uPython para ESP32

Sergio Fco. Hernández Machuca

uPython

- Empleo de las características básicas del lenguaje en una plataforma ESP32.
- Diferencia entre los directorios del uC y de la PC
- Es importante tener en carpetas las aplicaciones
  - Carpetas comunes en el uC: libs
  - ¿Cómo pasar un archivo de un lugar a otro?
- ¿Dónde está el ‘shell’ del uC? En la parte inferior del editor.
- Primer programa, desde el ‘shell’
  - `# Ejemplo básico de programa en uPython`
  - `print('Hola, iniciamos')`

uPython

- ¿Dónde se encuentran las librerías que facilita uPython?
- https://docs.micropython.org/en/latest/library/index.html
- Son librerías que contienen recursos que están disponibles de manera directa cuando se instala el 'firmware', es importante conocerles.
- Existen librerías asociadas a CPython (extender funcionalidad) y otras a soportar las características del uC (bajo nivel).
- Para usar estas librerías se debe incorporar el módulo correspondiente a través del estatuto '`import`'

uPython

- Programa típico de un LED para parpadeo
  - Usar el LED interconstruido en el módulo, si existe

```
# --> Programa del parpadeo de un LED
from machine import Pin
from utime import sleep
# El LED interconstruido para el módulo DEV KIT1 está en la terminal '2'
led = Pin(2, Pin.OUT)           # Creación de objeto LED en una terminal,'salida'

# Estructura cíclica
while True:
          led.on()              # Encendido del LED
          sleep(0.5)            # Retardo de tiempo (en segundos)
          led.off()             # Apagado del LED
          sleep(0.5)            # Retardo de tiempo
```

uPython

- Módulo ‘os’. Permite acceder al sistema operativo del uC (bajo uPython) (https://docs.micropython.org/en/latest/library/os.html)
- Proporciona servicios para el manejo de archivos en una estructura (filesystem, que es un esquema de programación que se utiliza para organizar y buscar archivos en una partición de memoria).
- Revisar algunas funciones del módulo ‘os’, ensayarlas desde consola.
- Para verificar en cuál dispositivo y sistema nos encontramos:
  - `>>> import os`
  - `>>> print(os.uname())`
- Para ver el directorio:
  - `>>> print(os.listdir())`
- Si no se pone argumento, se leerá el directorio actual en donde se encuentra `.`

uPython

- Ejecución de un programa desde un archivi entro del uC.
  - `>>> execfile('ejemplo.py’)`
- Módulo ‘machine’ de uPython para ESP32. Funciones relacionadas con la circuitería. Es dependiente de la arquitectura del módulo, recursos y modelo.

Este ejemplo es para uCs en general, quizás no funcione para el ESP32:

```
import machine
from micropython import const

GPIOA = const(0x48000000)
GPIO_BSRR = const(0x18)
GPIO_IDR = const(0x10)

# set PA2 high
machine.mem32[GPIOA + GPIO_BSRR] = 1 << 2

# read PA3
value = (machine.mem32[GPIOA + GPIO_IDR] >> 3) & 1
```

uPython

- Programa del parpadeo de un LED, ejercicios:

```
# Programa del parpadeo de un LED
from machine import Pin
from utime import sleep

led = Pin(21, Pin.OUT)    # Creación de un objeto LED en una terminal, una 'salida'

# Estructura cíclica
while True:
        led.on()          # Encendido del LED
        sleep(0.5)        # Retardo de tiempo (en segundos)
        led.off()         # Apagado del LED
        sleep(0.5)        # Retardo de tiempo
```

uPython

- Organización del código de una aplicación en uPython
- Dividir en módulos, o segmentos de código, las aplicaciones que se ensayen en uPython
- 1.- Crear un módulo que contenga las acciones (funciones) que se ejecutarán.
- 2.- Agregar un módulo que contenga la ejecución, invocación o llamado de las funciones o acciones, que se ejecutarán.

uPython

```
# Código para el archivo ‘uso_operaciones.py’

# Importación del módulo que contiene las acciones
import aritmetica

# Datos a emplear:
numero1 = 1.5
numero2 = 57

# Invocación de las acciones asociadas a los datos:
aritmetica.suma(numero1, numero2)
aritmetica.resta(numero1, numero2)
```

uPython

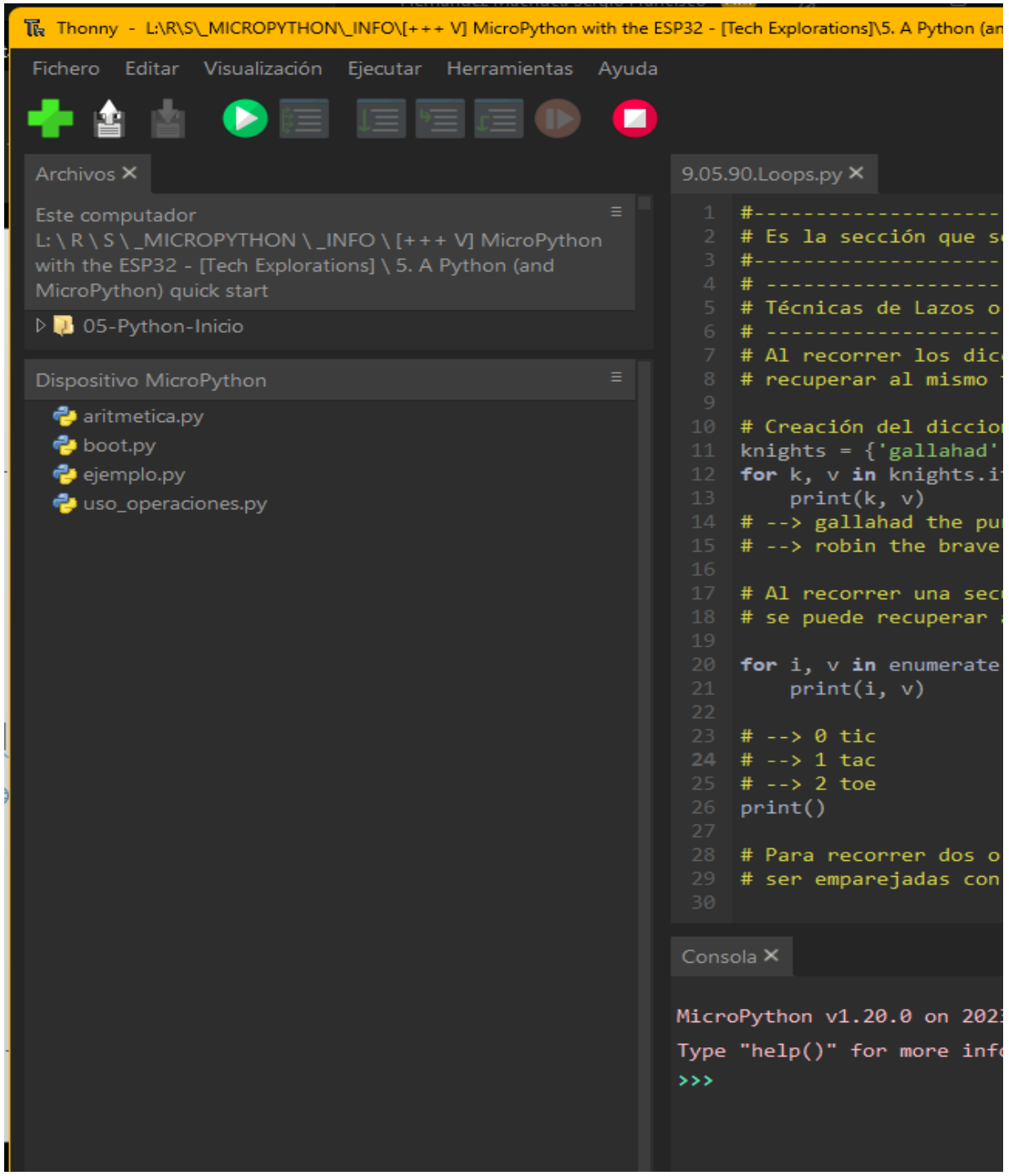
```
# Código para el módulo ‘aritmetica.py’

# Función que define una acción o actividad:
def suma(numero1, numero2):
        # Muestra el resultado de la suma:
        print('{0} + {1} = {2}'.format(numero1, numero2, numero1+numero2))

def resta(numero1, numero2):
        # Muestra el resultado de la suma:
        print('{0} - {1} = {2}'.format(numero1, numero2, numero1-numero2))
```

uPython

Aquí está el sistema de archivos del uC.



uPython

- ¿Cómo ejecutar un programa en la reiniciación? (archivo 'boot')
- Si existe, el archivo 'boot.py' en el directorio principal del uC es el que se ejecutará cada vez que ocurra una condición de ‘Reset’.
- Se puede copiar el código a ejecutarse cada vez que se encienda el módulo al archivo 'boot.py'. Entonces, si el módulo se alimenta por batería y / o se reinicia, la ejecución del sistema iniciará a partir de lo que se encuentre en el archivo 'boot.py'.
- También se puede usar, en el archivo 'boot.py', una función 'import’ del módulo en donde se encuentra el programa principal, aprovechando que ya se construyó.