Modélisation & Robotique - TP semaine 3

29 septembre 2016

Ce TP devra être déposé sur l'ENT à la fin de la séance.
Les exercices de la section 1 (Révisions du cours) ne sont pas à rendre.
Les exercices des sections 2 et 3 devront être mis dans un seul fichier.
Le nom de ce fichier sera composé des membres du binome (par exemple `Dupont_Martin.py`).

1 Révisions du cours

1.1 Quelle est l'erreur ?

Chaque petit programme suivant comporte une ou plusieurs erreurs. Testez ces code sur votre machine, et expliquez le message d'erreur que vous obtenez. Le cas échéant, proposez une correction de ces programmes.

1. Erreur de jeunesse :
```
age = 3
if age = 18
    print("tu es jeune !")
```
2. Effet :
```
3 = 1 + 2
```
3. Bonne note.
```
note = 14
bonus = 2,5
note_finale = note + bonus
```
4. Salut à toi !
```
print(bonjour)
```

1.2 Types.

Si on tape `type(1)` alors le REPL Python nous affiche : `<type 'int'>` pour nous signifier que 1 est du type *int* (c'est à dire entier).

Tapez les instructions suivantes et expliquez ce que vous obtenez :

```
>>> type(3 + 4 * 2)
>>> type(a)
>>> type("a")
>>> type("1")
>>> type([1, 'a'])
>>> a = 10
```

```
>>> type(a)
>>> print([a])
>>> a = "hello"
>>> type(a)
>>> def f():
>>>     print("bonjour")
```

```
>>> def g():
>>>     return "bonjour"
>>> type( f() )
>>> type( g() )
```

1.3 Itérations.

Lisez le programme suivant. Qu'est ce qui sera affiché ?

```
panier = ["le chou", "les carottes", "les pommes de terre", "les poireaux"]
n = len(panier)
print(n)

for i in range(n):
    print("eplucher et laver ",panier[i])
# # # # # # # # # # # # # # # # # # # # # # # # #
L = ["la soupe", "le pain", "une graine", "un abricot"]

for j in range( len(L) ):
    legume = L[j]
    print("mange " , legume, " c'est bon")

# # # # # # # # # # # # # # # # # # # # # # # # #
L2 = []
for j in range( len(L) ):
    legume = L[j]
    if legume[0] == 'u':
        L2 = L2 + [legume]

print(L2)
```

2 Petits programmes (à rendre)

Dans le fichier que vous créerez avec Pyzo, vous répondrez aux questions suivantes, en vous inspirant de la section 1.3 (c'est à dire que vous utiliserez des listes).

2.1 J'aime le chou !

Faire un programme qui affiche :

```
Je regarde le chou
Je regarde la salade
Je mange le chou
Je mange la salade
```

Indice : il faut créer deux listes. Une liste contenant `['regarde','mange']` et une liste contenant `['le chou','la salade']`, et il faut utiliser deux `for` imbriqués.

2.2 Et le navet !

Faire un programme qui affiche :

```
Je regarde le chou
Je mange 1 chou
Je mange 2 chou
Je mange tous les chou
Je regarde le navet
```

```
Je mange 1 navet
Je mange 2 navet
Je mange tous les navet
```

Indice : Créez une liste `['regarde le','mange 1','mange 2','mange tous']`.

2.3 Mais pas le brocolis

Modifier le programme précédent pour qu'il affiche (il faudra utiliser l'instruction if) :

```
Je regarde le chou
Je mange 1 chou
Je mange 2 chou
Je mange tous les chou
Je regarde le brocolis
Je ne mange pas un brocolis
Je regarde le navet
Je mange 1 navet
Je mange 2 navet
Je mange tous les navet
```

2.4 Quant aux haricots...

Ecrivez un programme qui affiche :

```
Je mange 1 chou
Je mange 2 chou
Je mange 1 brocolis
Je mange 1 navet
Je mange 2 navet
Je mange 3 navet
Je mange 1 haricot
Je mange 2 haricot
Je mange 3 haricot
Je mange 4 haricot
Je mange 5 haricot
```

Indice : vous pouvez faire une boucle utilisant range(n) en changeant la valeur de n selon le légume.

3 Dans notre micro-monde (à rendre)

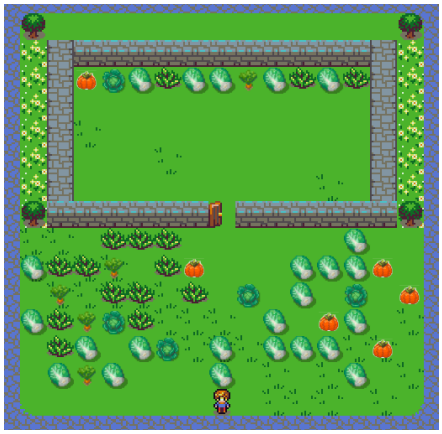

FIGURE 1 – Le jardin initial

Aujourd'hui dans notre micro-monde, il y a un jardin entouré d'un enclos de pierres,

qui occupe la moitié du terrain. Le personnage principal est un jardinier, et sa tâche sera se créer un beau jardin potager. Au tout début, le micro-monde ressemble à celui de la figure 1.

On voit que le jardinier a déjà disposé dans son jardin une rangée de 11 légumes d'une façon particulière. Pour l'instant, la plupart des légumes sont éparpillés hors du jardin. Le jardinier devra les ramasser et les placer à l'intérieur du jardin.

Plus précisément, le jardinier souhaite disposer encore 4 rangées de légumes exactement comme la première rangée. Lorsqu'il aura fini, le micro-monde devra ressembler à celui de la figure 2 :

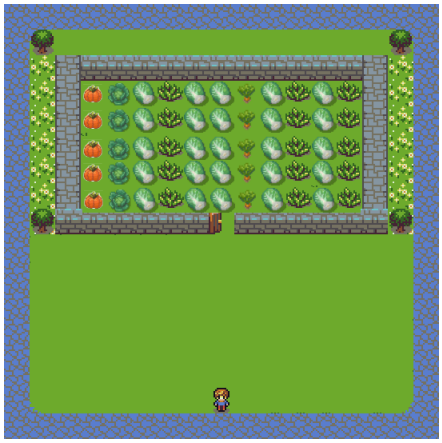

FIGURE 2 – Jardin bien rangé

Tout programme python utilisant ce micro-monde devra commencer ainsi :

```
from gardenworld import *
init('info2_1')
```

Par ailleurs, vous avez accès aux instructions suivantes :

```
avance()        # avance d'une case
tournegauche()  # pivote d'un quart de tour
tournedroite()  # pivote d'un quart de tour (autre sens)
cherche()       # cherche s'il y a un légume sous les pieds du jardinier
ramasse()       # ramasse le légume sous les pieds du jardinier
depose()        # dépose l'un des légumes ramassés
```

Ces instructions ont des abbréviations : `av(); tg(); td(); ra(); ch(); dp()`

Le jardinier peut ramasser autant de légumes qu'il veut avant de les déposer (il a un grand panier).

3.1 Exercices sur les listes

Dans un terminal, lancez python et faites :
— Créez une liste vide que vous nommerez `Panier`,
— positionez le jardinier sur un légume, et écrivez l'instruction qui rajoute à la liste `Panier` le nom de ce légume.

3.2 Plantons dans notre jardin

Dans ces exercices, évitez d'utiliser des variables globales, ce n'est pas nécessaire.
— Créez une fonction `printRangee()` qui va aller tout droit jusqu'au prochain obstacle, en affichant les noms des légumes sous les pieds du jardinier. Si le jardinier est positionné en haut à gauche de son jardin, orienté vers l'Est, l'appel à cette fonction affichera donc
```
citrouille
salade
chou
...
```
— Créez une fonction `ObserveRangee()` qui va aller tout droit jusqu'au prochain obstacle, en observant les légumes sous les pieds du jardinier. Cette fonction devra renvoyer une liste des légumes observés dans l'ordre. Si le jardinier est positionné en haut à gauche de son jardin, l'appel à cette fonction renverra la liste `['citrouille', 'salade', 'chou', ...]`
— Créez une fonction `fonceRamasse()` qui déplace le personnage tout droit jusqu'au prochain obstacle, et ramasse tous les légumes sur son passage.
— Créez une fonction `ramasseTout()` qui ramasse l'ensemble des légumes hors de l'enclos.
— Créez une fonction `RecopieRangee(listeLegumes)` qui prend en paramètre une liste de noms de légumes, et qui dépose chaque légume de la liste dans l'ordre.
Indice : l'appel à la fonction `depose('chou')` *dépose un chou, si le jardinier en a ramassé un au préalable. Cela fonctionne pour tous les autres légumes, bien entendu.*
— Assemblez le tout pour que le jardiner finalise son jardin à l'intérieur de l'enclos.

3.3 Exercices bonus

— Créez la fonction `ramasseEnregistre()` qui, s'il y a un légume sous les pieds du jardinier, ramasse le légume et renvoie une liste composée du nom du légume. S'il n'y a rien sous ses pieds, cette fonction renvoie la liste vide. Testez-là pour vérifier qu'elle fonctionne bien.
Par exemple, s'il y a une salade sous ses pieds, l'appel à cette fonction renvoie la liste `['salade']`
— Créez une fonction `fonceRamasseEnregistre()` qui fasse le même travail que `fonceRamasse`, et en plus renvoie la liste des noms des légumes ramassés (dans

l'ordre de ramassage).
Indice : cette fonction ressemble à fonceRamasse, mais utilise aussi la fonction ramasseEnregistre

— Créez une fonction `ramasseToutEnregistre()` qui ramasse l'ensemble des légumes hors de l'enclos, et renvoie la liste des noms de légumes dans l'ordre de ramassage.

— Créez la fonction `typeDeLegumes(listeLegumes)` qui prend en parametre une liste de légumes et renvoie une liste dans laquelle chaque type de légume n'apparait qu'une fois.
Par exemple : `typeDeLegumes(['chou', 'salade', 'chou', 'citrouille', 'citrouille'])` devra renvoyer `['chou', 'salade', 'citrouille']`
Indice : rappelez vous qu'on peut utiliser l'instruction `in` *pour tester si un élément est présent dans une liste. Par exemple* `'chou' in ['salade','chou']` *sera vrai.*

— Créer une fonction `Compte(ListeLegumes,nomLegume)` qui comptera le nombre de légumes de type `nomLegume` dans la liste `ListeLegumes`.

— Créez une fonction `resume(ListeLegumes)` qui affiche le nombre de légumes de chaque type.

— Créez une fonction `ramasseToutResume()` qui ramasse l'ensemble des légumes hors de l'enclos, et affiche le nombre de légume de chaque type.