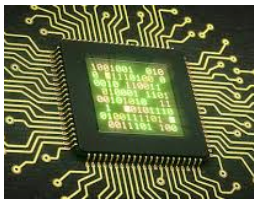
	SCIENCES NUMÉRIQUES ET TECHNOLOGIE	PROGRAMMATION	2nd
Ce que je dois retenir	Écrire et développer des programmes pour répondre à des problèmes et modéliser des phénomènes physiques, économiques et sociaux..		

Programme informatique

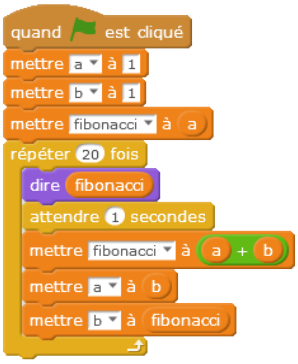


Un **programme** informatique est une suite d'instructions déterminées par le programmeur pour répondre à un problème (jeux, application, système réel, …).

Il est mis au point dans un **langage** de haut niveau (scratch, python, C, ...), testé et corrigé, puis traduit en langage compréhensible par le **microprocesseur** (ou microcontrôleur) dans lequel il sera transféré : sous forme de « 0 » et « 1 » , le code **binaire**.

Algorithme	Langage de programmation de haut niveau (graphique ou code)	Langage machine binaire

```
a = valeur 1
b = valeur 1
fibonacci = a
Répéter 20 fois :
   Afficher fibonacci
   fibonnacci = valeur a+b
   a = valeur b
   b = valeur fibonacci
Fin Répéter
```




```
a=1
b=1
fibonacci=a
for n in range(20):
    print(a)
    fibonacci=a+b
    a=b
    b=fibonacci
```

Affectations, variables

Une **variable** permet de **mémoriser** une information : nombre, texte, ... La **valeur associée** à la variable peut changer au cours de l'exécution du programme. L'instruction d'**affectation** est l'action d'associer une valeur à cette variable. On la note le plus souvent avec le signe égal « = » (ne pas confondre avec l'égalité mathématique). Lors de l'évaluation de l'instruction d'affectation, la partie droite est évaluée en premier et le résultat obtenu est associé à la variable.

Exemple 1 : a = 12 *>>> 12 est la valeur associée à la variable a*

Exemple 2 : b = a + 1 *>>> 13 (résultat de l'évaluation de l'expression a+1) est la valeur associée à la variable b*

On peut mémoriser différents types de données :

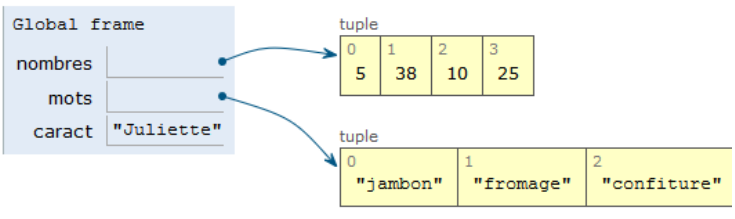
- des **entiers** : entiers relatifs comme 74 ou -2
- des **flottants** : nombres décimaux comme 12,97 ou -14,0
- les **chaînes de caractères** : successions de caractères encadrés par des guillemets "Ok !", "456", « Hello World »

Séquences

Une **séquence** est un ensemble fini et ordonné d'éléments (nombres ou caractères).

Exemple :

```
1  nombres = 5, 38, 10, 25
2  mots = "jambon", "fromage", "confiture"
3  caract = "Juliette"
```



On peut manipuler les séquences de différentes façons :

- isoler une donnée
- compter le nombre d'éléments d'une séquence
- diviser ou concaténer des éléments
- effectuer des opérations (addition, multiplication, …) ou des fonctions (min, max, ...)

Instructions conditionnelles

Pour aiguiller dans différentes direction l'éxécution du programme en fonction du souhait de l'utilisateur ou des intéractions avec l'environnement extérieur, il est possible d'utiliser des **instructions conditionnelles :**

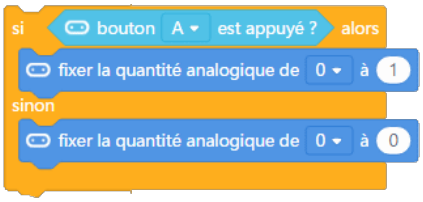
```
si condition :
    Instruction A
```

```
si condition :
    Instruction A
sinon:
    Instruction B
```

```
if x > 10:
    print(x, "est plus grand que 10")
```

```
if button_a.is_pressed():
    pin0.write_analog(1)

else:
    pin0.write_analog(0)
```



Boulces bornées et non bornées

Lorsque des instructions sont répétées plusieurs fois dans un programme, on utilise une **boucle bornée** :

```
Pour i allant de borne_mini à borne_maxi :
    Instruction A
```

```
for i in range(1,4):
    print("i apour valeur", i)
```

Dans d'autres cas, il est nécessaire de répéter des instructions jusqu'à ce qu'une certaine condition soit vérifiée. On utiliser alors une **boucle non bornée** :

```
Tant que condition :
    Instruction A
```

```
x = 1
while x < 10:
    print("x a pour valeur", x)
    x = x * 2
```

Définition et appels de fonctions

Dans un programme, il est possible d'écrire de petits programmes ou **sous-programmes** intermédiaires appelées **fonctions**. Elles permettent de simplifier le programme principal et de décomposer le problème de départ en sous-problèmes.

```
definir nom_fonction(liste de paramètres):
    bloc d'instructions


appeler nom_fonction(liste de paramètres)
```

```
def compteur(stop):
    i = 0
    while i < stop:
        print(i)
        i = i + 1

compteur(4)
```

Pour les projets les plus ambitieux il sera vite important d'organiser son travail. Les fonctions peuvent alors être enregistrées dans des fichiers distincts pour plus de flexibilité et de lisibilité.

Exemple : Utiliser l'afficheur de la carte programmable microbit


```
from microbit import *

while True:
    display.show(Image('00000:05050:00000:50005:05550'))
```