LYCÉE de l'ÉLORN
le choix de votre réussite

Programmation en Python
Découverte du robot Maqueen

I - Généralités sur les variables

a) Désignations :

Elles représentent des nombres, des états, des caractères, des chaînes de caractères dont la valeur peut être modifiée au cours de l'exécution du programme. Les variables sont définies par deux caractéristiques essentielles, à savoir :

L'identificateur : le nom de la variable.
Le type : la nature de la variable (entier, réel, caractère, chaîne de caractères …)

b) Les types de bases :

Nom français	Nom Python	Désignation	Exemple
Entier	Int ou long	Nombre entier (sans virgule)	1 10 20254
Réel	float	Nombre à virgule	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)	

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.
long() : transforme une valeur en long.
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).

II - 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("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.

```
nombre = input("Entrer un nombre")          # ou directement
n = float(nombre)                           # nombre = float(input("Entrer un nombre"))
print("Le carrée de", n,"est", n * n)
```

III - Les opérateurs de base en python

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

==	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

Tester (en mode console sous Python) la valeur d'une variable contenant un nombre.
Soit n une variable contenant le nombre 12

Test en français	***Écrit en Python 3***	***Renvoie quoi ?***
n est égal à 12	n==12	
n est égal à 10	n==10	
n est positif	n>0	
n est différent de 10	n!=10	
Si *n* est compris strictement entre 0 et 20	0<n<20 (n>0) and (n<20)	
Si *n* est divisible par 6	fmod(n,6)==0	
Si *n* est divisible par 5	fmod(n,5)==0	

IV- Les structures algorithmiques fondamentales

a) La structure "*SI ALORS SINON" (IF THEN ELSE)*

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 ».

Notation algorithmique

```
si condition alors
  action1
sinon
  action2
finsi
```

Exemple :

```
note_ds = 12

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

Dans cette structure l'exécution d'une des deux actions distinctes ne dépend que du résultat d'un test. Notez que le « *sinon* » est optionnel.

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

Dans cette structure on commence par tester la condition.

Si elle est vérifiée, le traitement est exécuté.

```
tant que condition
  action1
  action2
  …
Fintant que
```

Exemples en python :

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 validée")
```

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.

```
pour compteur de début à fin
    action1
    action2
   …
finpour
```

Exemples en python :

Écrit tous les nombres de 0 à 100 inclus

```
for i in range(0,101):
    print(i)
```

Écrit la table de multiplication de 7

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

V- Application sur la programmation du robot Maqueen

Nous écrirons nos programmes dans l'éditeur en ligne à l'adresse suivante :
http://python.microbit.org/v/1

5.1 Instructions de déplacement du robot

Pour piloter le robot nous aurons besoin de charger le fichier maqueen.py :

- **Etape 1 :** Télécharger le fichier maqueen.py depuis le réseau pédagogique (document en consultation) dans votre espace personnel.
- **Etape 2 :** Dans l'éditeur de programmes, cliquer sur Load et charger le fichier maqueen.py.
- **Etape 3 :** Vous pouvez entrer vos instructions à partir de la ligne 82.

A CHAQUE FOIS VOUS PRENDREZ BIEN LE SOIN DE METTRE LE ROBOT SUR OFF PENDANT LE TELECHARGEMENT, PUIS DE LE DEBRNACHER AVANT DE LE REMETTRE SUR ON

a) Tester le script ci-contre :

b) En vous inspirant de ce script, essayer de programmer le robot pour qu'il fasse un carré en se déplaçant sur le sol.

```
robot.avance(20)
time.sleep(2)
robot.moteurDroit(20)
robot.moteurGauche(-20)
time.sleep(0.6)
robot.avance(20)
time.sleep(2)
robot.stop()
```

5.2 Utilisation du capteur à ultrason

a) Tester le programme ci-contre :
« while True : » permet de réaliser une boucle infinie, c'est-à-dire que le robot va répéter indéfiniment les instructions qui suivent :

- Mesurer la distance devant le robot
- Si la distance est inférieure à 10, alors le robot s'arrête
- Sinon il avance à la vitesse 50

```
82 while True:
83     distanceAvant = robot.distance()
84     if distanceAvant < 10:
85         robot.stop()
86     else:
87         robot.avance(50)
```

b) Avec les instructions que vous connaissez, écrire un programme qui permet au robot d'éviter un obstacle qui se trouve devant lui en le contournant.

5.3 Utilisation des capteurs de suivi de ligne

Votre robot possède, sous le châssis, 2 capteurs Infrarouges suiveurs de ligne. Ces capteurs détectent s'ils sont sur une surface noire ou sur une surface blanche. Ainsi pour suivre une ligne noire suffisamment large, on pourra par exemple utiliser l'algorithme suivant :

Si les deux capteurs sont sur le noir :
- Avancer

Si le capteur gauche est sur le noir et le capteur droit est sur le blanc :
- Tourner à gauche

Si le capteur gauche est sur le blanc et le capteur droit est sur le noir :
- Tourner à droite

a) Prenez connaissance sur Internet de la déclaration et de l'utilisation des fonctions en python.

b) Insérer dans votre code les fonctions suivantes :

```
def capteurGauche():
    return microbit.pin13.read_digital()

def capteurDroit():
    return microbit.pin14.read_digital()
```

Ces deux fonctions revoient « 0 » si le capteur détecte du noir et « 1 » si le capteur détecte du blanc.
La fonction suivreLigne va donc commencer ainsi :

```
def suivreLigne():
    if capteurGauche() == 0 and capteurDroit() == 0 :
        robot.avance(20)
    elif capteurGauche() == 0 and capteurDroit() == 1 :
```

c) Compléter la fonction suivreLigne puis testez-là en écrivant le programme principal suivant :

```
while True :
    suivreLigne()
```