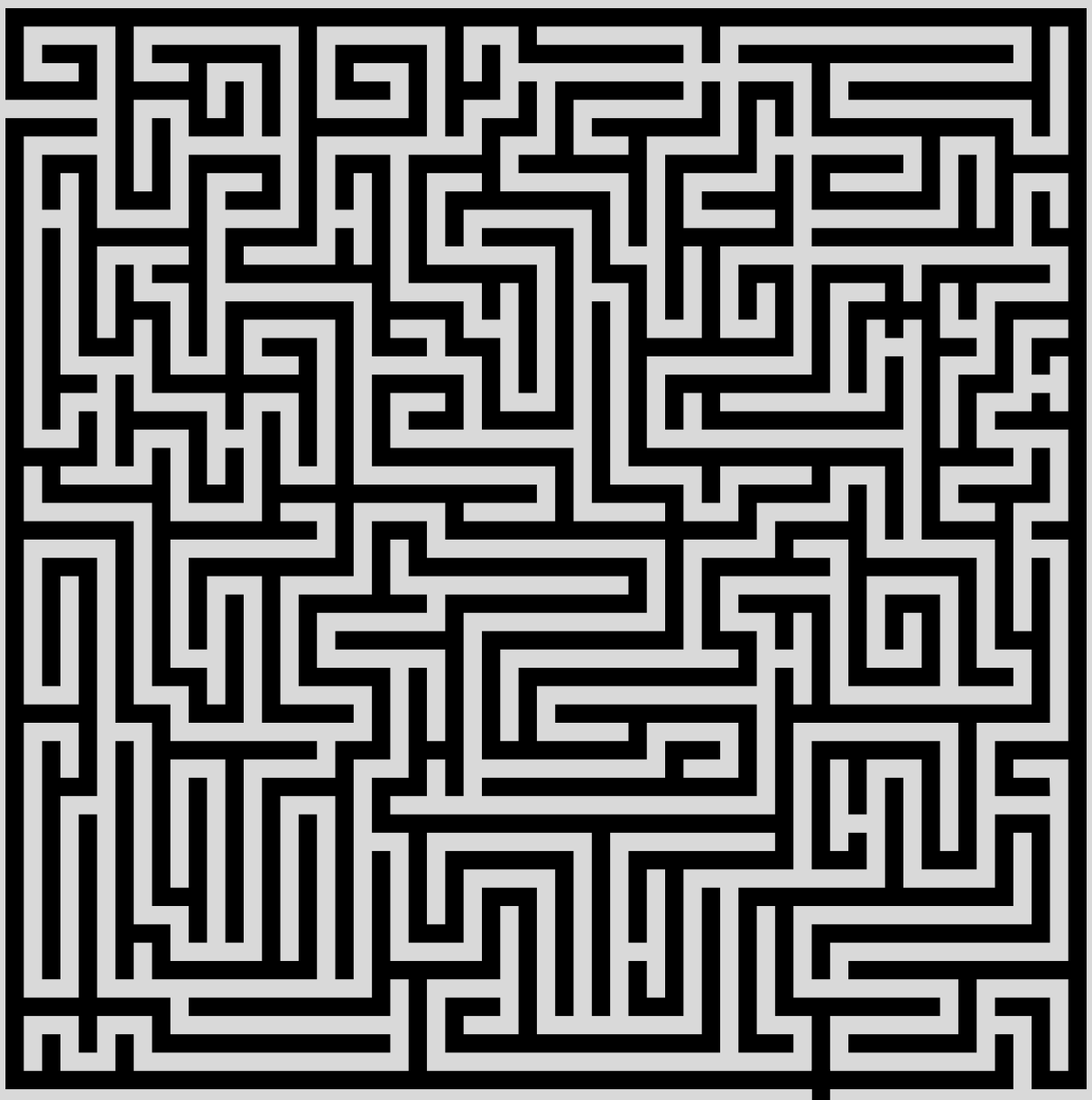


RAPPORT INFO3A

KHMAROU FATIMA-EZZAHRA
YOUSSEF EL AIDOUDI

SOMMAIRE:

1. Modélisons de la structure:

Pour modéliser informatiquement cette grille, nous pouvons utiliser une liste bidimensionnelle pour stoker chaque cellule.

a. Modélisation informatique:

Pour modéliser une telle grille on a initialiser La variable Murs est une matrice qui représente le labyrinthe. Chaque cellule contient une liste de quatre éléments (murs) indiquant s'il y a un mur à gauche, en haut, à droite et en bas respectivement.

- La matrice visite est utilisée pour marquer les cellules visitées pendant la génération.
- x et y représentent les coordonnées actuelles dans le labyrinthe.
- somme_visite est utilisée pour vérifier si toutes les cellules ont été visitées.

**La boucle principale continue jusqu'à ce que toutes les cellules soient visitées (somme_visite == (taillex * tailley)).

- Pour chaque cellule, les directions possibles sans mur sont évaluées et stockées dans la liste options.

**Choix aléatoire de la direction :

- Si aucune direction possible, la fonction revient à la cellule précédente.
- Sinon, une direction est choisie aléatoirement parmi les directions possibles.

**Retrait des murs :

- Les murs sont retirés dans la direction choisie, et la cellule opposée est mise à jour en conséquence.
- Les coordonnées actuelles sont mises à jour, et la cellule est marquée comme visitée.

**Stockage des informations sur les nœuds :

Les informations sur la nouvelle cellule sont stockées dans la liste visite_noeuds.

**Retour de la matrice représentant le labyrinthe :

Une fois que toutes les cellules ont été visitées, la fonction renvoie la matrice Murs représentant le labyrinthe généré.

2.Construisons d'un labyrinthe parfait

L'affichage de la grille sur pyplot:

Pour l'affichage on a initialiser

```
startx = -380
starty = -startx
```

qui définissent les coordonnées de départ pour le dessin.

```
taille_case = (2 * (-startx)) / taillex
```

représentant la taille de chaque cellule du labyrinthe.

```
turtle.penup()
turtle.goto(startx, starty)
turtle.pendown()
turtle.goto(-startx, starty)
turtle.goto(-startx, -starty)
turtle.setheading(0)
```

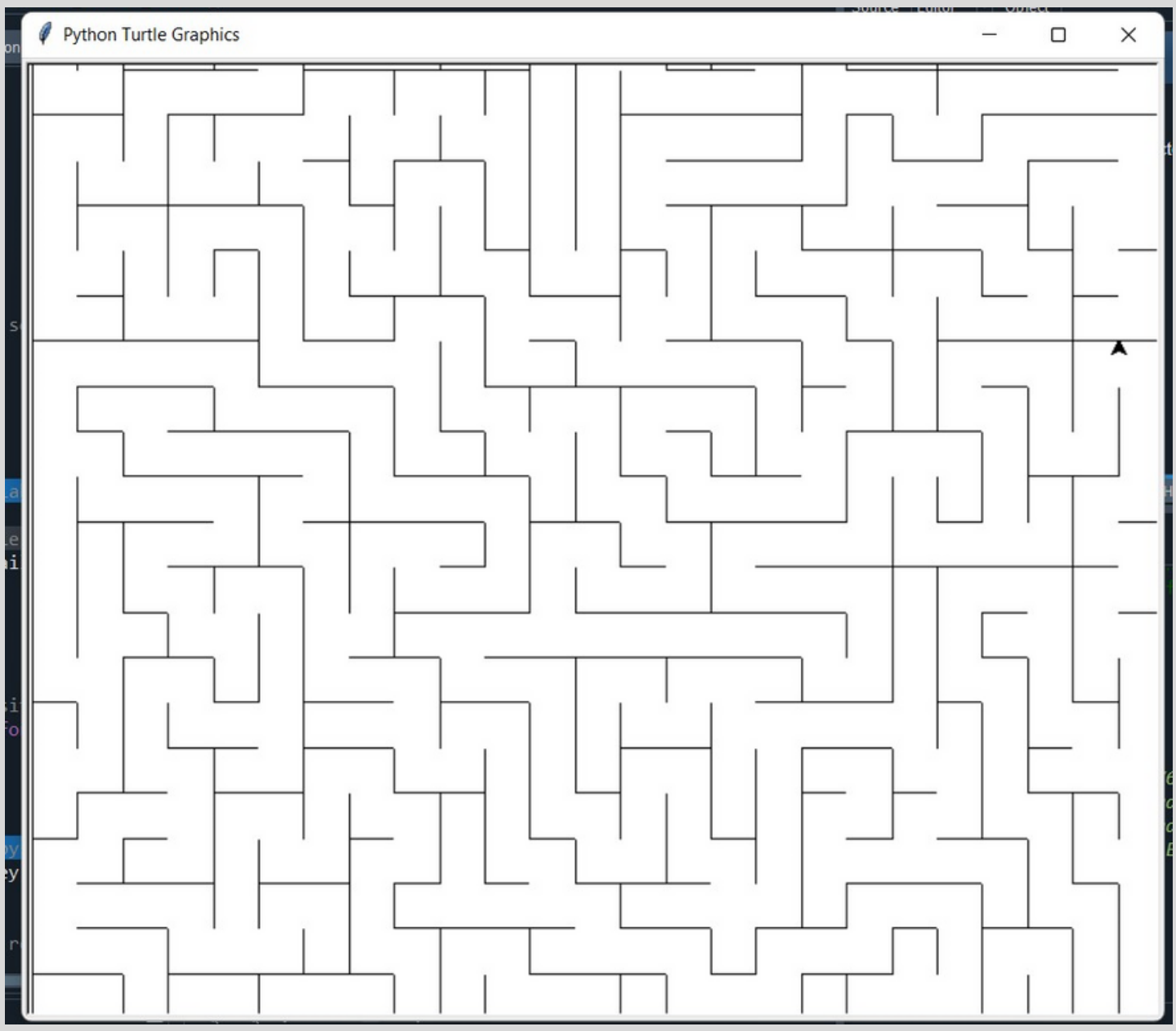
La tortue est positionnée pour dessiner les bords du labyrinthe.

Pour le dessin des murs horizontaux:

- Une boucle parcourt les lignes du labyrinthe (for y in range(tailley)).
- La tortue est déplacée à la position correcte pour dessiner les murs horizontaux de la ligne.
- Pour chaque case dans la ligne, si le mur en bas est présent (Murs[y][x][3] == 1), la tortue dessine le mur en se déplaçant vers l'avant d'une distance égale à taille_case. Sinon, le stylo est levé.

Pour le dessin des murs verticaux ::

- La direction de la tortue est changée de 90 degrés pour dessiner les murs verticaux (turtle.left(90)).
- Une boucle parcourt les colonnes du labyrinthe (for x in range(taillex)).
- La tortue est déplacée à la position correcte pour dessiner les murs verticaux de la colonne.
- Pour chaque case dans la colonne, si le mur à gauche est présent (Murs[y][x][0] == 1), la tortue dessine le mur en se déplaçant vers l'avant d'une distance égale à taille_case. Sinon, le stylo est levé.



3.Parcours de notre labyrinthe

pour ajuster la direction en fonction d'un paramètre direction et d'une liste binaire binaryvalue on a utilisé la fonction ajuster_cap

en paramètres on a utilisé:

- direction est un entier qui représente la direction actuelle (entre 0 et 3).
- binaryvalue est une liste binaire de longueur 4.
- binarynew est une nouvelle liste binaire initialisée à [0, 0, 0, 0].

Ajustement de la direction :

- La fonction parcourt les indices de 0 à 3 à l'aide de la boucle for a in range(4).
- Elle ajoute les valeurs de binaryvalue dans binaryvalue pour dupliquer les éléments.
- Les éléments de binarynew sont mis à jour en fonction de la direction actuelle.
    - binarynew[0] prend la valeur de binaryvalue[direction + 3].
    - binarynew[1] prend la valeur de binaryvalue[direction].
    - binarynew[2] prend la valeur de binaryvalue[direction + 1].
    - binarynew[3] prend la valeur de binaryvalue[direction + 2].

La technique de la main droite:

c’est une technique pour trouver la sortie d'un labyrinthe consiste à longer systématiquement le mur de droite du parcours, sans le lâcher.

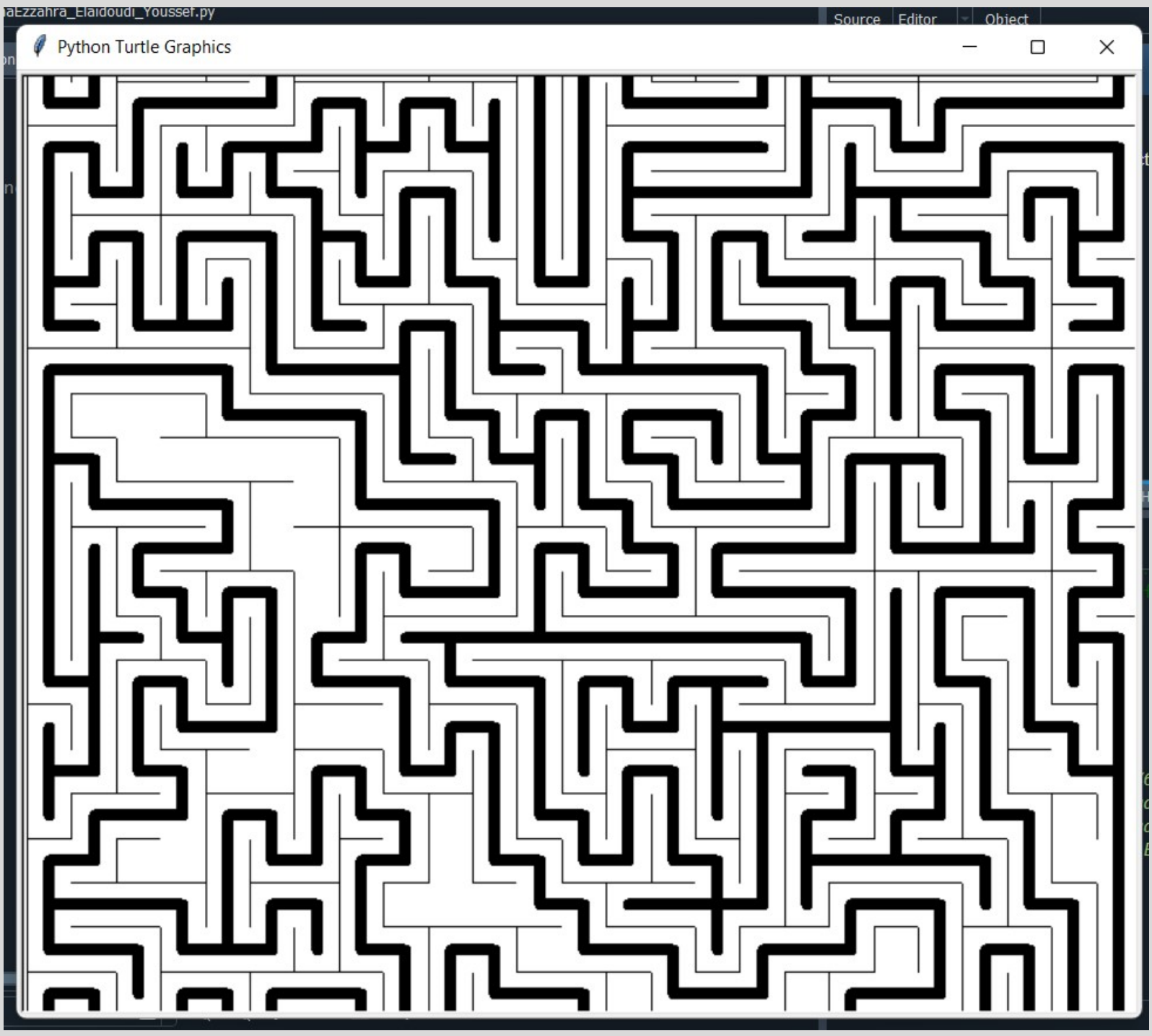
Pour cette technique on a utiliser la fonction suivre_mur cette dernière simule le déplacement d'une entité, représentée par une tortue, à travers un labyrinthe. L'objectif de cette fonction est de guider la tortue le long des murs du labyrinthe jusqu'à ce qu'elle atteigne la sortie.

Voici une explication détaillée de son fonctionnement :

1. Initialisation de la tortue :
    - La tortue est positionnée au centre de la première cellule du labyrinthe.
    - La taille du stylo est configurée en fonction de la taille de la grille (taillegrille), et la couleur du stylo est définie sur noir.
    - La tortue est orientée vers le haut.
2. Initialisation des coordonnées et de la direction :
    - x et y représentent les coordonnées actuelles dans le labyrinthe.
    - direction représente la direction de la tortue (0 pour la droite, 1 pour le haut, 2 pour la gauche, 3 pour le bas).
3. Boucle pour suivre le mur :
    - La boucle continue jusqu'à ce que la tortue atteigne la sortie du labyrinthe (coordonnées (taillex - 1, 0)).
    - La fonction ajuster_cap est appelée pour obtenir les informations sur les murs autour de la tortue, et la direction est ajustée en conséquence.

4.La fonction suivre_mur:

1. Ajustement de la direction :
    - La direction de la tortue est ajustée en fonction des murs autour d'elle.
    - Si le mur est à gauche, la tortue tourne à gauche et ajuste la direction en conséquence.
    - Si le mur est devant, la tortue continue tout droit.
    - Si le mur est à droite, la tortue tourne à droite et ajuste la direction en conséquence.
    - Si le mur est derrière, la tortue fait demi-tour et ajuste la direction en conséquence.
2. Gestion des limites de direction :
    - La direction est ajustée si elle dépasse les limites (inférieure à 0 ou supérieure à 3).
3. Mise à jour des coordonnées :
    - Les coordonnées de la tortue sont mises à jour en fonction de la direction.



Quels sont les inconvénients de cette méthode:

La méthode de la main droite ne garantit pas toujours la recherche du chemin le plus court. En effet, elle peut conduire à des boucles ou à des chemins plus longs dans certains types de labyrinthes. Aussi elle est basée sur le choix d'un seul mur à suivre (à droite). Si ce mur conduit à une impasse, la méthode ne prend pas en compte d'autres options possibles et peut nécessiter un retour en arrière.

4.Bonus

1/noeud_oppose(b, noeud_courant) :

Cette fonction prend une direction b (0, 1, 2, 3) et les coordonnées du nœud courant.
Elle renvoie les coordonnées du nœud opposé en fonction de la direction spécifiée.

2/recherche_Dijkstra(Murs, tailley, taillex) :

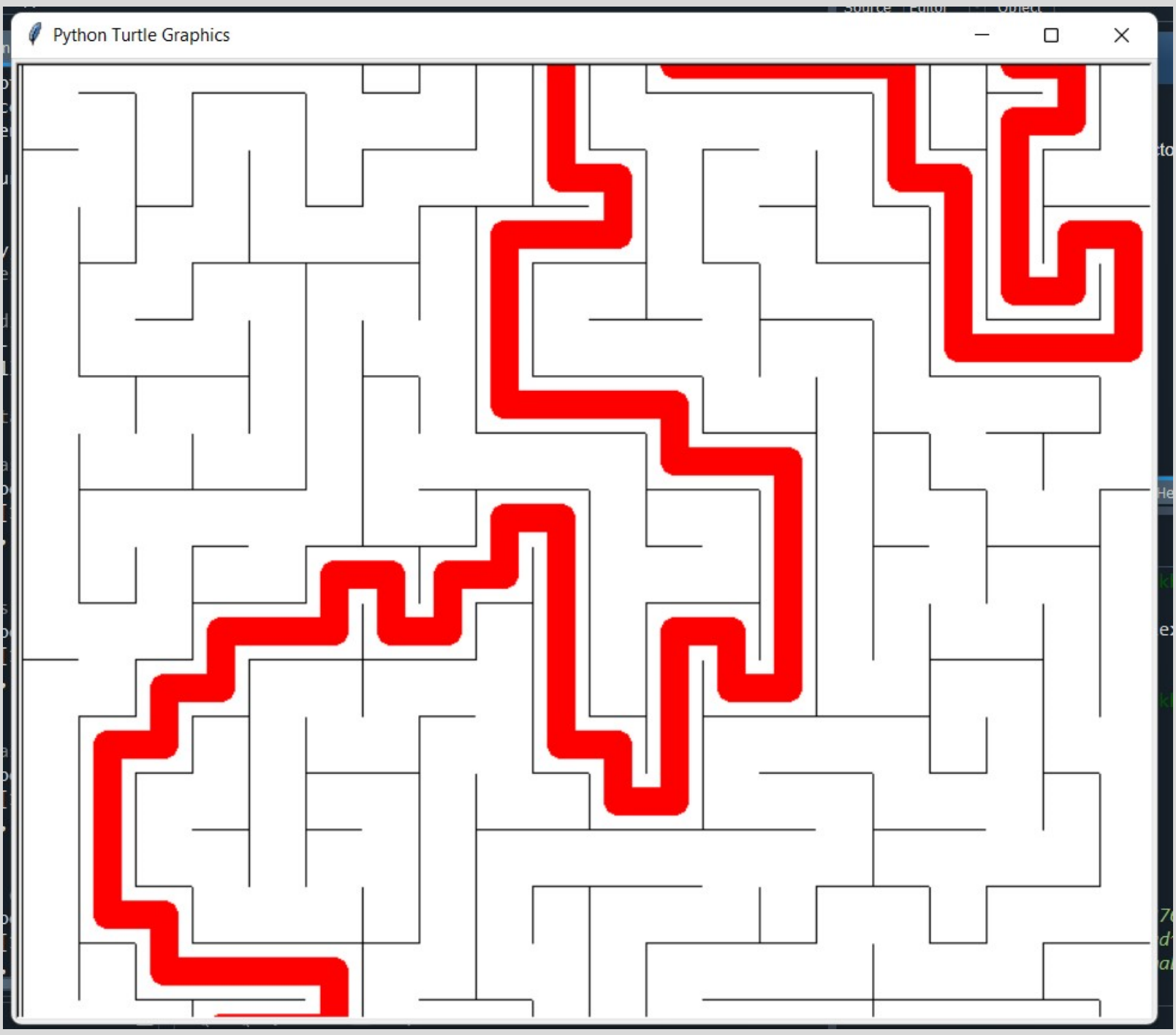
Cette fonction implémente l'algorithme de Dijkstra pour trouver les distances minimales depuis le coin inférieur gauche du labyrinthe jusqu'à chaque nœud atteignable.
Elle retourne une matrice représentant ces distances.

3/chemin_Dijkstra(visite, taillex, tailley, Murs) :

- Cette fonction prend la matrice de distances résultante de l'algorithme de Dijkstra, les dimensions du labyrinthe, et la matrice de murs.
- Elle retourne les coordonnées du chemin le plus court depuis le coin inférieur gauche jusqu'au coin supérieur droit.

4/afficher_chemin(coordonnees_chemin, startx, starty, taillegrille) :

- Cette fonction utilise la bibliothèque Turtle pour afficher visuellement le chemin trouvé.
- Elle prend les coordonnées du chemin, les coordonnées de départ, et la taille de la grille.



5.obstacles rencontré:

Grâce aux cours, aux travaux pratiques et à leurs corrigés, nous avons pu progresser significativement dans notre projet. Nous avons également effectué de nombreuses recherches sur internet pour trouver les méthodes les plus appropriées et logiques susceptibles de nous aider.

Cependant, certaines difficultés se sont présentées, telles que la méthode de la main droite, on a pu suivre le mur gauche mais en changeant les conditions pour suivre celui de droite notre code ne marchais plus.

- Malheureusement, nous avons également rencontré des problèmes que nous n'avons pas pu résoudre, que ce soit en raison d'incompréhensions ou simplement par manque de temps.
- on avait prévu d’ajouter Une technique plus efficace que celle de "la main droite" qui traite la question 4 , cela pour rendre notre programme plus efficace .
- Nous avons rencontré des difficultés pour mettre en place les conditions permettant de définir le point de départ en haut à gauche et la sortie en bas à droite dans le labyrinthe.

Finalement, malgré les défis rencontrés et la contrainte de temps, ce projet a contribué à renforcer notre connaissance du langage Python en général. Nous avons pris soin d'expliquer chaque section de notre code dans ce rapport, sans entrer dans des détails trop spécifiques vus en cours. De plus, nous avons ajouté des commentaires à certaines parties du code pour garantir sa clarté et son organisation. En conclusion, cette expérience nous a incités à approfondir nos connaissances et à explorer de nouvelles méthodes, contribuant ainsi à la maîtrise du module INFO3A.