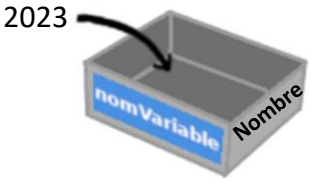
LYCÉE de l'ÉLORN
le choix de votre réussite

PROGRAMMATION
avec le Langage Python

I- Rappel : Le b.a.-ba du Python

1.1 Les variables

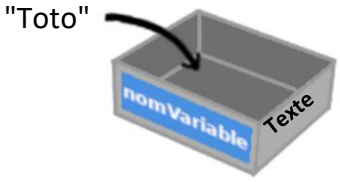
Une variable est un espace mémoire dans lequel il est possible de stocker une valeur. C'est un peu comme une « boite » sur laquelle il y aurait une étiquette désignant le nom de son contenu. Les algorithmes manipuleront le nom de la variables (nom des boites) plutôt que leur contenu.



- Le **nom** de la variable correspond à l'étiquette collée sur la boîte.
- La **valeur** de la variable est l'information stockée. Cette valeur peut changer au cours de l'exécution d'un programme.
- Les variables peuvent être de différents **types**. Il faut prendre soin d'adapter le contenu au contenant et vice-versa.

Dans cet exemple le contenu de la variable C sera 25 après l'exécution du programme

```
a = 10
b = 15
c = a + b
```



1.2 Les types de bases en Python

Nom français	Nom Python	Désignation	Exemple
Entier	Int ou long	Nombre entier (sans virgule)	1 10 20254
Réel	float	Nombre à virgule (nombre réel)	1,1 5425,87458 12,0
Chaîne de caractères	str	Suite quelconque de caractères	"Bonjour" "Ok"
booléen	bool	Peut prendre les valeurs True ou False (vrai ou faux)	True ou False

Il existe plusieurs fonctions en Python qui permettent de forcer le type d'une variable en un autre type :
int() : permet de modifier une variable en entier.
float() : permet la transformation en flottant.
str() : permet de transformer la plupart des variables d'un autre type en chaînes de caractères.

Exemple : Soit x une variable de type chaîne de caractères : x="50"

L'opération x+10 générera une erreur. Pour réaliser une opération arithmétique sur la variable x nous devons la convertir en type entier avec la commande suivante : x=int(x)

```
>>> x= "50"
>>> x
>>> x+10
>>> x=int(x)
>>> x+10
>>> x*10
```

1.3 Les opérateurs d'entrées/sorties en Python

a) Les sorties

Pour permettre au programme en cours d'exécution d'afficher un texte ou un nombre on utilise la commande print().

```
print("Bonjour !")      # c'est la chaîne de caractère "Bonjour" qui sera affichée.
a = 3
print(a)                # c'est le contenu de la variable a qui sera affiché.
print("Le carrée de", a,"est", a * a)   # affichera « Le carrée de 3 est 9 »
```

b) Les entrées

Il est souvent nécessaire de donner une valeur en utilisant le clavier. On utilisa alors la commande Input().

```
nom = input("Quel est votre nom ?")
#nom contiendra la chaîne de caractère saisie au clavier
```

Input() est une fonction **qui renvoie toujours une chaîne de caractères**. Il est donc parfois nécessaire de changer le type de la variable rentrée.

```
n = input("Entrer un nombre")          # n est une variable de type String
n = int(n)                             # n devient une variable de type entier
print("Le carrée de", n,"est", n * n)
```

1.4 Les opérateurs de base en python

+	addition	
-	soustraction	
*	multiplication	
/	division	5/2 donne 2.5
//	division entière	5//2 donne 2
%	reste de la division entière	5%2 donne 1 -> Numworks : fmod(5,2)
**	puissance	2**10 donne 1024

==	égalité (à ne pas confondre avec l'affectation)
!=	différent
<, >, <=, >=	inférieur, supérieur, inférieur ou égal, supérieur ou égal
and	opérateur booléen ET
or	opérateur booléen OU
not	opérateur booléen NON

1.5 Les structures algorithmiques fondamentales

a) La structure "*SI ALORS SINON*"

Nous utilisons cette structure dès que nous voulons exprimer une possibilité de choix simple à deux alternatives. C'est une des opérations les plus utilisées, nous pourrions tout décrire avec des opérations simples et des « SI ALORS SINON ».

Exemple :

```
If note_ds >= 10:
    print("Vous avez la moyenne")
else:
    print("Vous n'avez pas la moyenne")
```

Le bloc else: est optionnel

On peut aussi « imbriquer » les if avec l'alternative elif : (la contraction de alors si)

b) La structure de boucle *"REPETER TANT QUE" (WHILE)*

Dans cette structure on commence par tester une condition. Si elle est vérifiée, le traitement est exécuté.

Exemples :

Test d'un mot de passe

```
mot_de_passe=""

while mot_de_passe!="bidule":
    mot_de_passe=input("mot de passe svp : ")

print("le mot de passe est validé")
```

Écrit la table de multiplication de 7

```
i=1
while i<11:
    print(i, "x 7 =", i*7)
    i=i+1
```

c) La structure de boucle *"POUR … DE … A … " (FOR)*

Lorsque l'on souhaite répéter un nombre donné de fois la même instruction ou le même bloc d'instructions, la commande for est la plus appropriée.

Exemples :

Écrit tous les nombres de 0 à 99 inclus

```
for i in range(0,100):
    print(i)
```

Écrit la table de multiplication de 7

```
for i in range(1,11):
    print(i, "x 7 =", i*7)
```

II - Applications sur Edupython 

2.1 Donner un programme capable de saisir la taille puis le poids d'une personne afin d'afficher son IMC accompagné d'un bilan.

Pour information :
L'IMC permet de savoir si votre poids est idéal, autrement dit, s'il est adapté à votre taille. Il correspond au ***poids divisé par le carré de la taille*** *(****IMC = poids en kg/taille² en m****).*
Voici le bilan :

- *IMC < 18,5 kg/m² : insuffisance pondérale*
- *18,5 <= IMC < 25 : poids normal*
- *25 <= IMC < 30 : surpoids*
- *IMC >= 30 : obésité*

Les fonctions en Python permettent de structurer le code en le rendant plus clair, réutilisable et facile à maintenir. Elles favorisent la modularité, en divisant un programme en blocs logiques répondant à des tâches précises. Ainsi, elles améliorent l'efficacité du développement et facilitent la collaboration entre programmeurs.

2.2 Tester ces deux exemples. Modifier l'exemple 2 pour qu'il renvoie le résultat de la somme de 3 nombres.

```
# Exemple 1 --- Définition d'une fonction simple
def saluer(nom):
    # La fonction prend un paramètre "nom" et affiche un message
    print("Bonjour", nom)

# Appel de la fonction
saluer("Alice")
saluer("Bob")



# Exemple 2 --- Définition d'une fonction qui calcule la somme de deux nombres
def addition(a, b):
    somme = a + b
    # On utilise "return" pour renvoyer un résultat
    return somme

# Utilisation de la fonction et affichage du résultat

print("La somme est :", addition(5, 7))
```

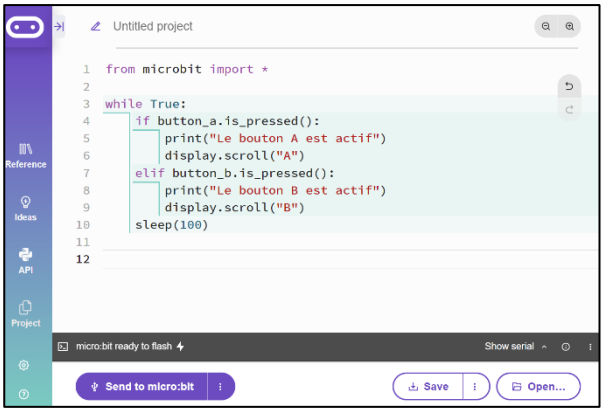
2.3 Créer une fonction capable de renvoyer la racine carrée d'un nombre si elle existe (vous devrez faire vérifier si le nombre est positif ou nul pour ce traitement, dans le cas où le nombre est négatif, vous écriverez « Il n'y a pas de racine carrée réelle pour ce nombre ».

2.4 Créer une fonction capable de calculer la ou les solutions d'une équation du second degré du type $Ax^2 + Bx + C = 0$.

III – Applications sur la carte Micro:Bit

Processeur : PU ARM® Cortex ™ M0 32 bits
Fréquence de l'horloge de processeur : 16 Mhz
RAM : 16 Ko
Courant des broches : 3,3 V
6 broches analogiques + 19 broches digitales

Vous pouvez programmer la carte avec un éditeur en ligne sur https://python.microbit.org/v/3



Nous débuterons par la découverte de quelques capteurs tout en mettant en œuvre une liaison série.

3.1 Prise en main de l'interface

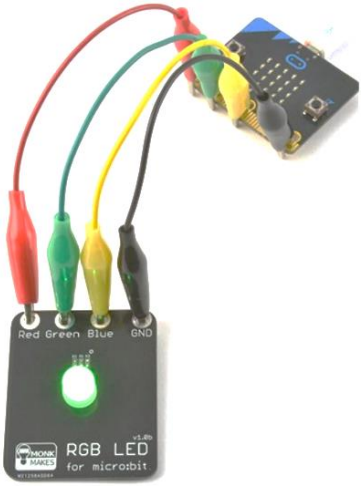
a) *Exécuter le script ci-contre (un copier/coller est possible). Faire fonctionner l'interface Série (Show serial) pour observer le fonctionnement dont le résultat des print().*

```
from microbit import *
while True:
    if button_a.is_pressed():
        print("Le bouton A est actif")
        display.scroll("A")
    if button_b.is_pressed():
        print("Le bouton B est actif")
        display.scroll("B")
    sleep(100)
```

b) *Modifier le script afin qu'il soit afficher sur l'interface série le nombre de fois que l'on a actionné les boutons.*
Ex : « le bouton A a été actionné 6 fois »

3.2 Câbler le module de LED sur la carte à l'aire des fils en respectant les liens suivants :

- Red → broche 0
- Green → broche 1
- Bleu → broche 2
- GND → GND



Programmer la carte en utilisant le code suivant puis tester-le.

```
from microbit import *

def prog_couleur(R, V, B):
    pin0.write_analog(R*4)
    pin1.write_analog(V*4)
    pin2.write_analog(B*4)

while True:
    prog_couleur (255, 0, 0)
    sleep(1000)
    prog_couleur (0, 255, 0)
    sleep(1000)
    prog_couleur(0, 0, 255)
    sleep(1000)
```

L'intensité lumineuse de chaque couleur est définie par la fonction *prog_couleur(R,V,B)* où les variables R, V et B peuvent prendre toutes les valeurs de 0 à 255.

a) Modifier le programme afin de générer le cycle de couleur suivant :

{ rouge, vert, bleu, magenta, cyan, jaune, blanc}

c) Programmer la micro:bit afin de respecter le comportement suivant :

Si l'on actionne le bouton A :

Ecrire « France » (sur la micro:bit puis faire défiler les couleurs bleu, blanc, rouge sur la Led.*

Si l'on actionne le bouton B :

Ecrire « Italie » sur la micro:bit puis faire défiler les couleurs vert, blanc, rouge sur la Led.

**Avec la méthode display.scroll('France')*