TP1_mouvement_et_fonctions_cor_prof

January 28, 2017

```
In [1]: from poppy.creatures import PoppyErgoJr

        poppy = PoppyErgoJr()
```

0.0.1 Initialisation

Ecrire un jeu d'instructions permettant

1. d'initialiser la vitesse de chacun des moteurs à 30°/s.
2. de mettre les moteurs m1 à m6 dans les positions données par la liste suivante *pos_init=[0,-90,30,0,60,0]*.

Remarque :
Lorsque la vitesse du moteur *poppy.m1==0* ou lorsque le moteur est dans l'état *compliant==True*, la commande *poppy.m1.goal_position=50* n'a pas d'effet.

```
In [2]: i=0
        pos_init=[0,-90,30,0,60,0]
        for m in poppy.motors:
            m.moving_speed=60
            m.compliant=False
            m.goal_position=pos_init[i]
            i=i+1
```

0.0.2 Quelques remarques

- L'itérateur *m* est ici une variable locale au sein de la boucle pour et il va parcourir dans l'ordre la liste *poppy.motors* On peut donc considérer qu'il est du type *poppy.mi* il possède alors tous les attributs de *poppy.mi*
- On a besoin d'un compteur pour parcourir la liste contenant les positions initiales. Ici, la variable *i* qui doit alors être incrémentée de 1 à chaque passage dans la boucle *pour* afin de faire coincider le moteur et la position à atteindre par ce moteur.

0.0.3 Faire de ce jeu d'instructions une procédure

On veut faire de ces instructions d'initialisation une procédure dont les arguments sont le robot nommé *bot* et la liste donnant les positions initiales des moteurs nommé *pos_initiale*. Le prototype

de cette procédure est : *f_init(bot,pos_initiale)*. A la fin de l'exécution de la procédure, on affichera un message pour identifier ce qui a été fait.

Remarque : En Python, on déclare une procédure à l'aide du mot réservé *def* suivi du prototype de la procédure. Cette ligne ce termine par *:*.

Ensuite, c'est l'indentation qui délimite le contenu de cette procédure.

Remarque : Il en est de même pour une fonction, celle-ci comportera le mot réservé *return* qui permettra à l'issue du traitement de retourner le contenu souhaité.

```
In [3]: def f_init(bot,pos_initiale) :
            i=0
            for m in bot.motors:
                m.moving_speed=30
                m.compliant=False
                m.goal_position=pos_initiale[i]
                i=i+1
            print "la vitesse de mouvement a été mise à jour à ", bot.m1.moving_spe
```

0.0.4 Quelques remarques

- La variable i est ici une variable locale
- bot et pos_initiale sont les deux arguments de la fonction. **Suivant la place de cette procédure, avant ou après l'instanciation du robot, elle aura besoin de l'argument bot ou non.**
- Ici, lors de l'appel de la procédure, poppy est une variabale globale et du fait des caractéristiques du robot (synchronisation), il est inutile de le passer en argument à moins d'avoir plusieurs robots instanciés.

0.0.5 Tester votre procédure

Faire fonctionner votre procédure avec poppy et *pos_init=[0,-90,30,0,60,0]* puis avec *[30,-60,30,-30,60,20]*

```
In [13]: f_init(poppy,pos_init)
         print(poppy.m1.moving_speed)

la vitesse de mouvement a été mise à jour à  30
30
```

0.0.6 Une nouvelle procédure f_init2

Définir une nouvelle procédure f_init dont le prototype est f_init2(bot,pos_initiale,vitesse) et qui permet cette fois d'initialiser la vitesse des moteurs à la valeur vitesse donnée en argument.

```
In [4]: def f_init2(bot,pos_initiale,vitesse) :
            i=0
            for m in bot.motors:
                m.moving_speed=vitesse
                m.compliant=False
                m.goal_position=pos_initiale[i]
                i=i+1
```

QUESTION :
Expliquer pourquoi deux porcédures ne “peuvent” pas avoir le même nom.

- Si elles avaient le même nom, l'exécution du bloc d'instructions permettant de la définir la seconde fois écraserait alors la définition première de la fonction. Il n'en resterait alors plus qu'une seule, la dernière.
- Ce n'est pas le nombre d'arguments qui va définir une fonction mais son nom.

Expliquer le rôle de la procedure définie ci-dessous. Par son appel, elle va permettre de mettre tous les moteurs du robot *bot* dans l'état *compliant* c'est à dire le rendre souple/mobile.

```
In [15]: def f_bouger_a_la_main (bot):
             for m in bot.motors :
                 m.compliant=True
```

Vérifier votre réponse en la testant.

```
In [16]: f_bouger_a_la_main(poppy)
```

Expliquer ce que doit faire la fonction f_pos_cible.
La fonction f_pos_cible va enregistrer la position courante de tous les moteurs du robot dans une liste et la retourner.

```
In [17]: def f_pos_cible (bot) :
             f_pos_cible=[]
             for m in bot.motors :
                 f_pos_cible.append( m.present_position )
             return f_pos_cible
```

0.1 Défi

On veut pouvoir créer un mouvement d'une position de départ à une position d'arrivée. Pour cela :

1. on va initialiser les positions de départ et d'arrivée
2. Faire bouger les moteurs un par un de la position de départ à la position d'arrivée.
3. Pendant toute la durée du mouvement, la led du moteur doit être rouge une fois le mouvement fini, elle doit passer au vert.

```
In [18]: pos_Arrivee=f_pos_cible(poppy)

In [5]: pos_arrivee=[30.65, -34.46, 63.49, -30.94, -86.36, -31.82]

In [19]: pos_depart=f_pos_cible(poppy)

In [6]: pos_depart=[3.37, -98.39, 80.79, -1.32, 11.58, -15.98]

In [7]: import time
```

```
In [8]: def f_mouv(bot,posD,posA):
            f_init2(poppy,posD,50)
            i=0
            time.sleep(3.0)
            i=0
            for m in poppy.motors:
                m.goal_position=posA[i]
                m.led='red'
                time.sleep(2*abs(posA[i]-posD[i])/m.moving_speed)
                i=i+1
                m.led='green'


In [9]: f_mouv(poppy,pos_depart,pos_arrivee)

In [10]: f_mouv(poppy,pos_arrivee,pos_depart)

In [ ]:
```