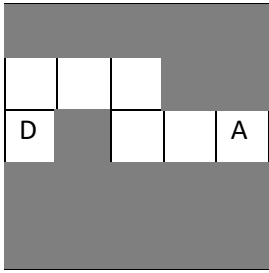


# Recherche de chemin dans un labyrinthe

Fichier source disponible : [labyrinthe-eleve.py](#)

On considère le « labyrinthe » ci-dessous :



L'objectif est de se déplacer de la case D (départ) pour arriver à la case A (arrivée)

Le labyrinthe est codé sous la forme d'une liste de liste nommée **lab**.

Une case blanche est codée par un 0, une case noire par un 1, la case départ par un 2 et la case arrivée par un 3.

Le labyrinthe ci-contre **lab** est donc codé ainsi :

```
[ [1,1,1,1,1],  
  [0,0,0,1,1],  
  [2,1,0,0,3],  
  [1,1,1,1,1],  
  [1,1,1,1,1],]
```

La case **lab[1][4]** est une case noire ou blanche ? \_\_\_\_\_

1/ écrire une fonction **fonction1** qui reçoit **lab** et les coordonnées **L et C**, et détermine si cette case est la case d'arrivée. Testez votre fonction.

def fonction1(lab, L,C) :

2/écrire une fonction **fonction2** qui détermine si les coordonnées l et m sont valides (c'est-à-dire si la case appartient au tableau. Elle renverra par exemple True pour les coordonnées 4,3 et False pour les coordonnées 1,6) . Testez votre fonction.

3/ écrire une fonction **trouve\_D** qui détermine les coordonnées de la case de départ. Elle retournera ces coordonnées sous forme d'une liste de deux entiers. Testez votre fonction.

4/ écrire une fonction **trouve\_A** qui détermine les coordonnées de la case d'arrivée. Elle retournera ces coordonnées sous forme d'une liste de deux entiers. Testez votre fonction.

Maintenant que l'on sait trouver les case de départ et d'arrivé, nous allons chercher le chemin à parcourir. L'objectif est d'écrire une liste qu'on nommera **chemin** qui listera la suite de coordonnées à suivre pour arriver à A

5/ vérifions d'abord qu'on a compris le problème :

Ecrivez à la main la valeur de chemin qu'on doit obtenir :

chemin = [ [2,0], ...

6/ soit le code ci-dessous :

```
def trouver_prochaine_case (lab,chemin) :  
    case_actuelle = chemin [-1]  
    L=case_actuelle[0]  
    C=case_actuelle[1]  
    if fonction2 (lab, L+1, C) :  
        if lab [L+1][C]==0 :  
            chemin.append( [L+1,C] )  
            print("yep")  
            return chemin  
    if fonction2 (lab, L, C+1) :  
        if lab [L][C+1]==0 or lab [L][C+1]==3 :  
            chemin.append([L,C+1])  
            print("yo")  
            return chemin  
    if (lab [L-1][C]==0 or lab [L-1][C]==3) and L-1>=0:  
        chemin.append([L-1,C])  
        print("yeah")  
    return chemin
```

6-1/ rappeler le rôle de la fonction **fonction2**

6-2/ que retourne **trouver\_prochaine\_case** si on lui donne en argument **lab** et **chemin=[2,0]** ? et quel mot sera affiché (yep, yo ou yeah) ?

6-3/ changer ces trois mots (yep, yo, yeah) par ce qui leur correspond entre haut, bas et droite.

yep :

yo :

yeah :

#### ATTENTION ! PIEGE A ELEVE (ET PLUS GENERALEMENT, PIEGE A PROGRAMMATEUR IMPRUDENT)

Souvenons-nous : lorsqu'on passe un argument dans une fonction, c'est la **valeur de la variable** et non la variable qu'on envoie. C'est ce qu'on appelle un **passage par valeur**. Autrement dit, les changements sur cette variable dans la fonction resteront **local à la fonction**. LA VALEUR DE CETTE VARIABLE A L'EXTERIEUR DE LA FONCTION NE BOUGERA PAS.

En Python, c'est vrai pour les variables de types de base (numérique, booléen, caractères)

**Par contre** : pour les variables construites ( les listes par exemple), le **passage de l'argument se fait par référence**. Cela signifie que si vous passez en argument une liste que vous modifiez dans la fonction, ELLE SERA AUSSI MODIFIEE EN DEHORS DE LA FONCTION !

Revoir le cours sur la portée des variables pour plus de détails.

7/ rassemblons et complétons notre code pour faire l'entier du boulot :

Dans notre fichier complet, il nous faut :

- ➔ Les définitions de nos fonctions (*fonction*, *fonction2*, *trouver\_prochaine\_case*, *trouve\_A*, *trouve\_D*)
- ➔ La déclaration de *lab*
- ➔ Affecter à une variable qu'on nommera *depart* les coordonnées de départ
- ➔ Affecter à une variable qu'on nommera *arrive* les coordonnées de la case d'arrivée
- ➔ Déclarer *chemin* en liste vide
- ➔ Ajouter à *chemin* la valeur des coordonnées de la case de départ
- ➔ Faire une boucle pour ajouter à *chemin* les « cases suivantes » jusqu'à ce qu' on arriv à l'arrivé.
- ➔ Afficher à l'écran *chemin*

A vous de jouer 😊