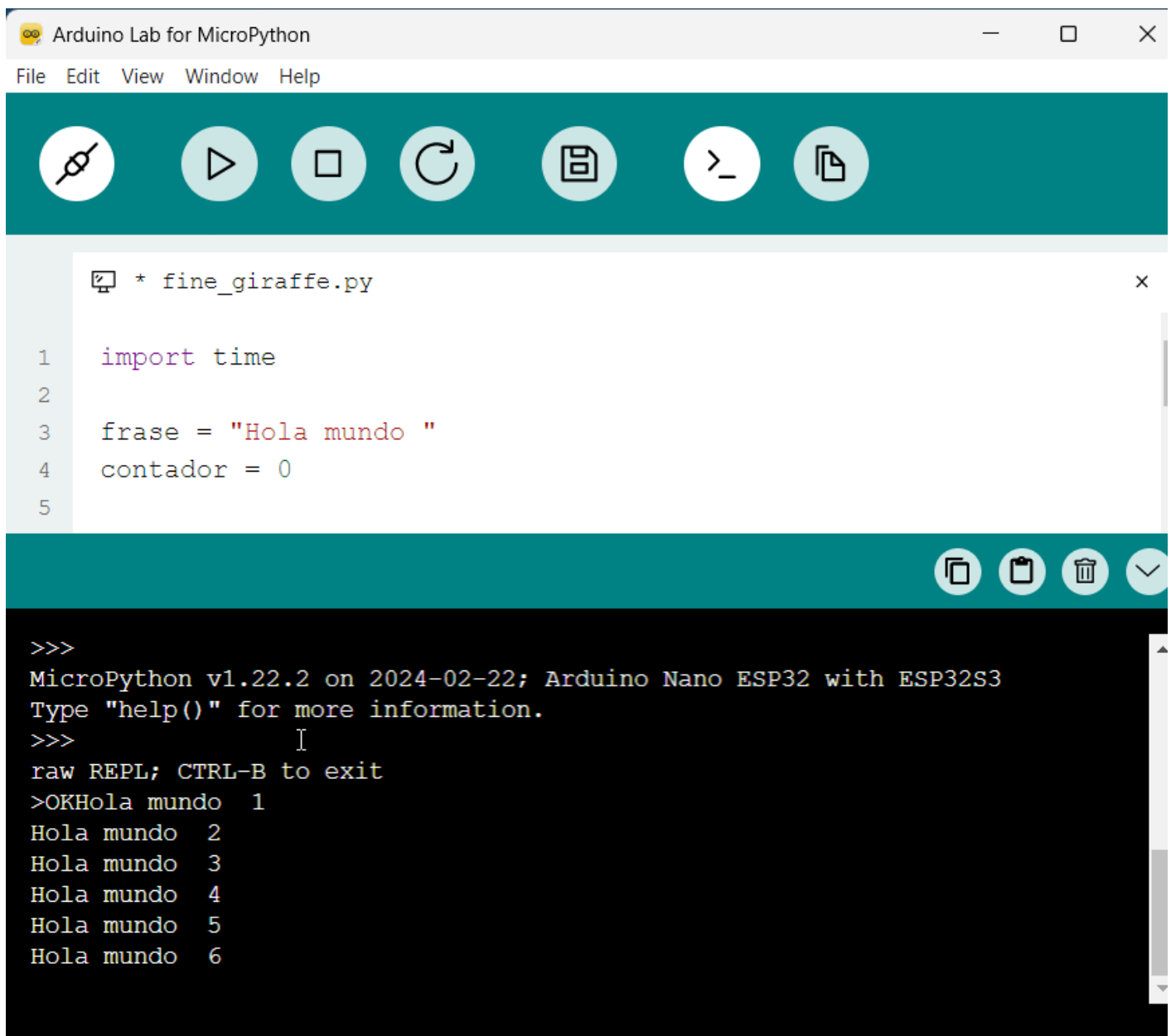
- una **función** con **def** una variable contador que en la función se declara **global** de esta manera se puede utilizar dentro de cualquier función del programa (en este caso el programa principal la funcion\_contar()).
- Vemos la típica operación de cuenta contador = contador + 1
- print visualiza dos cosas, la frase y el contador

```
import time

frase = "Hola mundo "
contador = 0
```

```
def funcion_contar():  
    global contador  
    contador = contador + 1  
  
while True:  
    funcion_contar()  
    print(frase, contador)  
    time.sleep(1)
```

El resultado:



The screenshot shows the Arduino Lab for MicroPython IDE. The top menu bar includes File, Edit, View, Window, and Help. Below the menu is a toolbar with icons for connecting, running, stopping, refreshing, saving, and opening files. The main editor displays a file named `* fine_giraffe.py` with the following code:

```
1 import time  
2  
3 frase = "Hola mundo "  
4 contador = 0  
5
```

Below the editor is a console window showing the execution output:

```
>>>  
MicroPython v1.22.2 on 2024-02-22; Arduino Nano ESP32 with ESP32S3  
Type "help()" for more information.  
>>>  
raw REPL; CTRL-B to exit  
>OKHola mundo 1  
Hola mundo 2  
Hola mundo 3  
Hola mundo 4  
Hola mundo 5  
Hola mundo 6
```

Aquí utilizamos el **condicional if** con su auxiliar **else** y la función **exit** para acabar el programa:

```

import time

frase = "Hola mundo "
contador = 0
maximo = 20

def funcion_contar():
    global contador
    contador = contador + 1

while True:
    funcion_contar()
    if contador > 20 :
        exit
    else :
        print(frase, contador)
        time.sleep(1)

```

Lo que provoca que a los 20 finalice

```

Hola mundo  16
Hola mundo  17
Hola mundo  18
Hola mundo  19
Hola mundo  20
Traceback (most recent call last):
  File "<stdin>", line 14, in <module>
NameError: name 'exit' isn't defined
>
MicroPython v1.22.2 on 2024-02-22; Arduino Nano ESP32 with ESP32S3
Type "help()" for more information.
>>>

```

Podemos usar en vez de variables numéricas, variables tipo **array** para los bucles :

```

Catedu = ['Javier', 'Santiago', 'Silvia', 'Berta', 'Cristina', 'Nacho', 'Arturo', 'Chefo', 'Vladi', 'Ruben', 'Pablo',
'JuanFran']

def printCatedus():
    for persona in Catedu:
        print(persona)

```

```
printCatedus()
```

```
MicroPython v1.22.2 on 2024-02-22; Arduino Nano ESP32 with ESP32S3
Type "help()" for more information.
>>>
raw REPL; CTRL-B to exit
>OKJavier
Santiago
Silvia
Berta
Cristina
Nacho
Arturo
Chefo
Vladi
Ruben
Pablo
JuanFran
>
MicroPython v1.22.2 on 2024-02-22; Arduino Nano ESP32 with ESP32S3
Type "help()" for more information.
>>>
```

---

Revision #2

Created 15 June 2024 07:22:27 by Javier Quintana

Updated 15 June 2024 08:10:02 by Javier Quintana