Nom : **Prénom :**

TP – Back to school

Objectifs

L'objectif de ce TP est de réactiver les notions vues en Première :

- Les instructions de base ;
- Les structures de données (listes, tuples) ;
- Les fonctions ;
- L'utilisation de modules.

Pendant qu'Averell finit de manger, les trois autres Daltons prennent l'air dans un champ de 10 mètres sur 10 mètres. Rantanplan traîne aussi dans le champ. Les trois Dalton appellent simultanément le chien qui se dirige aléatoirement vers l'un d'eux. Trahi par sa mémoire défaillante, il s'arrête au milieu de son trajet et creuse un trou. Les trois Daltons le rappellent, il repart au hasard vers l'un d'eux, s'arrête à mi-chemin, creuse un trou, etc.

Le but de cette activité est de représenter graphiquement les trous faits par Rantanplan lorsqu'il est appelé un grand nombre de fois. Le résultat est…surprenant.

1 Modélisation des personnages

Mise en place

Créer un fichier Python dans votre éditeur et l'enregistrer avec un nom explicite, dans un emplacement auquel vous aurez toujours accès. (Nous aurons préalablement déterminé ensemble une bonne façon de faire.)

Voici un début de programme Python :

```
from matplotlib import pyplot # pour la représentation graphique

averell = (5, 5)
pyplot.plot(averell[0], averell[1], '*')
pyplot.show()
```

Taper ce programme dans votre fichier et l'exécuter pour vérifier que vous n'avez pas fait d'erreur et aussi que vous avez le module `matplotlib`. À défaut, l'installer.

Question 1

1. Analyser le programme ci-dessus et répondre aux questions suivantes :
   (a) Quel est le type de la variable `averell` ?

   (b) Que représente l'instruction `averell[0]` ?

   (c) Quel est le rôle de l'instruction `pyplot.plot(averell[0], averell[1],'*')` ?

2. En complétant le programme ci-dessus, placer les Dalton Joe, Jack et William qui seront respectivement représentés par les points `(0,0)`, `(10,0)` et `(5,10)` et en utilisant les noms de variables `joe`, `jack` et `william`.

Question 2

On stocke désormais les coordonnées initiales des trois Dalton (on laisse tomber Averell car trop feignant) à l'aide d'un tableau.

Compléter le programme pour faire apparaître sur le graphique les positions initiales des trois Dalton en utilisant une liste :

`dalton = [joe, jack, william]`.

2 Déplacements de Rantanplan

Rantanplan se situe au point de coordonnées `(5,5)`.

Question 3

En utilisant le déplacement de Rantanplan décrit en introduction, déterminer pour chacun des cas le point où il creusera un trou :

- Si Rantanplan est appelé par Joe ;
- Si c'est par Jack ;
- Si c'est par William.

Question 4

Bouge Rantaplan

Pour calculer le déplacement du chien d'un trou au suivant, nous allons créer la fonction `bouge_rantanplan(x, y, dalton)`.

Cette fonction prend en paramètres :

- x et y : les coordonnées courantes du chien, de type `float` ;
- dalton : la liste des Dalton, de type `list` de `tuple`.

Cette fonction renvoie `(x,y)` : les nouvelles coordonnées du chien après déplacement vers un des Dalton, de type `tuple`.

On rappelle que pour générer des nombres aléatoires, on utilise la fonction `randint` contenue dans le module `random` dont voici la documentation (qu'on peut afficher dans sa console Python à l'aide de la commande `help(randint)` après avoir importé le module) :

```
randint(a, b) method of random.Random instance
Return random integer in range [a, b], including both end points.
```

Compléter la fonction ci-dessous en respectant la spécification précédente :

```
from random import randint  # pour générer des nombres aléatoires

def bouge_rantanplan(x, y, dalton):
    indice_dalton_hasard =                  # choix aléatoire de l'un des Dalton
    x_dalton_hasard, y_dalton_hasard =              # coordonnées du Dalton
    x_rantanplan =                         # calcul des nouvelles coordonnées de Rantanplan
    y_rantanplan =
    return (x_rantanplan, y_rantanplan)
```

Tester en exécutant plusieurs fois l'instruction `bouge_rantanplan(5, 5, dalton)`.

Fait des trous

Voici une écriture possible de la fonction `fait_des_trous(nb_trous, coord_rantanplan, dalton)` dont la signature est :

`trous(int, tuple, list) -> list` et où :

- `nb_trous` est un entier représentant le nombre de trous à calculer ;
- `coord_rantanplan` est un tuple représentant les coordonnées initiales de Rantanplan ;
- `dalton` est une liste/tableau de tuples représentant les Dalton.

Cette fonction renvoie une liste/tableau de tuples contenant les coordonnées des `n` trous creusés.

```
1  def fait_des_trous_while(nb_trous, coord_rantanplan, dalton):
2      liste_trous = []
3      indice = 0
4      x_rantanplan, y_rantanplan = coord_rantanplan
5
6      while indice < nb_trous:
7          x_rantanplan, y_rantanplan = bouge_rantanplan(x_rantanplan, y_rantanplan, dalton)
8          liste_trous.append((x_rantanplan, y_rantanplan))
9
10     return liste_trous
```

Question 5

1. Expliquer le rôle des instructions lignes 7 et 8.

2. Juste en lisant ce programme, on constate que cette fonction comporte une erreur : laquelle ? La corriger et tester son fonctionnement.

3. Écrire cette même fonction en utilisant cette fois une boucle `for` (en nommant celle-ci différemment, par exemple `fait_des_trous_for`).

4. Tester ces fonctions pour 10 trous.

3 Affichage des trous

Pour finir, on souhaite afficher sur le graphique les trous creusés par Rantanplan.

Question 6

Afficher sur le graphique les trous creusés par Rantanplan.

Exécuter de nouveau le code en augmentant le nombre de trous creusés par Rantanplan. Comme Rantanplan est un peu fougueux, on pourra lui faire creuser jusqu'à un bon millier de trous.

(A) plop (B) toto (C) coucou