TP1_mouvement_et_fonctions_eleve_vf

February 6, 2017

```
In [ ]: 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 *poppy.m1* à *poppy.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 [ ]:
```

0.0.2 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 [ ]: def f_init(bot, pos_initiale):
            #debut du corps de la procédure
```

0.0.3 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 [ ]: f_init(poppy, pos_init)
```

Tester : Mettre la vitesse du moteur m1 du robot à 50°/s puis exécuter la procédure *f_init* avec la position initiale de votre choix. Pour finir afficher la vitesse du moteur m1 du robot.

QUESTION : Que remarquez-vous ? Expliquer.

0.0.4 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 [ ]: def f_init2(bot, pos_initiale, vitesse):
            # début du corps de la procédure
```

QUESTION : Expliquer pourquoi deux procédures ne "peuvent" pas avoir le même nom.
Expliquer le rôle de la procédure définie ci-dessous.

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

Vérifier votre réponse en la testant.

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

Expliquer ce que doit faire la fonction *f_pos_cible*.

```
In [ ]: 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 [ ]: # initialiser la position de départ et la sauver dans la liste pos_dep
```

```
In [ ]: # initialiser la position d'arrivée et la sauver dans la liste pos_arr
```

```
In [ ]: import time
        def f_mouv(bot, pos_D, pos_A):
            # corps de la procédure
            # Pourquoi la commande suivante va-t-elle permettre de temporiser
            # suffisement longtemps pour terminer le mouvement ?
            # time.sleep(2 * abs(pos_A[i] - pos_D[i]) / m.moving_speed)
```

Tester votre procédure avec l'instruction suivante :

```
In [ ]: f_mouv(poppy, pos_dep,pos_arr)
```