

Séquence 3

Algorithme et programmation

Extrait de cours

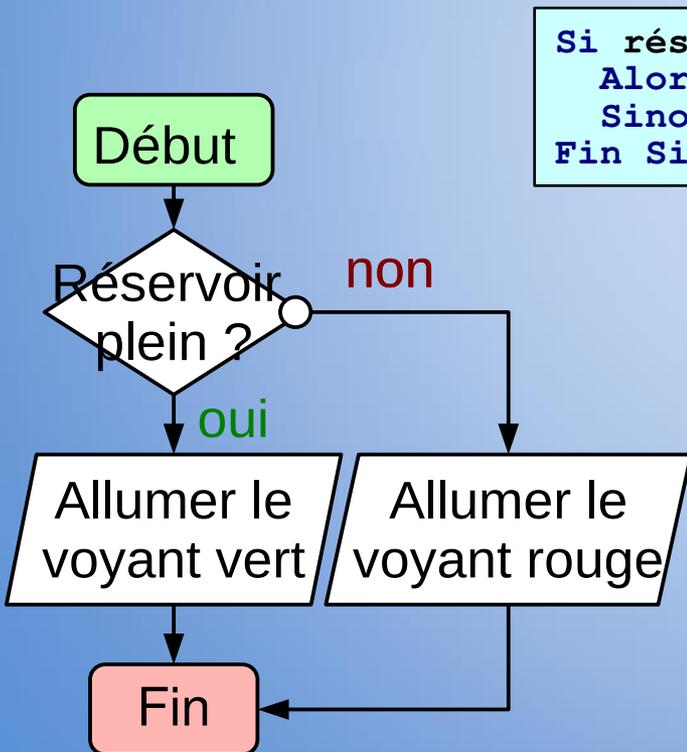
**Les structures
algorithmiques**



1 - Langage de programmation



Les **règles de fonctionnement** d'un système sont généralement définies par programmation (micro-contrôleur, automate) avec des **algorithmes** (suite d'instructions pour réaliser une fonction) codées avec un **langage de programmation**.



```
Si réservoir plein
  Alors allumer le voyant vert
  Sinon allumer le voyant rouge
Fin Si
```

```
IF C.1 = 1 THEN
  high B.1
ELSE
  high B.2
ENDIF
```

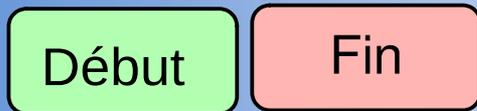
```
if (digitalRead(cpt_plein) == HIGH) {
  digitalWrite(led_v, HIGH);
}
else {
  digitalWrite(led_r, HIGH);
}
```

```
if cpt_plein.read() == True :
  → led_v.write(1)
else :
  → led_r.write(1)
```

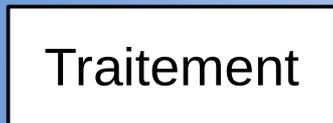
2 - Algorithme



Un **algorithme** (organigramme de programmation, flowchart, logigramme, ordinogramme) est une représentation graphique d'un algorithme (symboles normalisés ISO 5807).



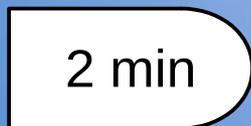
Début et fin
d'un algorithme



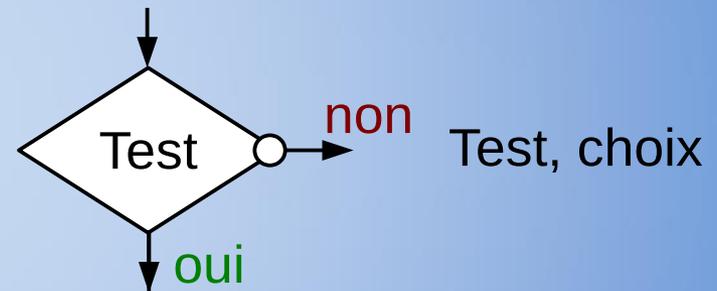
Opération sur des
données, instruction



Lecture d'une entrée
ou écriture d'une
sortie



Temporisation

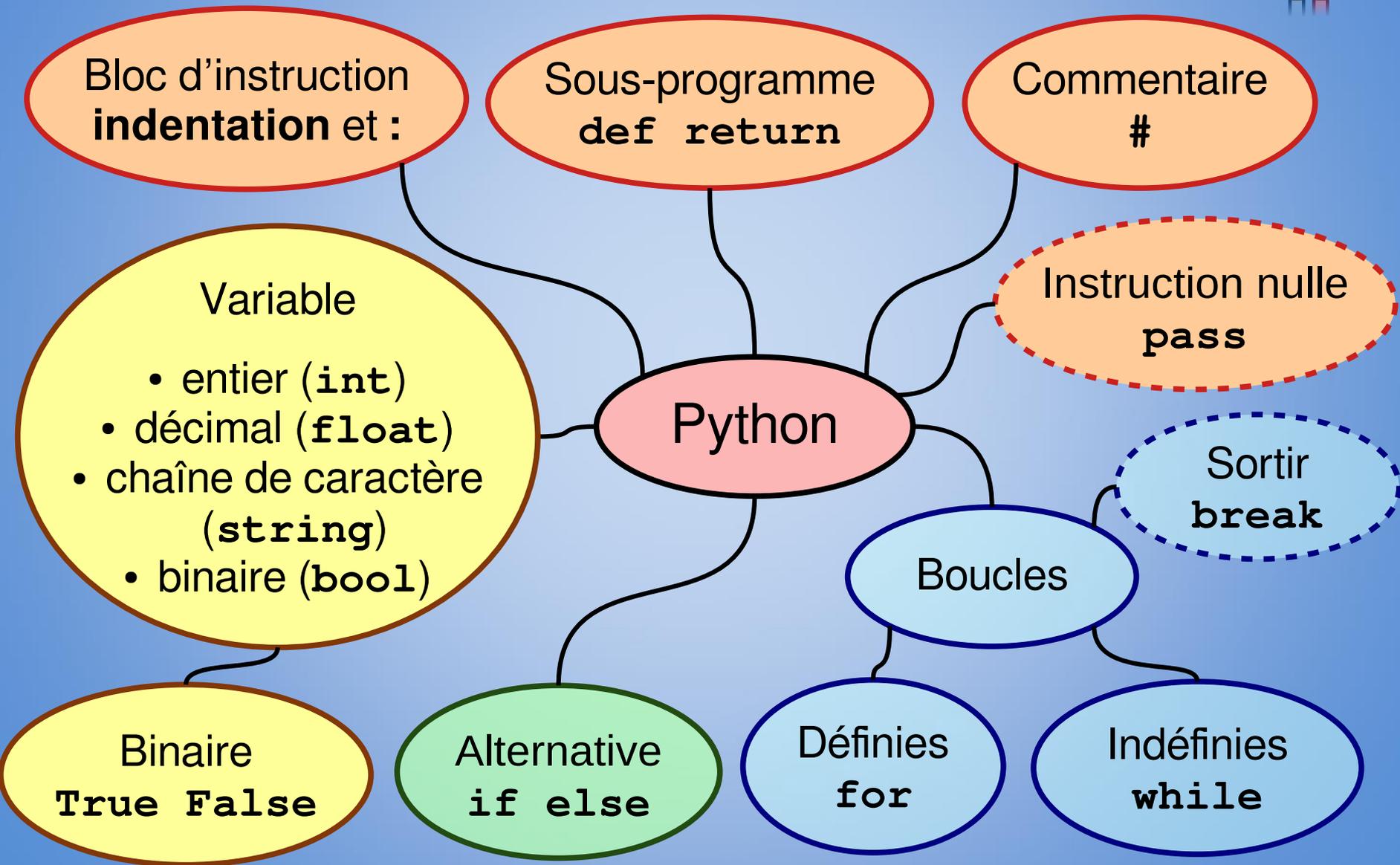


Test, choix

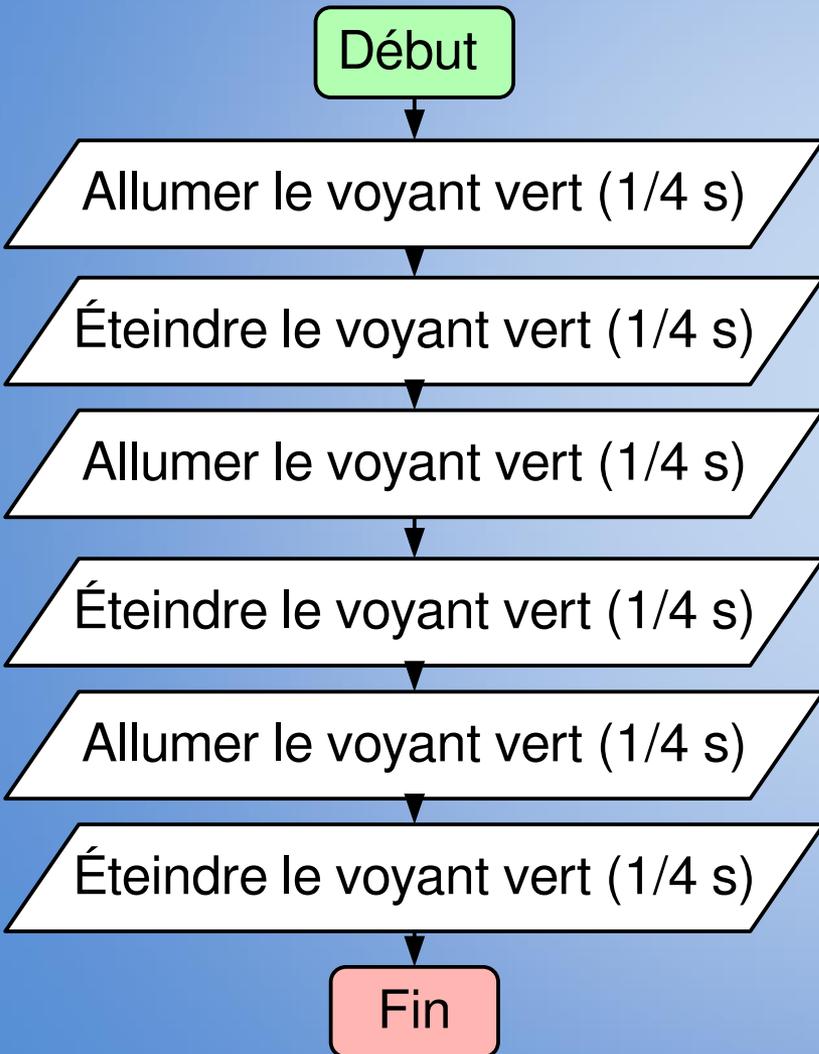


Appel d'un
sous-programme

3 – Panorama du langage Python



4 - Structure linéaire



Algorithme

```
Allumer le voyant vert pendant 1/4 s
Éteindre le voyant vert pendant 1/4 s
Allumer le voyant vert pendant 1/4 s
Éteindre le voyant vert pendant 1/4 s
Allumer le voyant vert pendant 1/4 s
Éteindre le voyant vert pendant 1/4 s
```

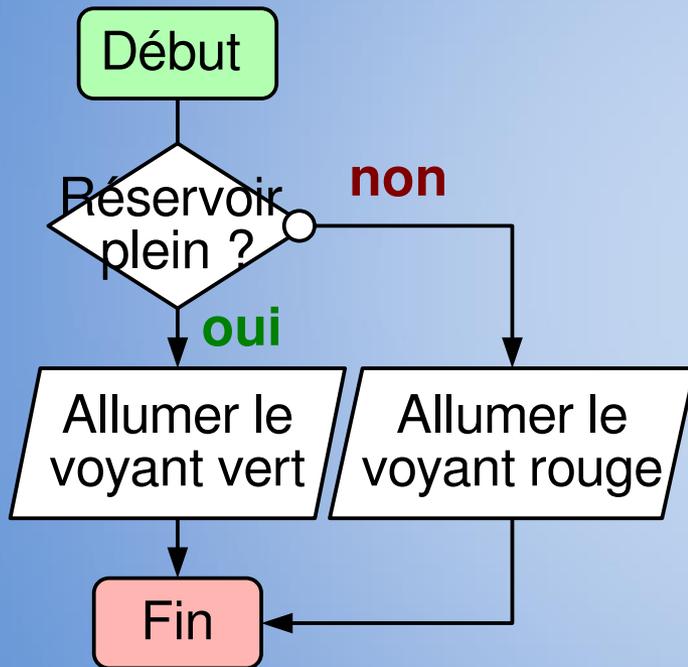
Programme Python

```
led_v.write(1)
time.sleep(0.25)
led_v.write(0)
time.sleep(0.25)
led_v.write(1)
time.sleep(0.25)
led_v.write(0)
time.sleep(0.25)
led_v.write(1)
time.sleep(0.25)
led_v.write(0)
time.sleep(0.25)
```

5 - Structure conditionnelle



Une **structure conditionnelle** permet de mettre en place une **alternative** en fonction d'une condition.



Algorithme

```
Si réservoir plein
  Alors allumer le voyant vert
  Sinon allumer le voyant rouge
Fin Si
```

Programme Python

```
if cpt_plein.read() == True :
    → led_v.write(1)
else :
    → led_r.write(1)
```

Les conditions peuvent être

- $a == b$: a est égal à b
- $a != b$: a est différent de b
- $a < b$: a est strictement inférieur à b
- $a <= b$: a est inférieur ou égal à b

- $a == b$ **and** $c == d$: les deux conditions doivent être vrai (fonction ET)
- $a == b$ **or** $c == d$: une des deux conditions doit être vrai (fonction OU)

6 – Structure itérative indéfinie



Une **structure itérative** permet de répéter plusieurs fois l'exécution d'une même action (boucle). Elle est **indéfinie** quand nombre de répétition est inconnu.

While

Algorithme

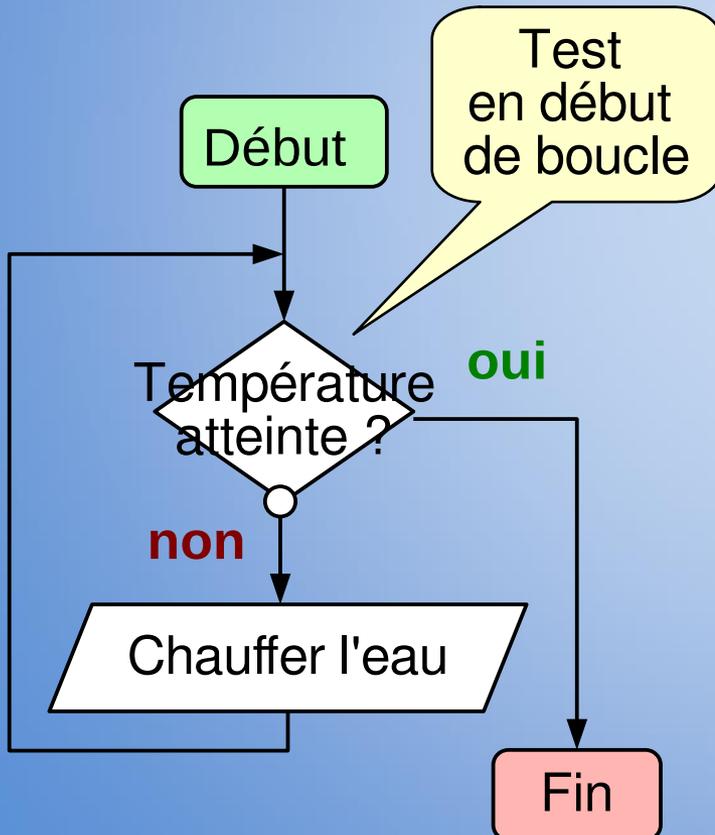
Tant que Température non atteinte Répéter
Chauffer l'eau

Jusqu'à Température atteinte Répéter
Chauffer l'eau

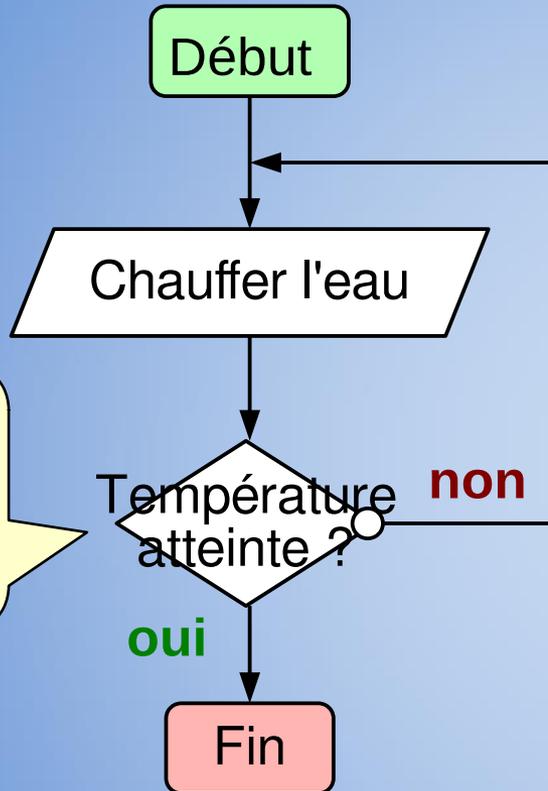
Until (n'existe pas en Python)

Programme Python

```
temperature = cpt_temp.read()
while temperature < 520 :
    → chauffage.write(1)
    temperature = cpt_temp.read()
chauffage.write(0)
```



6 – Structure itérative indéfinie



Test en fin de boucle

Algorithme

Répéter

Chauffer l'eau

Tant que Température non atteinte

Répéter

Chauffer l'eau

Jusqu'à Température atteinte

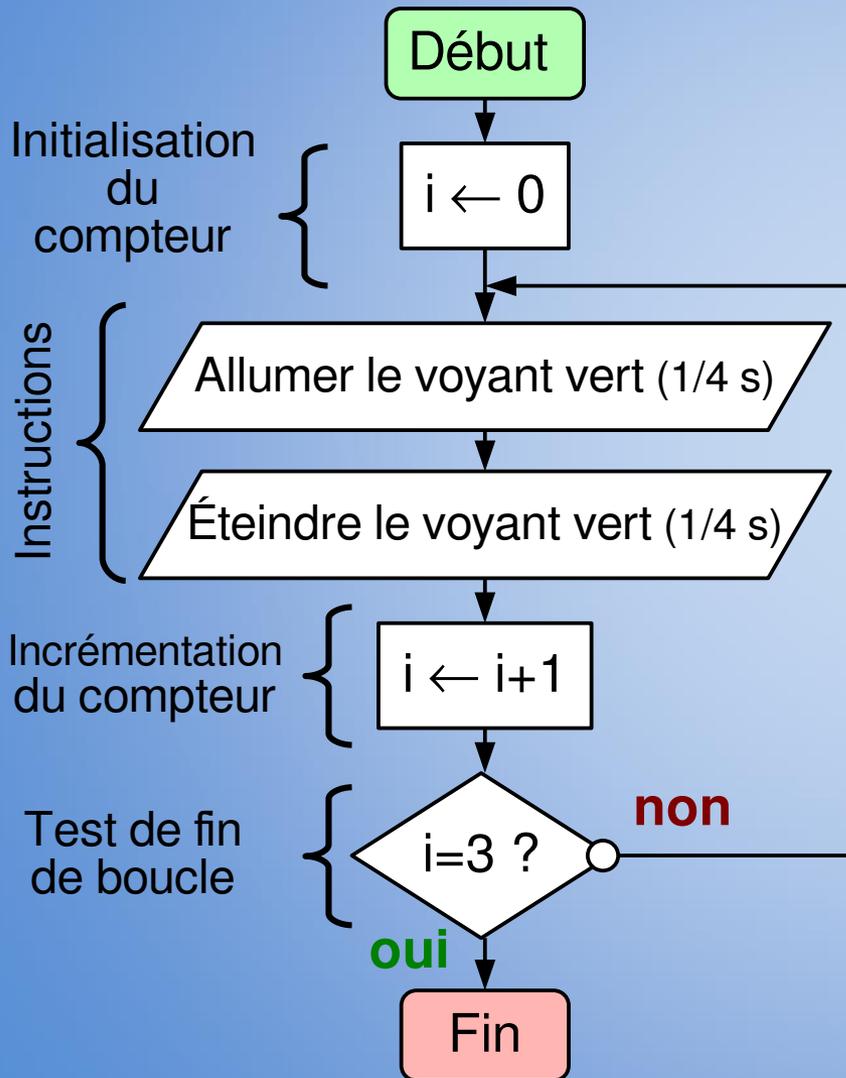
Programme Python

```
temperature = cpt_temp.read()
while True :
    → chauffage.write(1)
    temperature = cpt_temp.read()
    if temperature > 520 :
        → → break
chauffage.write(0)
```

while True : Boucle infinie

break : Sortir de la boucle

7 – Structure itérative définie



Quand le nombre de répétition est connu (avant de commencer la boucle), la structure itérative est dite **définie**, on utilise un **compteur**.

Algorithme

```
Pour i de 0 à 2 avec un pas de 1
  allumer le voyant vert pendant 1/4 s
  éteindre le voyant vert pendant 1/4 s
Fin Pour
```

Programme Python

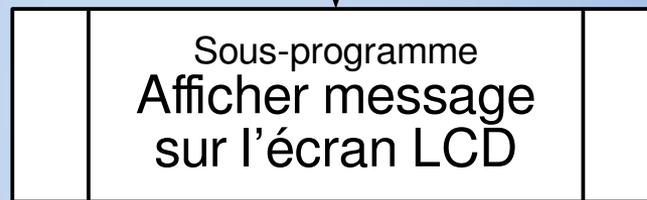
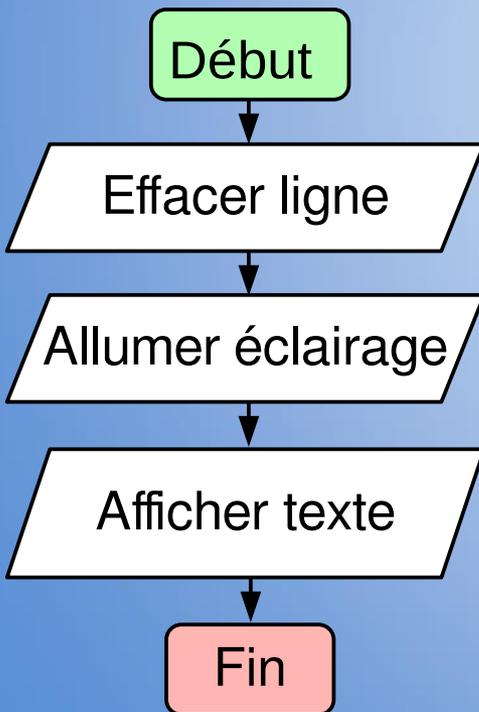
```
for i in range (3) :
  → led_v.write(1)
  time.sleep(0.25)
  led_v.write(0)
  time.sleep(0.25)
```

8 - Sous-programme



Un **sous-programme** (procédure, fonction) est une portion de programme qui peut être **appelée plusieurs fois**.

- Éviter de ré-écrire plusieurs fois la même suite d'instructions.
- Structurer le programme en **tâche élémentaire**.



Définition
de la fonction

Programme Python

Argument

```
def afficher_msg(texte) :  
→ lcd.send_sysex( DISPLAY_TEXT_ADDR, 0x80, 0x01 )  
  lcd.send_sysex( DISPLAY_TEXT_ADDR, 0x80, 0x08 )  
  lcd.send_sysex( STRING_DATA, texte )
```

```
afficher_msg("En préparation ...")
```

Argument

Appel de
la fonction