TD SEMAINE 6 - DICTIONNAIRES - SIGNATURES

1. Révisions sur les Listes et Dictionnaires

On a une liste de prénoms `Prenoms=['jean',...]` et une liste de numeros de téléphone `Tel=['0662243216',...]`.

(1) Créez deux dictionnaires `dPrenom` et `dTel` à partir de ces listes. Le premier permettra d'accéder aux numéros de téléphone à partir des prénoms, et le deuxième permettra d'accéder aux prénoms à partir des numéros de téléphone.

(2) Supposons maintenant que les listes `Prenoms` et `Tel` aient été détruites. Recréez ces listes à partir du dictionnaire `dPrenom`

(3) A partir du dictionnaire `Prenom`, construisez un dictionnaires `PrenomOk` comportant uniquement les prenoms ayant un numero de telephone comportant 10 caractères.

Correction.

```
# question 1
dPrenom=dict()
dTel=dict()
for i in range(len(Prenoms)):
    dPrenom[Prenoms[i]]=Tel[i]
    dTel[Tel[i]]=Prenoms[i]

# question 2
Prenoms = []
Tel = []
for p,t in dPrenom.items():
    Prenoms = Prenoms + [p]
    Tel = Tel + [t]

# question 3
PrenomOk = dict()
for p,t in dPrenom.items():
    if len(t) == 10:
        PrenomOk[p] = t
```

2. Simulateur de robot

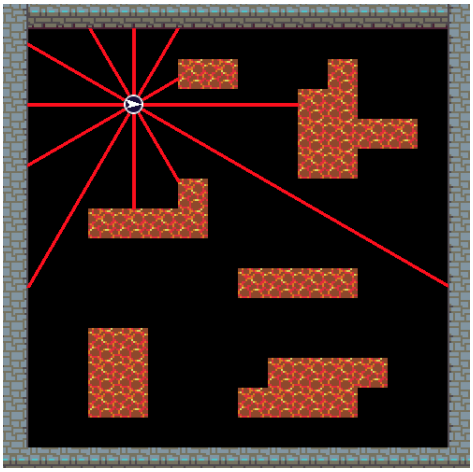
2.1. **Introduction.** On se place dans le monde du simulateur de robots. Les instructions suivantes sont disponibles :

```
av(x) # avance de x pixels
td(d) # tourne a droite de d degrés
tg(d) # tourne a gauche de d degrés
telemetre() # active le capteur télémétrique laser, et renvoie la distance
            # en pixels à l'obstacle le plus proche en face du robot.
```

On appelle « signature » une liste de relevés du télémètre, obtenue par le robot (en simulation ou non) en tournant successivement sur lui-même avec un angle constant, pendant deux tours au moins.

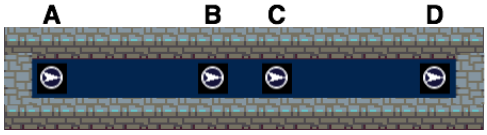
Attention ! le télémètre n'est pas précis et renvoie des mesures bruitées ! ! !

Exemple 1. Dans la situation représentée dans l'image ci-dessous, le robot initialement orienté vers l'Est a effectué deux tours complets sur lui même, et activé le télémètre (dont le laser est représenté par un trait rouge) tous les 30 degrés. Les relevés obtenus sont les suivants : `signature=[165.0, 237.9, 101.0, 148.8, 182.0, 143.5, 130, 135.8, 132, 121, 136, 60, `*`164.9, 236.9, 100.6, 147.8, 181.2, 142.3, 129.2, 134.0 , 133.0, 121.9,135,65 ,167`*`]`.
La deuxième moitié de la liste, qui correspond au deuxième tour sur lui-même, est écrite en italique.



Exemple 2. Dans l'image ci-dessous, le robot se déplace dans un couloir d'environ 200 pixels de long. Il prélève une signature dans chaque position A,B,C et D. Dans les positions A et D, il se trouve à environ 180 pixels de l'extrémité opposée du couloir. Pour chaque signature, on active le télémètre tous les 90 degrés. Cela donne les signatures suivantes :

```
#  Est Sud Ouest Nord Est   Sud Ouest Nort Est
A=[180,5.2   8,   6,   181, 5,  8.2,  3.8, 181]
B=[118,5.2,  70.1,5,   118, 5,  72,   5,   117]
C=[70, 5.1,  117, 4,   69.3,5,  116,  4,   71]
D=[8,  4.9,  182, 5,   8,   5,  181,  4,   7]
```



Le but des exercices qui suivent est de permettre au robot de comparer plusieurs signatures, et de déterminer si elles sont similaires ou non, si elles ont été vraissemblablement prises aux mêmes endroits ou non. Ainsi, si l'on suppose que le robot dispose d'une bibliothèque de signatures prélevées dans différentes positions, en comparant sa signature actuelle avec celles de la bibliothèque, *le robot pourra déterminer où il se trouve probablement.*

2.2. **Exercices.**

(1) Ecrire une fonction `creeSignature(d)` qui crée une signature en effectuant au moins deux tours, en tournant de `d` degrés à chaque fois.

(2) Soient deux signatures `d1` et `d2`. Ecrivez une fonction `distance` qui calcule la somme des valeurs absolues des différences . Par exemple, `distance([3,10],[5,6])` renvoie `6` car $|5-3|+|10-6|=6$.

Remarque. Grace à cette fonction, on peut déterminer si deux signatures ont été prises au même endroit, à condition que l'angle initial du robot ait été le même.

(3) Ecrivez une fonction `distDecal(s1,s2,k,n)` qui calcule la distance des `n` premiers éléments de `s2` avec la sous-liste de `s1` allant du $k^{ième}$ au $k+n^{ième}$ exclus.

(4) Ecrivez une fonction `distBestDecal(s1,s2,n)` qui renvoie la valeur de `k` pour laquelle `distDecal(s1,s2,k,n)` est la plus petite.

(5) Ecrivez la fonction `DistanceRobuste` qui renvoie la distance calculée avec `distDecal` avec la meilleure valeur de `k`. On choisira comme valeur de `n` la longueur de la liste `s1` divisée par 2.

Remarque. Grace à `DistanceRobuste`, on peut déterminer si deux signatures ont été prises au même endroit, *même si l'angle initial du robot n'était pas le même !*

(6) On suppose que `dic` est un dictionnaire de signatures, dont les étiquettes représentent les noms des endroits où ces signatures ont été prélevées. Par exemple, `dic={'coin_haut_gauche':[13.4,33.9,...],'centre':[86.4,102.5,...]}`. Ecrivez une fonction `Similarite` qui prend en paramètre ce dictionnaire et une signature, et renvoie un dictionnaire associant à chaque endroit, par exemple : `{'coin_haut_gauche':28.4,'centre':54.2,...}`.

Remarque. Lors de l'appel aux fonctions `distBestDecal` et `distDecal`, la valeur de `n` choisie sera la moitié de la longueur de la liste.

(7) Ecrivez une fonction `OuSuisJe` qui prend en paramètre le dictionnaire de signatures `dic`, une signature `s` et un seuil `t`, et qui renvoie la liste des endroits mentionés dans `dic` dont la distance robuste à `s` est inférieure à `t`.

Correction.

```
# question 1

def creeSignature(d):
    angle=0
    R = [telemetre()]
    while angle < 360*2:
        td(d)
        angle = angle + d
        R = R + [telemetre()]
    return R

# question 2

def distance(s1,s2):
    d = 0.0
    for i in range(len(s1)):
        d = d + abs(s1[i]-s2[i])
    return d

# question 3
def distDecal(s1,s2,k,n):
    return distance(s1[k:n+k],s2[:n])

# question 4
def distBestDecal(s1,s2,n):
    k  = 0
    bestd = distDecal(s1,s2,0,n)
    for i in range(n):
        d = distDecal(s1,s2,i,n)
        if d < bestd:
            bestd = d
            k = i
    return k

# question 5
def distanceRobuste(s1,s2):
    n = len(s1)//2   #rappel: l'operateur // est une division entiere
    k = distBestDecal(s1,s2,n)
    return distDecal(s1,s2,k,n)

# question 6
def Similarite(dic,s):
    v = dict()
    for e in dic:
```

```
        v[e] = distanceRobuste(s,dic[e])
    return v

# question 7
def OuSuisJe(dic,s,t):
L = []
    v = Similarite(dic,s)
    for e in v:
        if v[e] < t:
            L = L + [e]
    return L
```