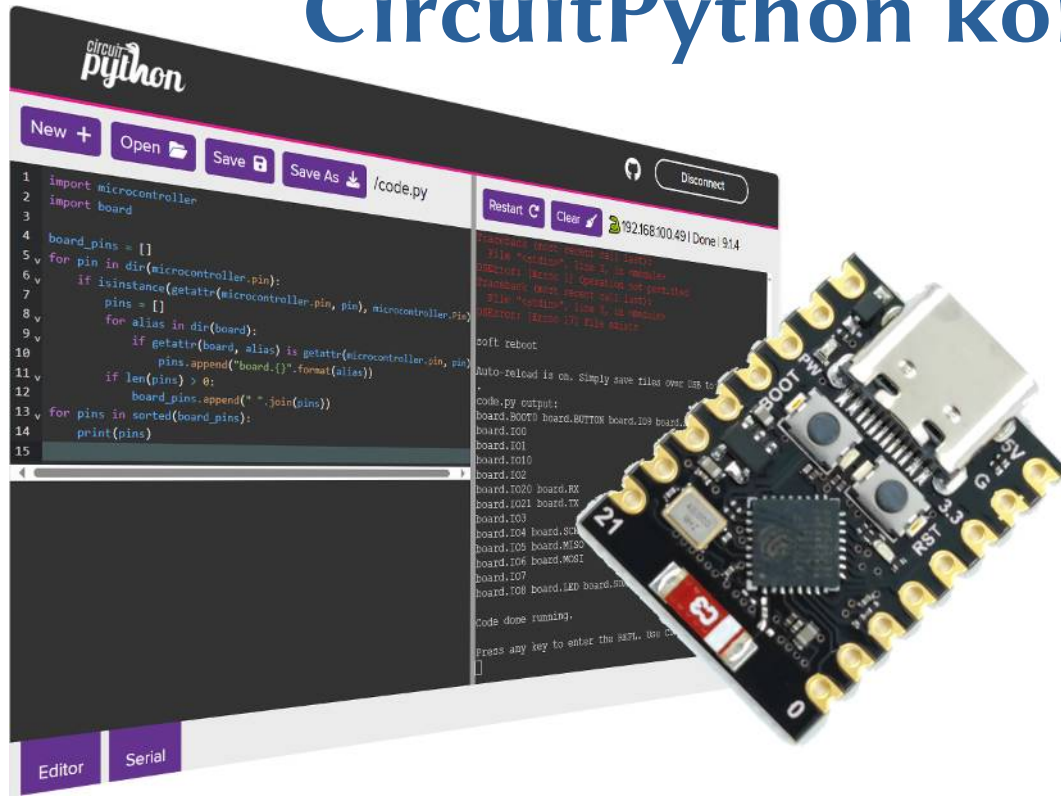


ESP32-C3 mikrovezérlők programozása CircuitPython környezetben



1. CircuitPython és az ESP32-C3 – a kezdő lépések

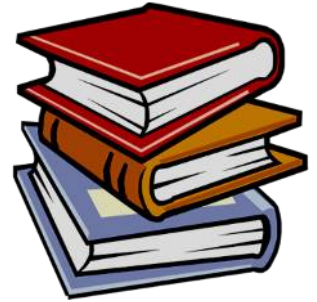
Felhasznált és ajánlott irodalom

❖ Python:

- Mark Pilgrim/Kelemen Gábor: [Ugorj fejest a Python 3-ba!](#)

❖ CircuitPython:

- Adafruit: <https://circuitpython.org/downloads>
- Learn Adafruit: [Welcome to CircuitPython](#)
- Learn Adafruit: [CircuitPython Essentials](#)
- Adafruit: [Adafruit CircuitPython API Reference](#)
- Adafruit: [github.com/adafruit/Adafruit CircuitPython Bundle](https://github.com/adafruit/Adafruit-CircuitPython-Bundle)



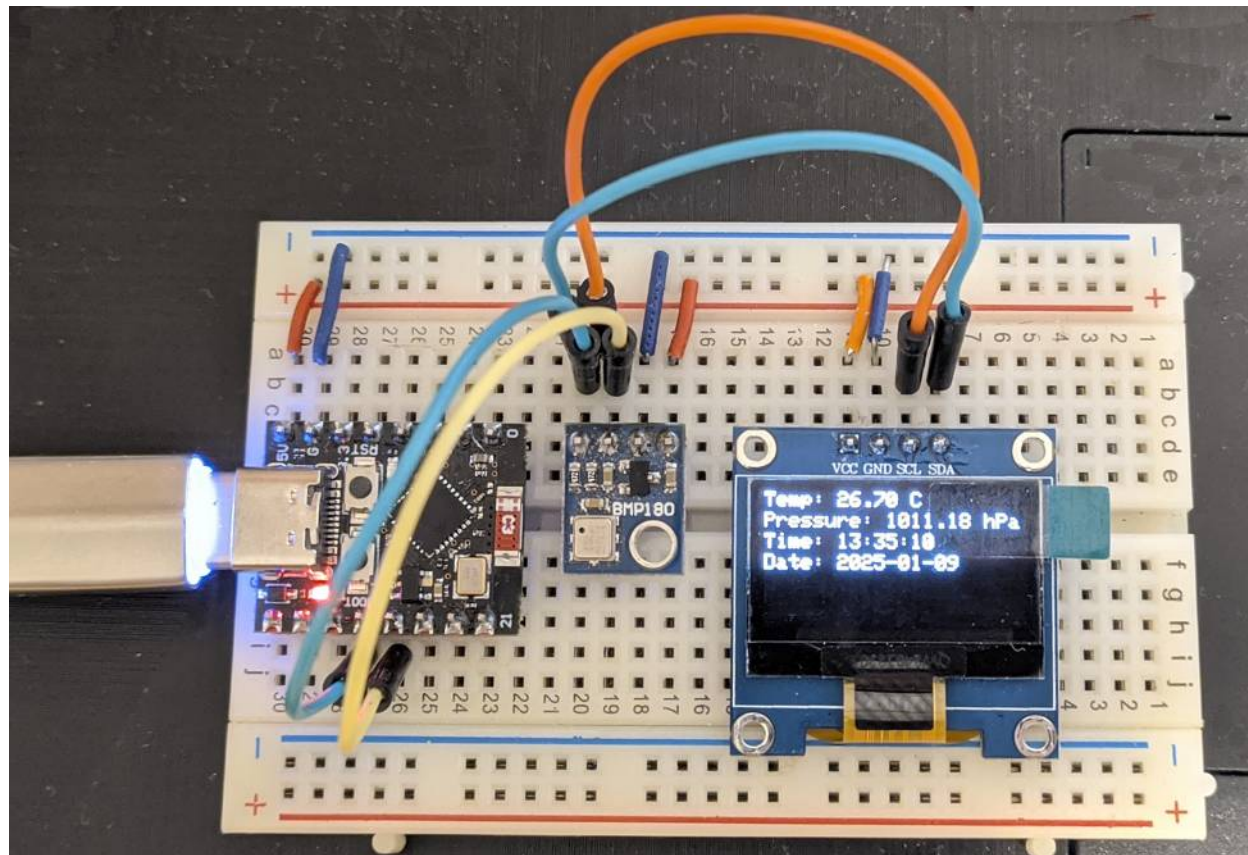
❖ Online eszközök és támogatás:

- Learn Adafruit: [CircuitPython on ESP32 Quick Start](#)
- Adafruit: [Adafruit Web Serial ESPTool](#)
- Adafruit: [CircuitPython Code Editor](#)



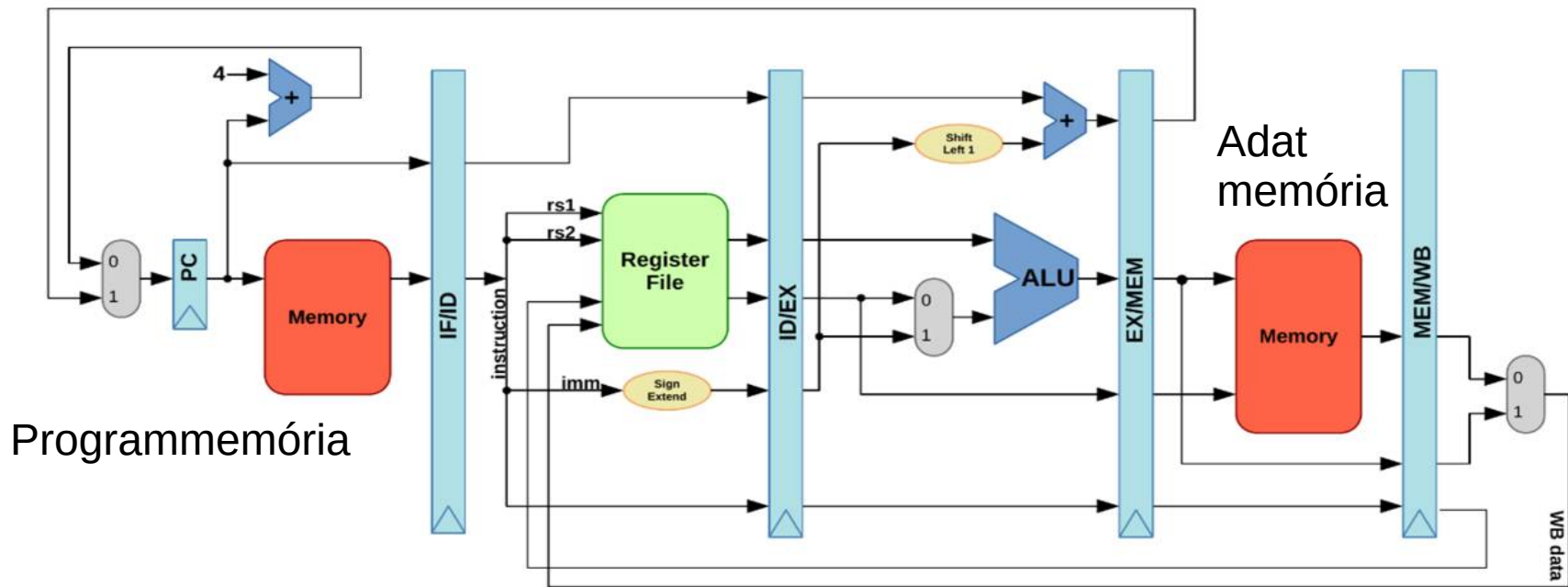
Miért ezekre esett a választás?

- ❖ Az **ESP32-C3** egy olcsó, de nagytudású mikrovezérlő (sok memóriával, WiFi és Bluetooth képességekkel)
- ❖ A **CircuitPython** pedig egy magasszintű, könnyen elsajátítható programnyelv, fejlett támogatással, sokféle célra és eszközhöz



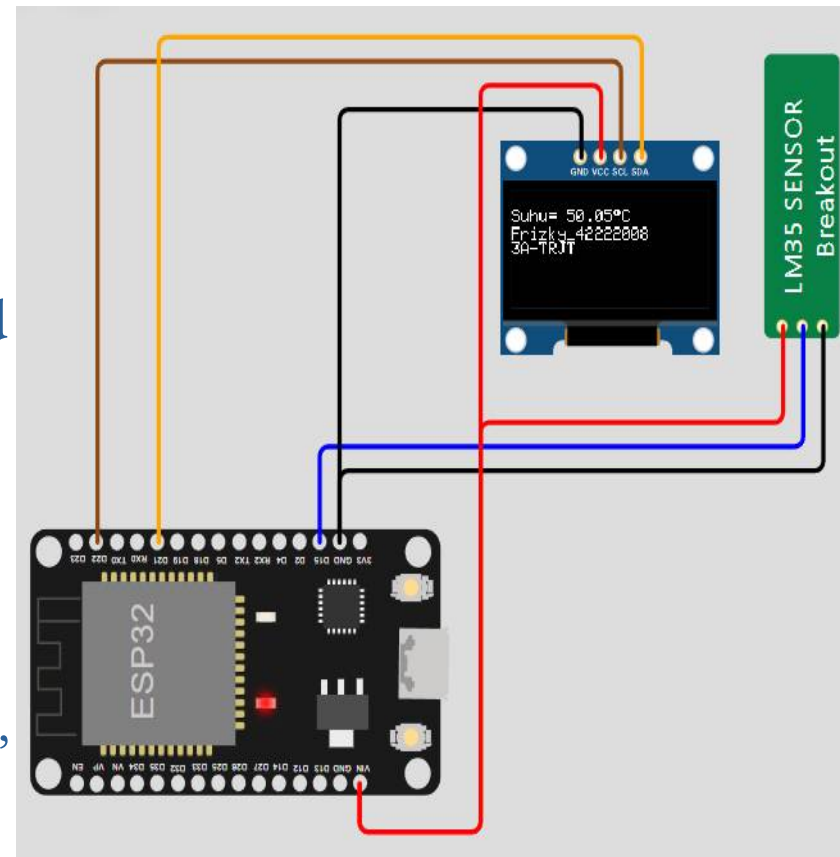
A programozható eszközök működése

- ❖ A CPU (Central Processing Unit) főbb részei a PC programszámláló, a programmemória, az utasításdekóder, az aritmetikai logikai egység (ALU), a regiszterkészlet, az adatmemória



Fizikai programozás

- ❖ A fizikai programozás (vagy fizikai számítástechnika) olyan interaktív rendszerek létrehozását jelenti hardverek és szoftverek segítségével, melyek képesek érzékelni a világban létrejövő jeleket és reagálni is tudnak rá
- ❖ A mi esetünkben ez saját készítésű csináld-magad projektek vagy alkotások megnevezésére szolgál, melyek mikrovezérlők (vezérlési feladatokra optimalizált extra kicsi számítógépek) be- és kimeneteire kötött különféle digitális és analóg érzékelők segítségével vezérelnek elektromechanikus eszközöket (jelfogó, motorok), világító vagy egyéb eszközöket, kijelzőket, de akár szoftvereket is



Mi az a CircuitPython?

- ❖ **CircuitPython:** programozási nyelv, arra tervezve, hogy segítse a tanulás és a kísérletezés folyamatát a mikrovezérlő kártyákkal
- ❖ Nem szükséges bonyolult keresztfordító rendszereket telepíteni a számítógépünkre. Amint csatlakoztattuk a kártyánkat, csak egy szövegszerkesztőre van szükség. Ilyen egyszerű...
- ❖ A **CircuitPython** a **Python** nyelven alapul. A korlátozott hardver erőforrások nyilvánvalóan korlátozott lehetőségeket biztosítanak, ugyanakkor a **CircuitPython** többletet is nyújt: hardvertámogatást a mikrovezérlő kártya perifériáinak elérésére és kezelésére, mellyel ún. „*fizikai programozást*” valósíthatunk meg.
- ❖ **Fizikai programozás:** olyan interaktív rendszerek létrehozását jelenti hardverek és szoftverek segítségével, melyek képesek érzékelni a világban létrejövő jeleket és reagálni is tudnak rá, azaz a programjaink hatására nemcsak a képernyőn történik valami, hanem a fizikai valóságban is...

A támogatott kártyák

- ❖ A támogatott kártyák száma e sorok írásakor: **571** (számuk napról napra növekszik!)
- ❖ Csak példa gyanánt néhány ismertebb típus:
Raspberry Pi Pico, Circuit Playground Express, Feather M4 express, **ESP32 Doit Devkit**, Nano RP2040 Connect, Metro ESP32-S2, nRF52840 Dongle, Feather STM32F405 Express, **STM32F411CE blackpill**, Arduino Nano 33 IOT, STM32H743 Nucleo, BBC micro:bit v2, Pyboard, Raspberry Pi 4 model B és még sokan mások...
- ❖ A CircuitPython aktuális változatának letöltése innen: <https://circuitpython.org/downloads> (a kártya kiválasztása és rákattintás után a felbukkanó lapon a jobb felső sarokban elérhető a letöltési link)
- ❖ Mi a **Maker Go ESP32-C3 Super mini** kártyát fogjuk használni, a **CircuitPython** legfrissebb kiadásával (lásd: circuitpython.org/board/makergo_esp32c3_supermini/)

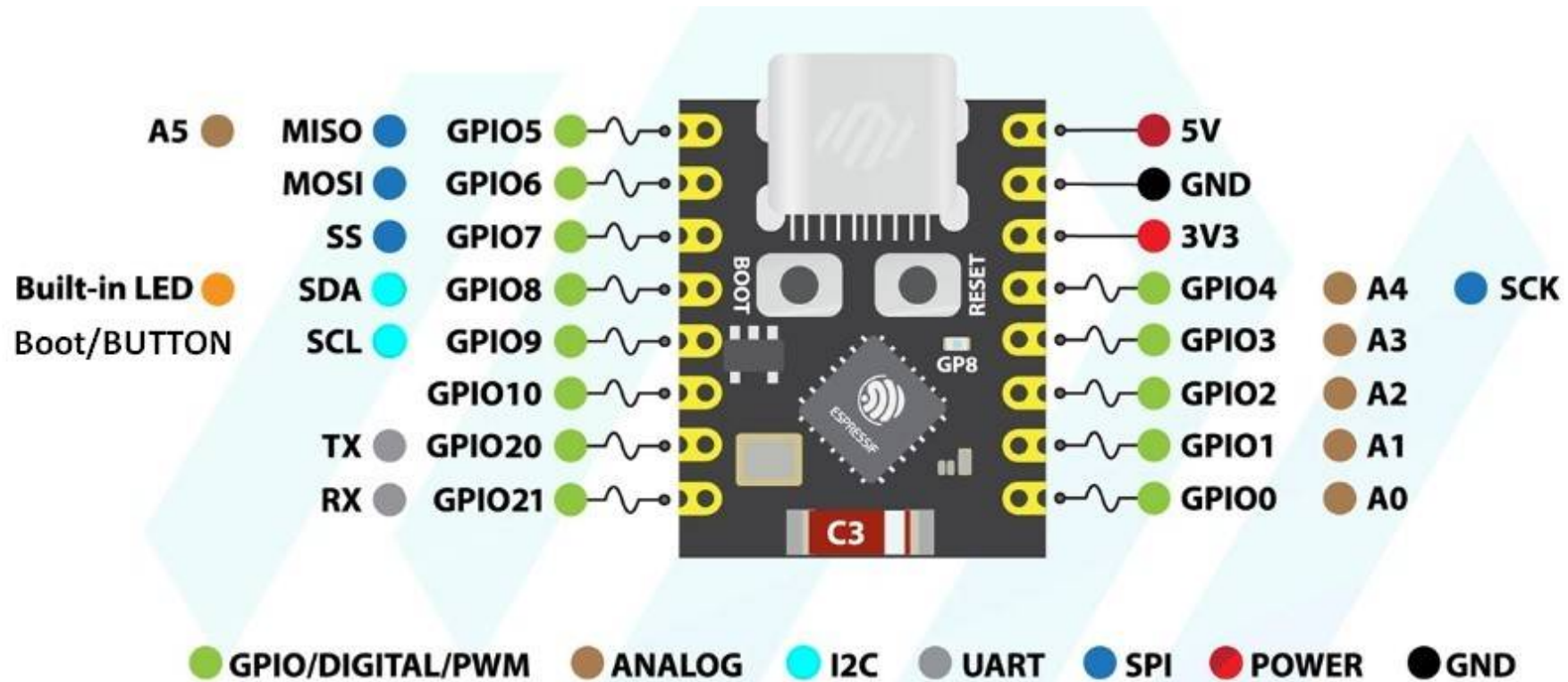


Az ESP32-C3 Super Mini kártya

- ❖ A **Maker Go ESP32-C3 Super Mini** egy kompakt fejlesztői kártya
- ❖ Mikrovezérlő: **ESP32-C3**, 32 bites **RISC-V** architektúra
- ❖ Órajel: Akár 160 MHz
- ❖ Memória: 400 KB SRAM, 384 KB ROM
- ❖ Tároló: 4 MB Flash memória
- ❖ Wi-Fi: 802.11 b/g/n (2.4 GHz)
- ❖ Bluetooth: Bluetooth 5.0 LE
- ❖ GPIO: Több mint 20 általános célú bemenet/kimenet (GPIO), melyek közül 13 van kivezetve
- ❖ Interfészek: SPI, I2C, UART, ADC, PWM



Az ESP32 C3 Super Mini kártya kivezetései



ESP32 C3 Super Mini

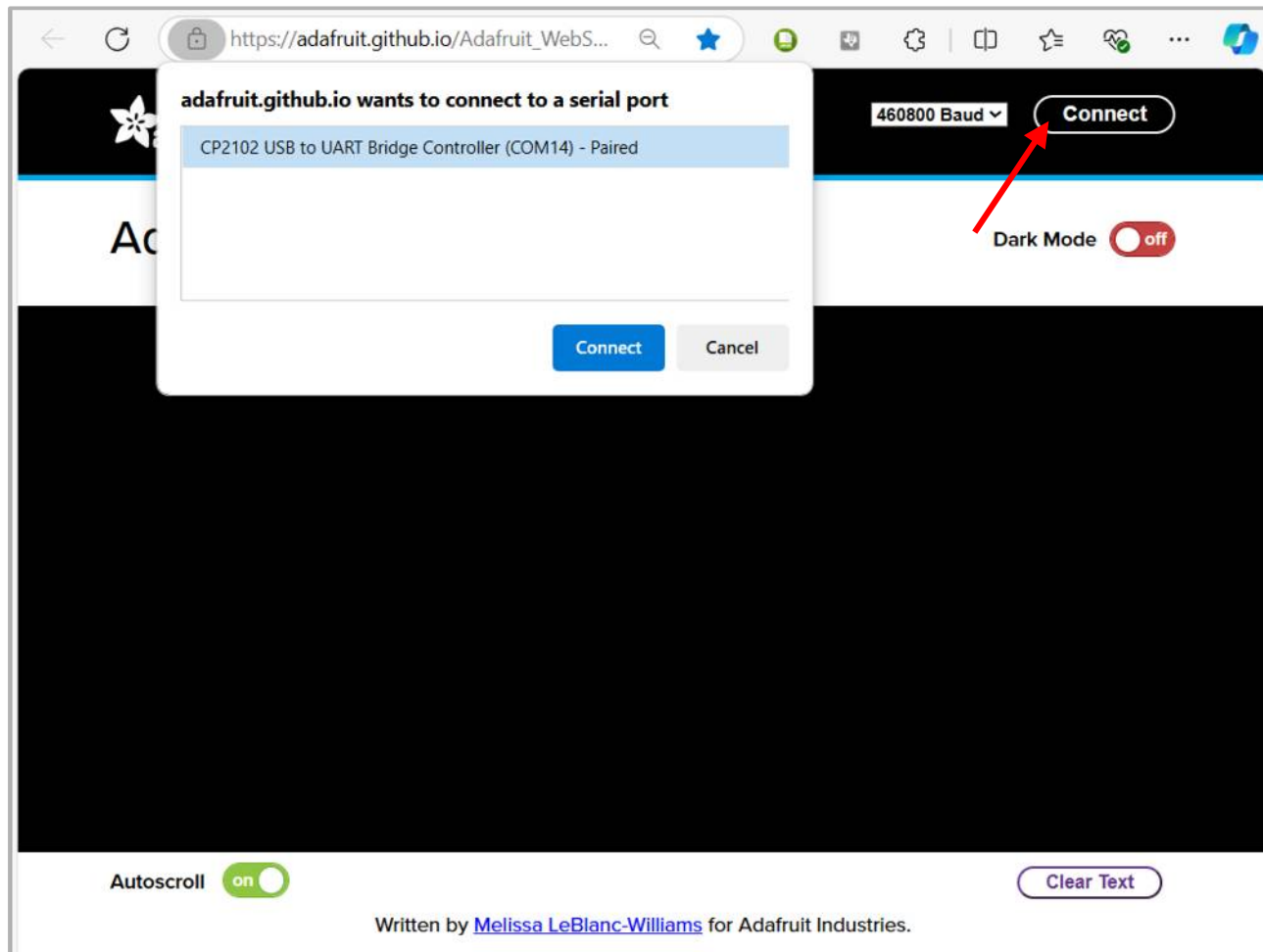
Megjegyzés: Az A5 analóg bemenet (ADC2) nem használható, ha a WiFi használatban van!

A CircuitPython firmware telepítése

- ❖ Mivel a kártyánk gyárilag nem tartalmazza a **CircuitPython** firmware-t, így bele kell töltenünk (csak egyszer kell végrehajtani)
- ❖ Az **ESP32 Doit Devkit** kártyához a CircuitPython 9.2.1 változatát innen töltöttük le: circuitpython.org/board/makergo_esp32c3_supermini/
- ❖ A felprogramozást a mikrovezérlő gyárilag beégetett bootloadere segítségével végezhető el, ehhez vagy a **Web Serial ESPTool**, vagy a Pythonban írt parancssori **ESPTool** letöltő programot használjuk
 - A **Web Serial ESPTool** használata egyszerűbb, de **Chrome**, vagy más, Chrome alapú (*Edge, Opera*) böngésző kell hozzá, amiben engedélyezhető a soros portok elérése (a <chrome://flags> URL megnyitása után az **Experimental Web Platform features** opció legyen **Enabled**-re állítva)
 - Az **ESPTool.py** letöltőt **Python** környezetbe telepíteni kell a `pip install esptool` paranccsal

Kapcsolódás Web Serial ESPTool-hoz

- ❖ A böngészővel lépünk a Web Serial ESPTool nyitólapjára!
- ❖ Csatlakoztassuk BOOT módban a kártyánkat és tartsuk a BOOT gombot lenyomva!
- ❖ Kattintsunk a böngészőben a **CONNECT** gombra!
- ❖ Válasszuk ki az eszközünkhöz tartozó soros portot és folytassuk a kék **CONNECT** gombbal



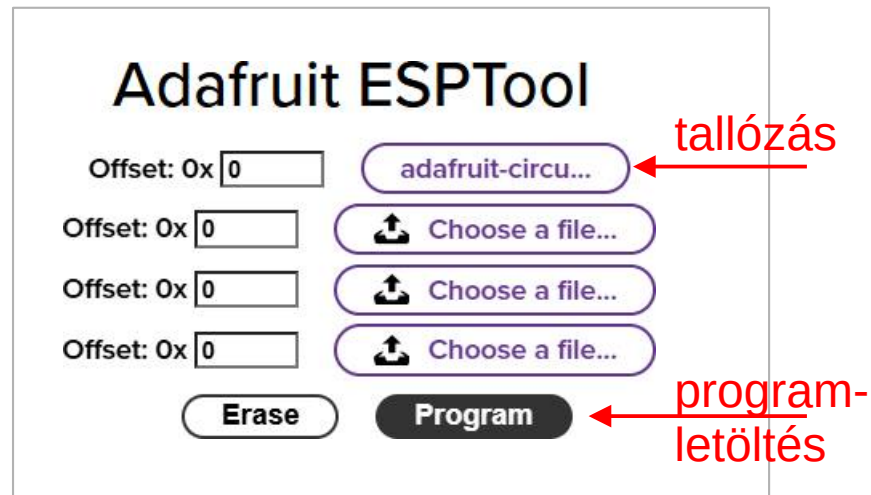
A Web Serial ESPTool használata

- ❖ Sikeres kapcsolódás esetén az ábrán látható kiírások jelennek meg a képernyőn
- ❖ Az ERASE gomb segítségével törölhetjük a flash memória tartalmát
- ❖ Az DISCONNECT gomb segítségével megszakíthatjuk a kártyával a soros porti kapcsolatot (most még ne nyomjuk!)

The screenshot displays the Adafuit ESPTool web interface. At the top, the Adafuit logo is on the left, and the baud rate is set to 460800 with a Disconnect button on the right. Below the header, the title 'Adafuit ESPTool' is centered. To the right of the title is a 'Dark Mode' toggle set to 'off'. The main area contains four 'Offset: 0x' input fields, each followed by a 'Choose a file...' button. Below these is an 'Erase' button (highlighted with a red arrow) and a 'Program' button. At the bottom, a terminal window shows the following output: 'Connecting...', 'Connected successfully.', 'Try hard reset.', 'Chip type ESP32', 'Connected to ESP32', 'MAC Address: 24:6F:28:A9:A8:38', 'Uploading stub...', 'Running stub...', 'Stub is now running...'. Below the terminal is an 'Autoscroll on' toggle and a 'Clear Text' button. At the very bottom, it says 'Written by [Melissa LeBlanc-Williams](#) for Adafuit Industries.'

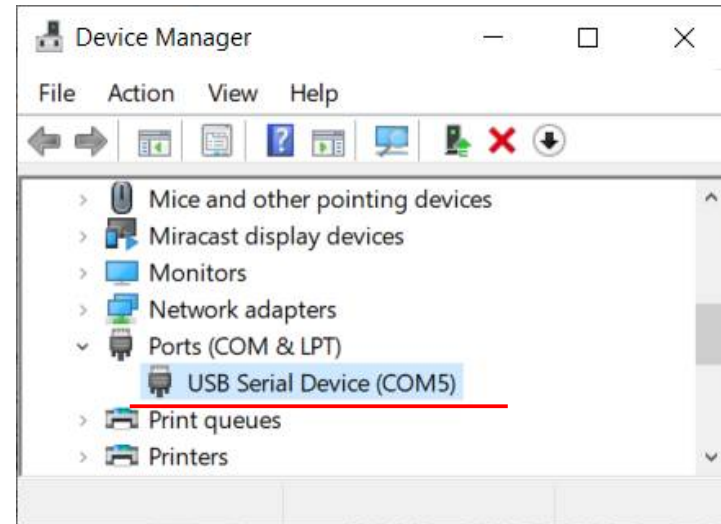
A Web Serial ESPTool használata

- ❖ A jobb felső gombra kattintva tallózzuk be az előzőleg letöltött CircuitPython firmware-t (adafruit-circuitpython-makergo_esp32c3_supermini-en_US-9.2.1.bin)
- ❖ Ügyeljünk arra, hogy a betöltési kezdőcím **0x0** legyen!
- ❖ Az **ERASE** gombra kattintva a flash memória teljes tartalmát törölhetjük
- ❖ A **PROGRAM** gombra kattintva csak a felülírt flash memóriaterület törlődik, majd beírásra kerül a CircuitPython firmware
- ❖ A programletöltés után a **DISCONNECT** gombra kattintva szakítsuk meg a kapcsolatot, majd a kártya resetelése után kapcsolódjunk egy terminálablakhoz



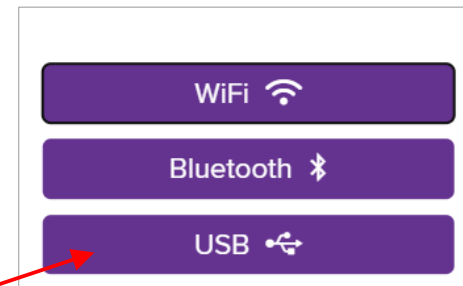
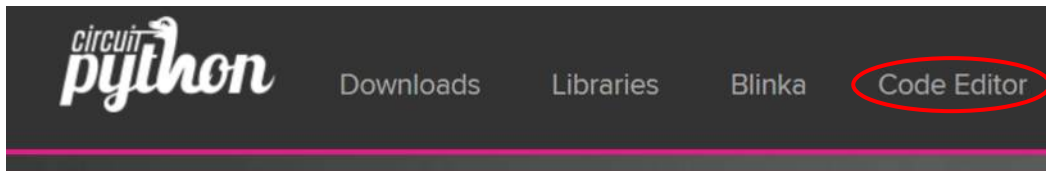
ESPTool.py – a parancssori letöltőprogram

- ❖ Programletöltésre **egy másik lehetőség** az offline, parancssori megoldás, az **esptool.py** (ehhez telepíteni kell a Python-t is, ha még nincs telepítve)
- ❖ A telepítéséhez adjuk ki az alábbi parancsot:
`pip install esptool`
- ❖ A **Device Manager**-ben keressük meg a kártyánkhoz tartozó port nevét (pl. **COM5**)
- ❖ A flash memória teljes törlése:
`esptool --port COM5 erase_flash`
- ❖ A **CircuitPython firmware** letöltése:
`esptool -p COM5 write_flash -z 0x0 firmware.bin`
ahol *firmware.bin* helyére az előzőleg letöltött firmware nevét, ill. elérési útvonalát kell megadni
- ❖ A parancsok kiadása után szükség lehet a **BOOT** gomb lenyomására!

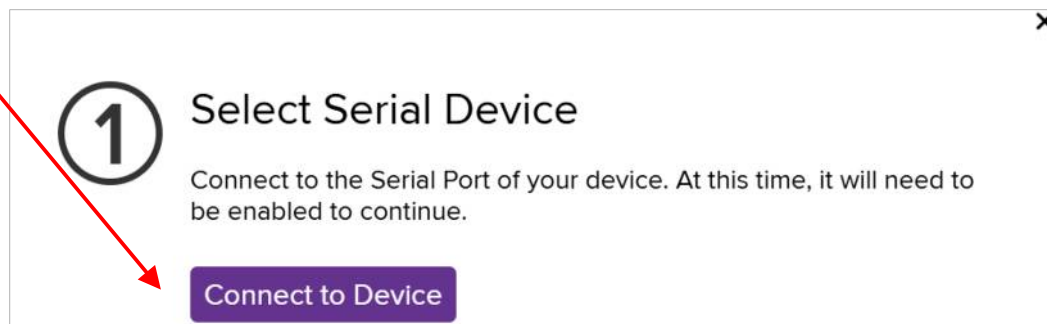


Webes munkakörnyezet

- ❖ A circuitpython.org honlapon a **Code Editor** gombra kattintva elindíthatjuk a webes munkakörnyezetet

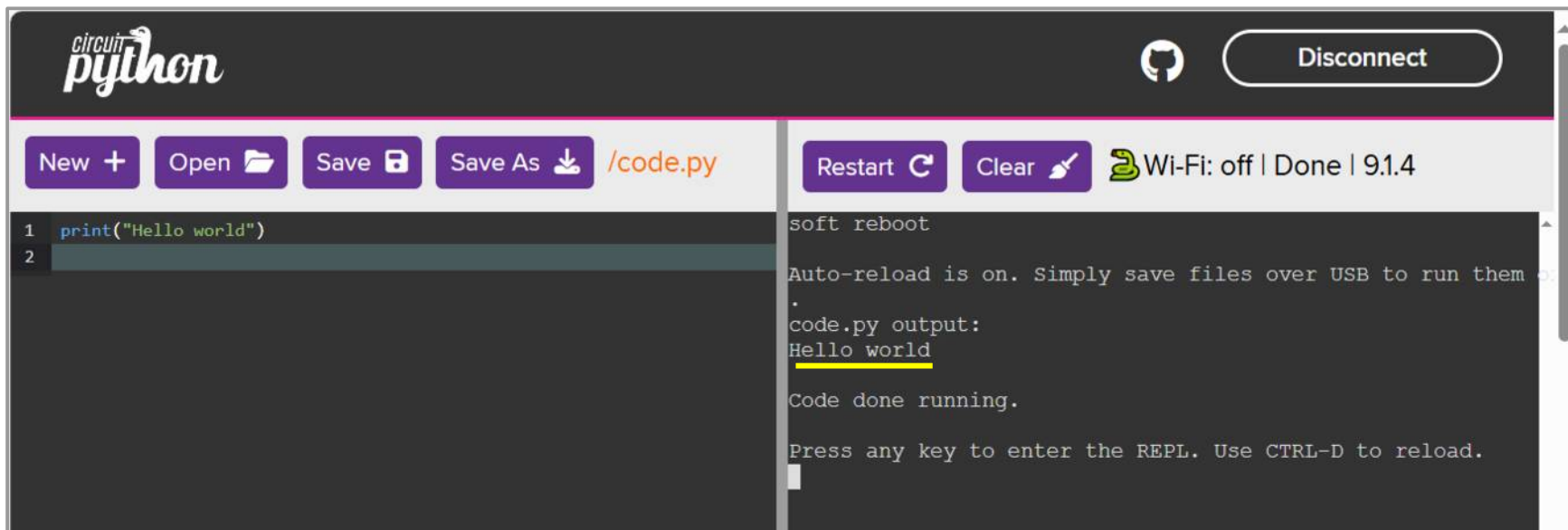


- ❖ A kapcsolódáshoz az USB opciót kell választanunk
- ❖ A **Connect to Device** gombra kattintva a szokásos módon ki kell választani a **Web Serial** kapcsolat számára a kártyához tartozó soros portot
- ❖ A kék **Connect** gombra kattintva megnyílik a kódszerkesztő és fájlkezelő ablak (**USB port híján nincs CIRCUITPY: meghajtó!!!**)



Helloworld.py

- ❖ Írjuk be a `print("Hello World!")` parancsot, vagy nyissuk meg a `code.py` állományt az **Open** gombra kattintva!
- ❖ A megszerkesztett programot `code.py` néven mentjük el
- ❖ A **Restart**, vagy **Save+Run** gombra kattintva lefut a program



The screenshot shows the CircuitPython IDE interface. At the top left is the 'circuitpython' logo. On the top right, there is a GitHub icon and a 'Disconnect' button. Below the logo, there is a toolbar with buttons for 'New +', 'Open', 'Save', and 'Save As', followed by the filename '/code.py'. To the right of the toolbar are buttons for 'Restart' and 'Clear', and a status bar showing 'Wi-Fi: off | Done | 91.4'. The main area is split into two panes. The left pane is a code editor showing two lines of code: '1 print("Hello world")' and '2'. The right pane is the REPL (Read-Eval-Print Loop) showing the output of the code: 'soft reboot', 'Auto-reload is on. Simply save files over USB to run them', 'code.py output:', 'Hello world' (underlined), and 'Code done running.'. At the bottom of the REPL, it says 'Press any key to enter the REPL. Use CTRL-D to reload.'

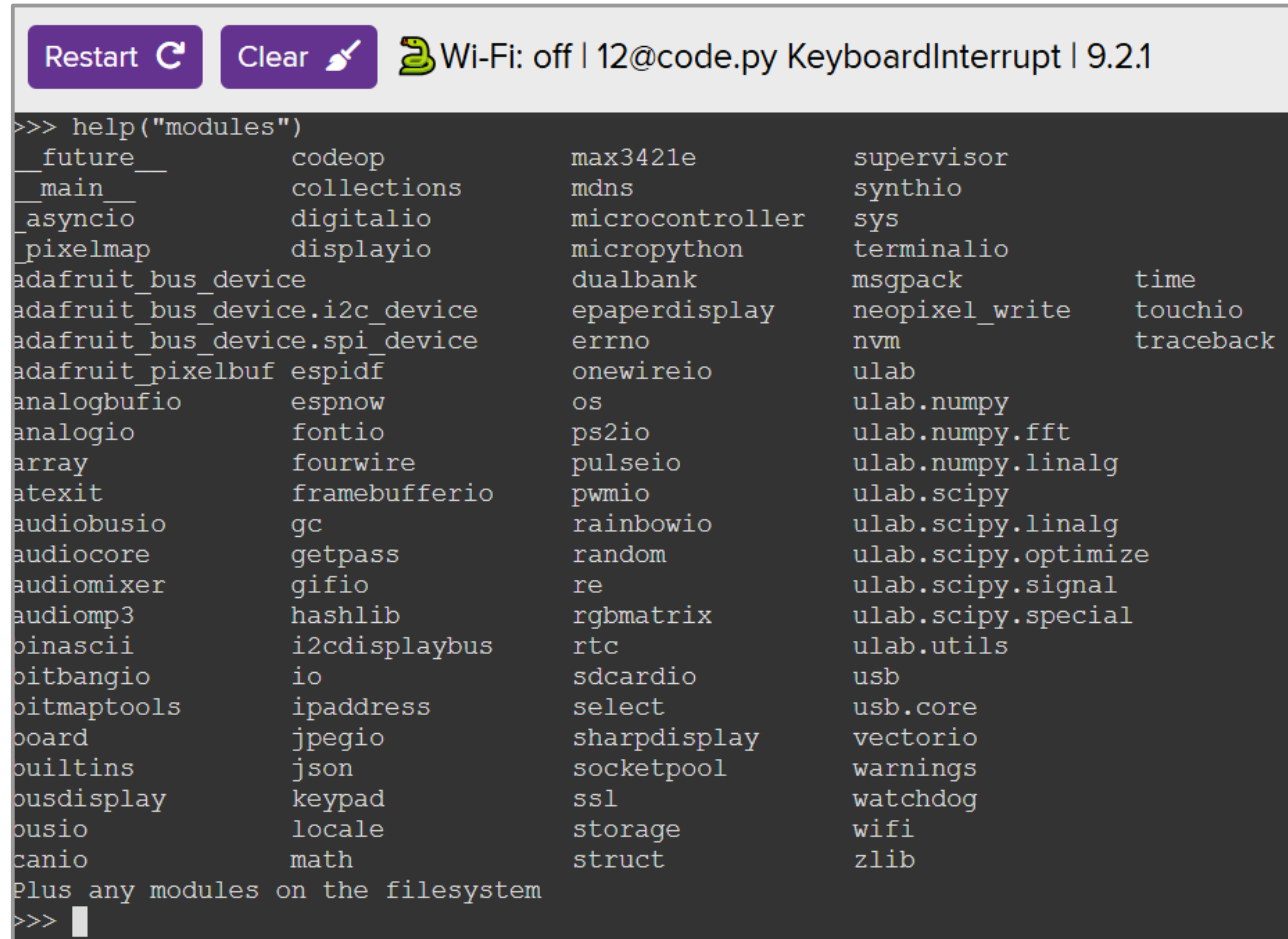
A REPL mód használata

❖ A Serial (soros porti) ablakban **Enter** gomb nyomására beléphetünk az interaktív REPL módba (Read-Eval-Print Loop)

❖ Adjuk ki például az alábbi parancsokat

```
>>> print(5 + 8)  
13
```

```
>>> help(„modules”)  
__future__  
__main__  
_asyncio  
adafruit_bus_device  
...
```



The screenshot shows a Python REPL terminal window with a dark background. At the top, there are buttons for 'Restart' and 'Clear', and a status bar indicating 'Wi-Fi: off | 12@code.py KeyboardInterrupt | 9.2.1'. The terminal prompt is '>>>', and the user has entered 'help("modules")'. The output is a list of modules arranged in four columns. The first column includes modules like __future__, __main__, _asyncio, and adafruit_bus_device. The second column includes codeop, collections, digitalio, displayio, and others. The third column includes max3421e, mdns, microcontroller, and others. The fourth column includes supervisor, synthio, sys, terminalio, and others. The list ends with 'Plus any modules on the filesystem' and a new prompt '>>>'.

```
>>> help("modules")  
__future__      codeop           max3421e        supervisor  
__main__        collections     mdns            synthio  
_asyncio        digitalio       microcontroller sys  
_pixelmap       displayio      micropython    terminalio  
adafruit_bus_device  dualbank       msgpack         time  
adafruit_bus_device.i2c_device  epaperdisplay neopixel_write touchio  
adafruit_bus_device.spi_device  errno          nvmm            traceback  
adafruit_pixelbuf  esp8266        onewireio      ulab  
analogbufio       espnow         os              ulab.numpy  
analogio          fontio         ps2io           ulab.numpy.fft  
array             fourwire       pulseio         ulab.numpy.linalg  
atexit            framebufferio  pwmio           ulab.scipy  
audiobusio        gc             rainbowio       ulab.scipy.linalg  
audiocore         getpass        random          ulab.scipy.optimize  
audiomixer        gifio         re              ulab.scipy.signal  
audiomp3          hashlib        rgbmatrix       ulab.scipy.special  
oinascii          i2cdisplaybus rtc             ulab.utils  
oitbangio         io             sdcardio        usb  
oitmptools        ipaddress     select          usb.core  
oboard           jpegio        sharpdisplay   vectorio  
ouiltins          json          socketpool     warnings  
ousdisplay        keypad        ssl            watchdog  
ousio             locale        storage         wifi  
ocanio            math          struct          zlib  
Plus any modules on the filesystem  
>>>
```

A REPL mód használata

❖ Az első LED villogtató program megírásához derítsük ki, hogy hogyan hívják az egyes kivezetéseket!

❖ Adjuk ki az alábbi parancsokat

```
>>> import board
```

```
>>> dir(board)
```

```
['_class__', '__name__', 'BOOT0', 'BUTTON', 'I2C', 'I00', 'I01',  
'I010', 'I02', 'I020', 'I021', 'I03', 'I04', 'I05', 'I06', 'I07',  
'I08', 'I09', 'LED', 'MISO', 'MOSI', 'RX', 'SCK', 'SCL', 'SDA',  
'SPI', 'TX', 'UART', '__dict__', 'board_id']
```

```
>>> print(board.LED)
```

```
board.I08
```

❖ A beépített LED tehát `board.I08`, vagy `board.LED` néven érhető el

A beépített LED villogtatása: ledblink.py

- ❖ Írjuk be az alábbi programot!
- ❖ A **Save as** gombbal töltsük le a mikrovezérlőre **code.py** néven!
- ❖ **Restart**, vagy **Ctrl-D** nyomására a program elindul és végtelen ciklusban kétmásodpercenként felvillantja a beépített LED-et
- ❖ A **time.sleep()** függvény paraméterét másodpercben kell megadni
- ❖ **Megjegyzés:** a beépített LED katódját vezéreljük, ezért a **kimenet lehúzásakor** világít a LED!

```
import board
import digitalio
import time

led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT

while True:
    led.value = False
    time.sleep(0.25)
    led.value = True
    time.sleep(1.75)
```

ledblink.py

A kártya kivezetéseinek listázása: pin_list.py

- ❖ Listázzuk ki a kivezetések neveit és másodlagos neveit!

```
import microcontroller
import board

board_pins = []
for pin in dir(microcontroller.pin):
    if isinstance(getattr(microcontroller.pin, pin), microcontroller.Pin):
        pins = []
        for alias in dir(board):
            if getattr(board, alias) is getattr(microcontroller.pin, pin):
                pins.append("board.{}".format(alias))
        if len(pins) > 0:
            board_pins.append(" ".join(pins))
for pins in sorted(board_pins):
    print(pins)
```

pin_list.py

code.py output:

```
board.BOOTH0 board.BUTTON \
board.IO9 board.SCL
board.IO0
board.IO1
board.IO10
board.IO2
board.IO20 board.RX
board.IO21 board.TX
board.IO3
board.IO4 board.SCK
board.IO5 board.MISO
board.IO6 board.MOSI
board.IO7
board.IO8 board.LED board.SDA
```

- ❖ A **GPIO n** és **D n** jelölések számozása megegyezik
- ❖ A **GPIO6 – GPIO11** kivezetések a flash memória kezelésére fenntartottak, ezért hiányoznak a listáról

Digitális ki- és bemenetek kezelése

- ❖ A digitális ki- és bemenetek kezelése a DigitalInOut objektumok segítségével történik, például:

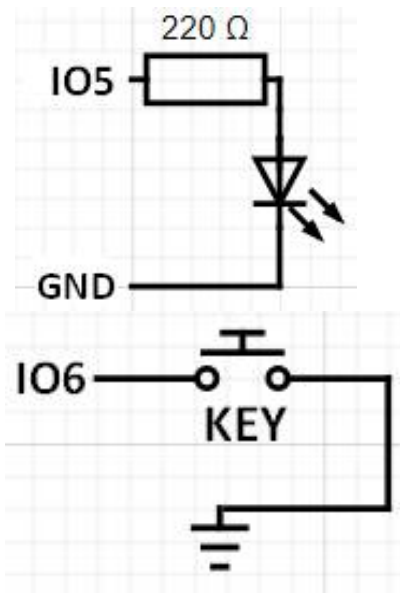
- ❖ **Kimenet konfigurálása:**

```
import digitalio, board
led = digitalio.DigitalInOut(board.IO5)
led.direction = digitalio.Direction.OUTPUT
led.drive_mode = digitalio.DriveMode.OPEN_DRAIN
```

- ❖ **Bemenet konfigurálása:**

```
import digitalio, board
button = digitalio.DigitalInOut(board.IO6)
button.direction = digitalio.Direction.INPUT
button.pull = digitalio.Pull.UP
```

- ❖ **Megjegyzés:** nem kell ilyen hosszú litániákat írni, ha így importálunk:
`from digitalio import DigitalInOut, Direction, DriveMode, Pull` vagy
`from digitalio import *`



LED vezérlése nyomógommbal: ledswitch.py

- ❖ Kapcsolgassuk egy LED-et (IO5) az IO6 és a GND közé kötött nyomógomb segítségével!

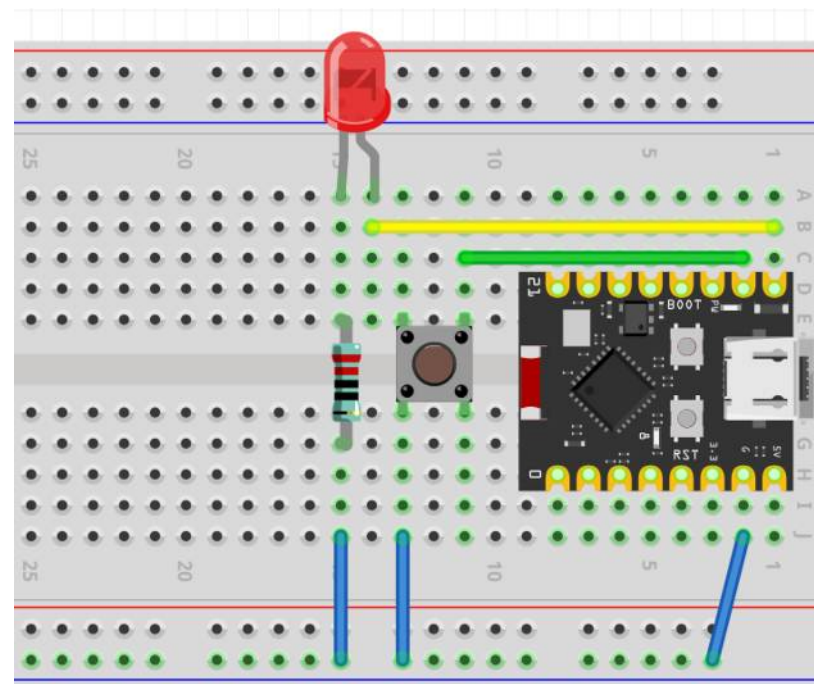
ledswitch.py

```
from digitalio import *           # DigitalInOut, Direction,
import time, board                # Pull, DriveMode

led = DigitalInOut(board.IO5)     # LED (IO5)
led.direction = Direction.OUTPUT
led.drive_mode = DriveMode.PUSH_PULL
led.value = False                 # Initially OFF

button = DigitalInOut(board.IO6)  # Pushbutton (IO6)
button.direction = Direction.INPUT
button.pull = Pull.UP

while True:
    while button.value: pass      # Wait for keypress
    led.value = not led.value    # Flip LED state
    time.sleep(0.02)
    while not button.value:      # Wait for key release
        pass
    time.sleep(0.02)
```



LED vezérlés nyomógommbal másképp: ledswitch2.py

- ❖ Kapcsolgassuk a beépített LED-et (IO8) a beépített nyomógomb (IO9) segítségével!

ledswitch2.py

```
from digitalio import *           # DigitalInOut, Direction,
import time, board                # Pull, DriveMode

led = DigitalInOut(board.LED)    # LED (IO8)
led.direction = Direction.OUTPUT
led.drive_mode = DriveMode.PUSH_PULL
led.value = True                 # Initially OFF

button = DigitalInOut(board.BUTTON) # Pushbutton (IO9)
button.direction = Direction.INPUT
button.pull = Pull.UP

while True:
    while button.value: pass      # Wait for keypress
    led.value = not led.value     # Flip LED state
    time.sleep(0.02)
    while not button.value:      # Wait for key release
        pass
    time.sleep(0.02)
```

A beépített LED-nek a katódja van kivezetve az **IO8** lábon, ezért „fordított logikával” kell vezérelni

Ezért a `led.value = True` parancs hatására a LED kialszik

A nyomógomb helyes kezelése:

- 1, Lenyomásra várunk
- 2, Elvégezzük a tevékenységet
- 3, Pergésmentesítési várakozás
- 4, Felengedésre várunk
- 5, Pergésmentesítési várakozás

A Python nyelv elemei

- ❖ A **Python** objektumorientált nyelv, ami lehetővé teszi az adatok és a funkciók objektumokba szervezését
- ❖ **Osztályok és objektumok:**
 - **Osztály (class):** Az objektumok „tervrajza”
 - **Objektum:** Az osztály egy példánya
- ❖ Példa:

```
class Személy:  
    def __init__(self, név, kor):  
        self.név = név  
        self.kor = kor  
anna = Személy("Anna", 25) # 'anna' egy Személy objektum
```

- ❖ A gyakorlatban a fentihez hasonló objektumosztályok különféle funkciókat is definiálnak, amely minden objektumpéldányban rendelkezésre áll

Változók létrehozása és használata

- ❖ A **változó** egy névvel ellátott objektum, amely egy értéket tárol
- ❖ Például: $x = 5$ - itt az **x** változó az 5 értéket tárolja
- ❖ **Változók létrehozása:** változónév = érték
- ❖ Példák:

név = "Anna"
kor = 25
- ❖ **Változók típusai:**
 - **Szöveg** (string): név = "Anna"
 - **Egész szám** (integer): kor = 25
 - **Lebegőpontos szám** (float): magasság = 1.75

Összetett változók

- ❖ **Lista (list):** Több érték tárolása egy változóban.

Példa:

```
gyümölcsök = ["alma", "banán", "cseresznye"]
```

- ❖ **Tuple (tuple):** Több érték tárolása egy változóban, de az értékek nem módosíthatók.

Példa:

```
színek = ("piros", "zöld", "kék")
```

- ❖ **Szótár (dictionary):** Kulcs-érték párok tárolása.

Példa:

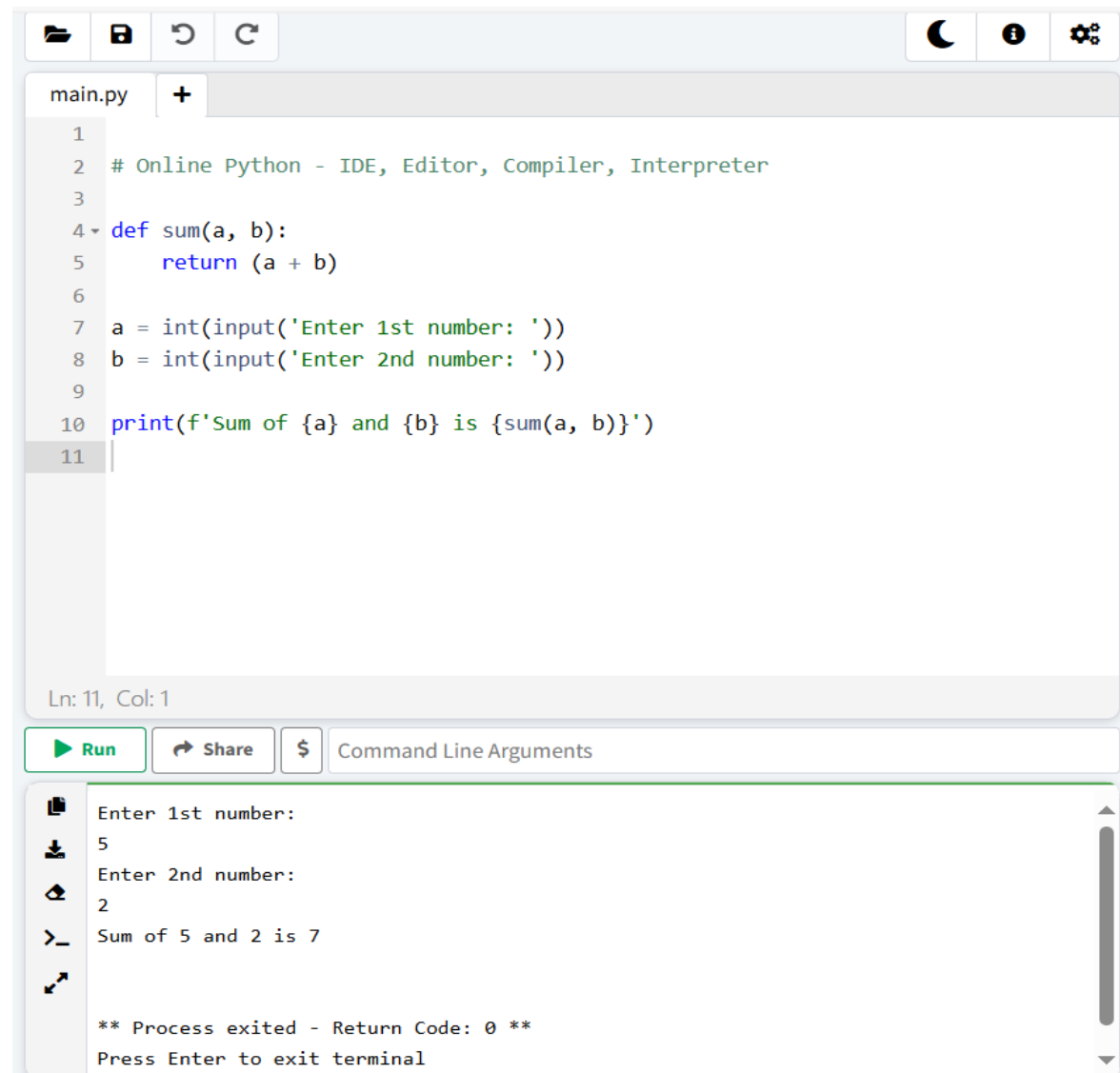
```
diák = {"név": "Anna", "kor": 25, "város": "Budapest"}
```

Alapvető műveletek

- ❖ A változókkal az értékedáson kívül műveleteket is végezhetünk. Az alapvető operátorok a Pythonban: $+$, $-$, $*$, $/$, $//$, $\%$ és $**$, amelyek az összeadást, kivonást, szorzást, osztást, egész osztást, maradékot és hatványozást jelölik
- ❖ Legyen például $x = 5$, $y = 2$, akkor:
 - Összeadás: $x + y = 7$
 - Kivonás: $x - y = 3$
 - Szorzás: $x * y = 10$
 - Osztás: $x / y = 2.5$
 - Egészosztás: $x // y = 2$ (lefelé kerekít a legközelebbi egész számra)
 - Maradék: $x \% y = 1$ (az osztási maradékot adja vissza)
 - Hatványozás: $x ** y = 25$ (az 5 második hatványa)

Python szimulátor

- ❖ Itt az online felületen a Python nyelvet próbálhatjuk ki (ami bővebb a CircuitPythonnál, viszont fizikai programozásra nem tudjuk használni)



The screenshot shows an online Python IDE interface. At the top, there are navigation icons for file operations and system settings. The main area is a code editor for a file named 'main.py'. The code defines a function 'sum(a, b)' that returns the sum of two numbers, and then uses 'input()' to get two numbers from the user and 'print()' to display their sum. Below the code editor, there are buttons for 'Run', 'Share', and 'Command Line Arguments'. At the bottom, a terminal window shows the execution output: 'Enter 1st number: 5', 'Enter 2nd number: 2', and 'Sum of 5 and 2 is 7'. The terminal also shows the process exit message: '** Process exited - Return Code: 0 **' and a prompt to 'Press Enter to exit terminal'.

```
1
2 # Online Python - IDE, Editor, Compiler, Interpreter
3
4 def sum(a, b):
5     return (a + b)
6
7 a = int(input('Enter 1st number: '))
8 b = int(input('Enter 2nd number: '))
9
10 print(f'Sum of {a} and {b} is {sum(a, b)}')
11
```

Ln: 11, Col: 1

Run Share \$ Command Line Arguments

```
Enter 1st number:
5
Enter 2nd number:
2
Sum of 5 and 2 is 7

** Process exited - Return Code: 0 **
Press Enter to exit terminal
```