Ce TT vous introduit les classes en python. On va réaliser une chasse au trésor.

1 Présentation

On définit un graphe comme un ensemble de *sommet* connectés les uns aux autres. Un somme a est connecté à un sommet b s'il est possible de passer de a à b.

2 Mise en place

Question 1

Définir une classe *sommet* ayant pour attribut un *nom* et une liste de *voisins* (vide par défaut), et un booléen *tresor* qui vaudra True si et seulement si ce sommet contient un tresor.

A l'aide de la méthode *repr*, faire en sorte que lorsqu'on print un sommet, son nom et ses voisins soient affichés.

Question 2

Définir une classe *graphe* ayant un attribut *sommets* contenant la liste des sommets du graphe et *nb_tresor* indiquant le nombre de trésors dans le graphe.

Question 3

Ajouter un attribut *graphe* à la classe sommet pour savoir dans quel graphe un sommet se trouve. On modifiera la méthode *init* de la classe *graphe* pour que quand on crée un graphe G, les sommets qu'il contient aient la bonne valeur pour l'attribut graphe.

Question 4

Créer une méthode *genere_tresor* dans la classe graphe qui génère un trésor sur un sommet aléatoire du graphe et mets à jour l'attribut *nb_tresor*. S'il y a déjà un trésor sur le sommet choisi aléatoirement, on ne fera rien.

Question 5

Créer une classe *explorateur* qui a pour attribut une *position* correspondant à un sommet où il est (None par défaut), et un *butin* correspondant au nombre de trésors amassé. On commencera avec 0 butins.

Ajouter à cette classe une méthode *deplacement* qui permet à l'explorateur d'aller sur un sommet voisin de celui où il se trouve. On renverra un message indiquant que le déplacement est impossible si le sommet visé n'est pas accessible.

3 Recherche de trésor

Question 6

Écrire une fonction *recherche_tresor* qui prend en entrée un explorateur, et à chaque étape affiche les sommets accessibles par l'explorateur, et demande à l'utilisateur de choisir où aller. On s'arrêtera dès que l'explorateur aura récupéré un trésor. S'il n'y a pas de trésor dans le graphe, on en génèrera un avant de commencer les recherches.

Question 7

Ajouter un attribut *distance* dans la classe sommet initialisée à -1.

Modifier la méthode *genere_tresor* pour qu'elle mette cette valeur à 0 lorsqu'un trésor est généré sur un sommet.

Question 8

Écrire une fonction *calcul_dist* qui prend en entrée un graphe et calcule les distances au trésor de tous les sommets du graphe. La distance étant définie comme le nombre minimal de déplacements à effectuer pour atteindre le trésor.

Question 9

Écrire une fonction *recherche_ distance* qui fonctionne comme *recherche_ tresor* mais qui à chaque étape indique à l'utilisateur à quelle distance il se trouve d'un trésor.

Question 10

Ajoutez désormais un attribut *energie* à l'explorateur qu'on initialisera à 10. Modifier la méthode *deplacement* pour que changer de sommet réduise de 1 l'énergie de l'explorateur.

Question 11

Ajoutez un attribut *nourriture* aux sommets qui prendra la valeur True ou False. Modifier la méthode *deplacement* tel que si nourriture vaut True, quand l'explorateur passe sur le sommet en question, il regagne 5 d'énergie et mets la valeur à False. Les sommets seront initialisés avec une valeur de nourriture aléatoire : True dans 10% des cas, False dans 90% des cas.

Question 12

Écrire une fonction *recherche_ nourriture* qui fonctionne comme *recherhce_ distance* mais qui à chaque étape mets à jour l'énergie de l'explorateur. On s'arrêtera quand l'explorateur n'aura plus d'énergie et on renverra le nombre de trésors trouvés.