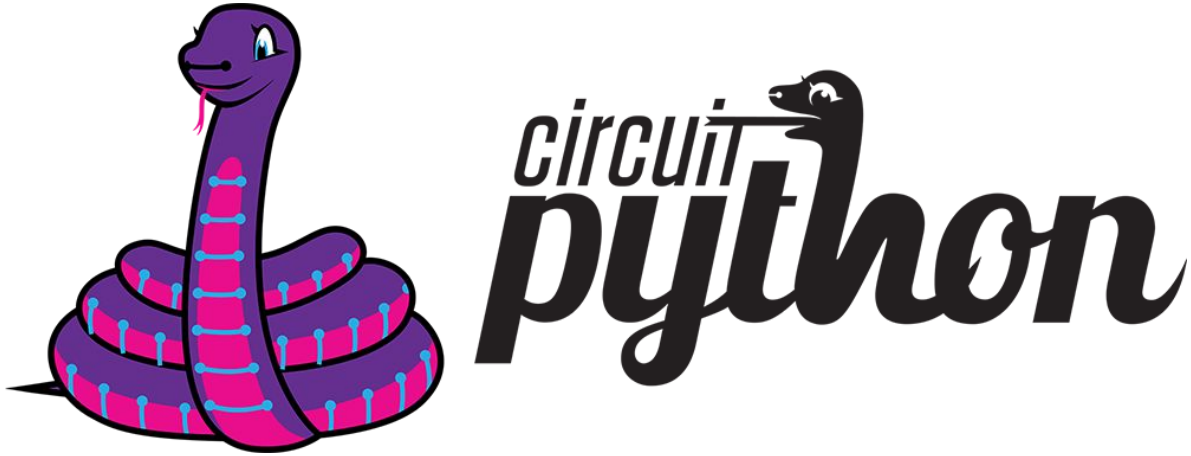
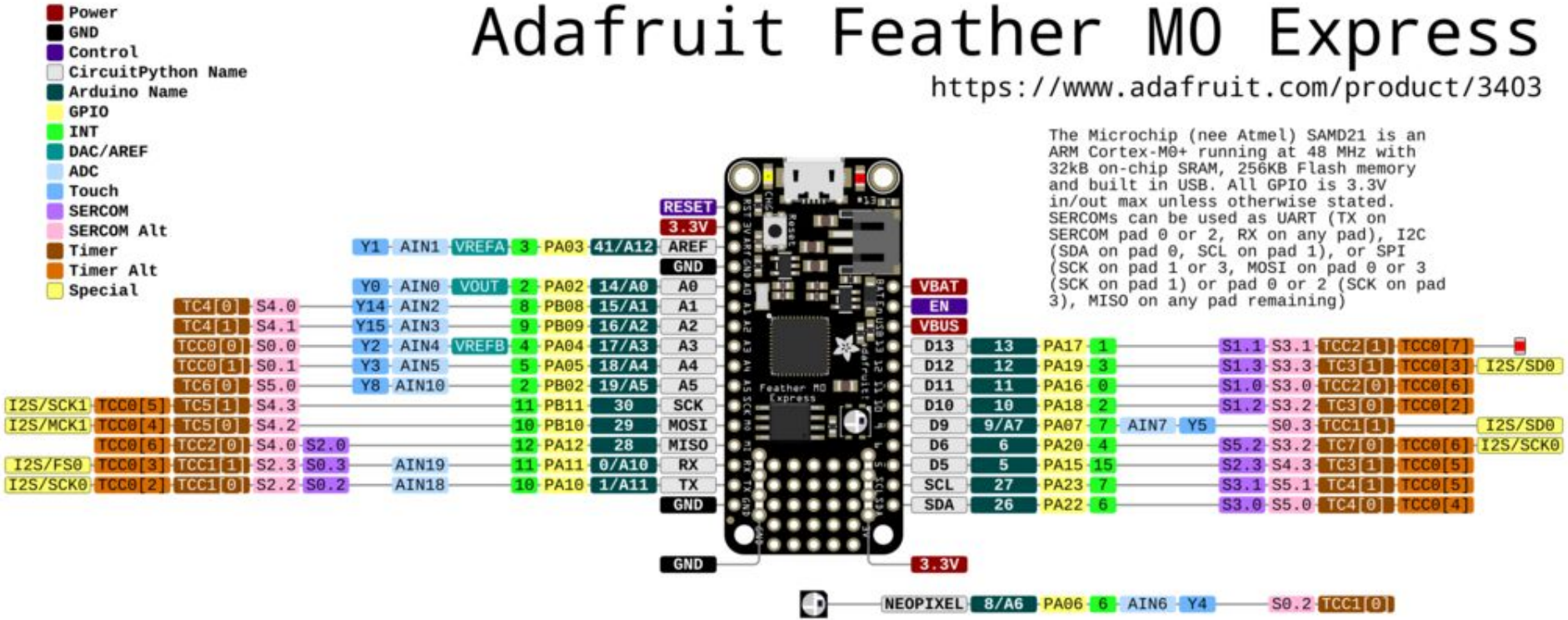


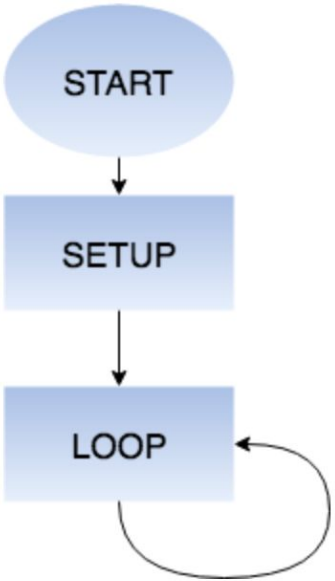
CircuitPython è un *linguaggio di programmazione* progettato per semplificare la sperimentazione e l'apprendimento della programmazione di schede a microcontrollore.

Adafruit Feather M0 Express



Programmare un microcontrollore

Dopo il *boot* (avvio), un microcontrollore esegue una fase di ***setup*** (inizializzazione) e poi ripete una sequenza di operazioni, il ***loop*** (anello) principale.



SETUP

importazione moduli

```
# Lab 2.1 - onboard LED blinking
import board
import digitalio
import time
```

variabili globali

```
led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT
```

```
while True:
    led.value = True
    time.sleep(1)
    led.value = False
    time.sleep(0.5)
```

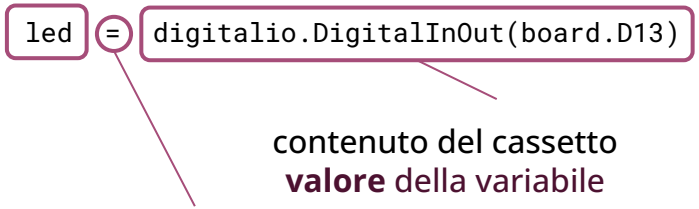
loop principale

Variabili

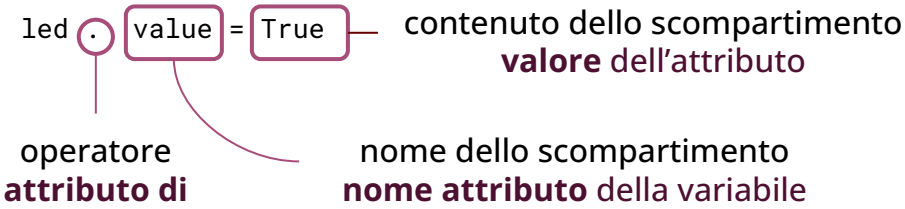
```
# Lab 2.1 - onboard LED blinking
import board
import digitalio
import time

led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT


while True:
    led.value = True
    time.sleep(1)
    led.value = False
    time.sleep(0.5)
```

Una **variabile** si può pensare come l'etichetta incollata su un cassetto di una cassettiera che ha moltissimi cassetti (*memoria RAM*)

Ciascun cassetto può contenere oggetti diversi



Un cassetto può avere al suo interno diversi scompartimenti **attributi** di una variabile



Ciclo *while*

```
while True:
    led.value = True
    time.sleep(1)
    led.value = False
    time.sleep(0.5)
```

spazi vuoti (o ***tab**ulazione*) **indentazione**

blocco, **block**

Ciclo while, **while** *loop*

Il **ciclo while** ripete un blocco di istruzione fino a quando la condizione che viene valutata è *vera*.

in Python un **blocco** è un insieme di istruzioni che sono indentate rispetto all'istruzione che le contiene

la **condizione** da valutare se *vera* o *falsa* si trova tra parentesi tonde

```
while (condizione) :
        istruzione 1
        istruzione 2
istruzione 3
```

operatore **:** (*due punti*) rappresenta l'inizio di un blocco (la riga successiva **deve essere indentata)**

Time

```
# Lab 2.1 - onboard LED blinking
import board
import digitalio
import time

led =
digitalio.DigitalInOut(board.D13)
led.direction =
digitalio.Direction.OUTPUT

while True:
    led.value = True
    time.sleep(1)
    led.value = False
    time.sleep(0.5)
```

viene *importato* nel codice il modulo **time**

il modulo time mette a disposizione delle *funzioni* sul tempo

in Python una **funzione** è una procedura che svolge un determinato compito e può avere degli *argomenti*

sleep() è una **funzione** del modulo time

```
time.sleep(0.5)
```

0.5 è l'**argomento** che viene preso in *input* dalla funzione

Istruzioni condizionali *IF… ELIF … ELSE*

```
# Lab 2.3 - toggle onboard LED
# toggle onboard LED with pushbutton,
# using external pull-up resistor
import time
import board
import digitalio

led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT

switch = digitalio.DigitalInOut(board.D6)
switch.direction = digitalio.Direction.INPUT

while True:
  if switch.value:
    led.value = False
  else:
    led.value = True
  time.sleep(0.01)
```

Nelle **istruzioni condizionali** le istruzioni che il programma eseguirà dipendono da una ***condizione*** che viene valutata

se A allora B

```
if ( A ):
  B
```

se A allora B, altrimenti C

```
if ( A ):
  B
else:
  C
```

se A, allora B, altrimetni se C allora D , altrimenti E

```
if ( A ):
  B
elif ( C ):
  D
else:
  E
```

Istruzioni condizionali *IF… ELIF … ELSE* *(segue)*

```
while True:
    if switch.value:
        led.value = False
    else:
        led.value = True
    time.sleep(0.01)
```

condizione che viene valutata

blocco di istruzioni che vengono eseguite se la condizione è *vera*

blocco di istruzioni che vengono eseguite se la condizione è *falsa*

```
led.value = True
…
led.value = False
```

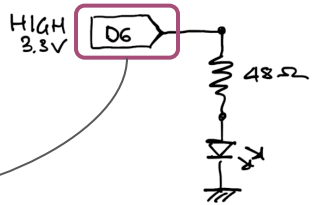
True, False

True e False equivalgono a *Vero/Falso* oppure *0/1*

Sono i due **livelli logici**

Ogni piede GPIO ha un suo **livello logico**, che viene associato al suo attributo ***value***

Per vedere se il pulsante è premuto possiamo indagare *led.value*



0 V —— **True**

3.3 V —— **False**