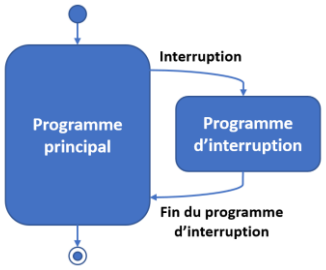
1. Les interruptions

Une **interruption** consiste à interrompre momentanément le **programme principal** exécuté pour effectuer un **autre travail**. Quand cet autre travail est terminé, on retourne à l'exécution du programme principal à l'endroit exact où il avait été laissé. Cet autre travail s'appelle le **programme d'interruption** ou **ISR** (Interrupt Service Routine).



Le programme d'interruption doit avoir un temps d'exécution le plus court possible. Par conséquent, il ne fera aucun calcul compliqué ni d'appel à des fonctions longues.

2. Les différentes interruptions

Il existe plusieurs sources d'interruption :

- **Les interruptions matérielles**, aussi appelé interruption externe (interruption liée au changement de la tension présente sur une broche numérique.
- **Les interruptions logicielles** (déclenchées par des instructions spéciales du processeur (timers…)).

L'interruption matérielle évite de surveiller en permanence l'état d'une broche sur laquelle est raccordé un capteur ou un bouton poussoir. Au changement d'état, le programme d'interruption est exécuté.
Pour l'ESP32, toutes les broches du GPIO peuvent être utilisées en interruption, à l'exception de GPIO 6 à GPIO 11.

3. Configurer une interruption dans MicroPython

3.1. Fonction de gestion des interruptions

La fonction de gestion des interruptions doit être aussi simple que possible, afin que le processeur revienne rapidement à l'exécution du programme principal. La meilleure approche consiste à signaler au code principal que l'interruption s'est produite en utilisant une variable globale, par exemple.

```
def fct_interruption(pin):
    global impulsion
    impulsion = True
```

3.2. Configuration du GPIO

La broche d'interruption agira en tant qu'entrée.

```
bp = Pin(26, Pin.IN)
```

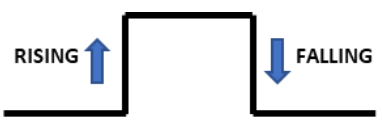
3.3. Méthode irq()

La méthode **`irq()`** permet d'attacher une interruption à la broche d'interruption.

```
bp.irq(trigger = Pin.IRQ_RISING, handler = fct_interruption)
```

La méthode **`irq()`** accepte les arguments suivants :

- **`trigger`** définit le mode de déclenchement :
  - **`Pin.IRQ_RISING`** : déclenche l'interruption sur front montant (passage de 0 à 1) ;
  - **`Pin.IRQ_FALLING`** : déclenche l'interruption sur front descendant (passage de 1 à 0) ;



- **`handler`** définit la fonction qui sera appelée lorsqu'une interruption sera détectée (ici **`fct_interruption`**).

3.4. Exemple

L'action sur le bouton poussoir (front montant) est détecté même si le programme principal est occupé à une autre tâche (temporisation de 5 secondes).
La LED change d'état à l'issue de la fin de l'autre tâche (temporisation) si besoin.


```
from machine import Pin
from time import sleep

impulsion = False
etat_led = False

led = Pin(25, Pin.OUT)
bp = Pin(26, Pin.IN)

# La fonction d'interruption
def fct_interruption(pin):
    global impulsion
    impulsion = True
    print("Appui sur BP détecté")

# Spécifie la fonction à appeler lorsqu'une interruption externe survient
bp.irq(trigger = Pin.IRQ_RISING, handler = fct_interruption)

# Le programme principal
while True:
    if impulsion:
        if led.value()==1:
            led.value(0)
        else:
            led.value(1)
        impulsion = False
    sleep(5)
```

```
bp.irq(trigger = Pin.IRQ_RISING, handler = fct_interruption)
```

La fonction **fct_interruption** est appelée à chaque fois qu'un **front montant** est détecté sur le bouton poussoir **bp** (GPIO26).

```
impulsion = True
```

La variable **impulsion** de type booléen passe à **True** quand on lance le programme d'interruption (fonction **fct_interruption**) et repasse à **False** (dans le programme principal) à l'issue du changement d'état de la LED.
Cette variable est utilisée dans le programme principal pour allumer ou éteindre la Led.
L'utilisation de la variable **impulsion** permet d'avoir un temps d'exécution le plus court possible de la fonction **fct_interruption**.

Sources : https://www.gcworks.fr/tutoriel/esp/Lesinterruptions.html
https://www.raspberryme.com/micropython-interruptions-avec-esp32-et-esp8266/
https://docs.micropython.org/en/latest/library/machine.Pin.html