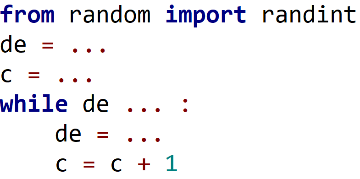
OBJECTIF: Découverte d’un modèle discret de décroissance exponentielle permettant d’introduire l’étude de la décroissance radioactive.

Travail de préparation : À faire à la maison

**PARTIE 1 : lancer un dé**

On lance un dé parfaitement équilibré à six faces, si on obtient 6 on s’arrête, sinon on continue à lancer le dé jusqu’à obtenir 6.

Afin de modéliser ce jeu, on décide de d’écrire un script, qui renvoie le nombre de lancers nécessaires à l’arrêt du jeu.

**1.** Compléter le script ci-contre afin qu’il modélise le jeu.

**2.** Quel est le rôle de la variable  ?

**3.** Que faut-il saisir dans la console pour obtenir le nombre de lancers avant d’obtenir 6 ?

**4.** Implanter ce script dans un éditeur Python.

Modéliser six lancers de dés successifs et noter les résultats obtenus dans le tableau ci-dessous

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| lancer n° | 1 | 2 | 3 | 4 | 5 | 6 |
| résultats |  |  |  |  |  |  |

Salle informatique : 1heure



**PARTIE 2 : lancer n dés**

Cette fois ci on décide de lancer simultanément dés à six faces, parfaitement équilibrés.

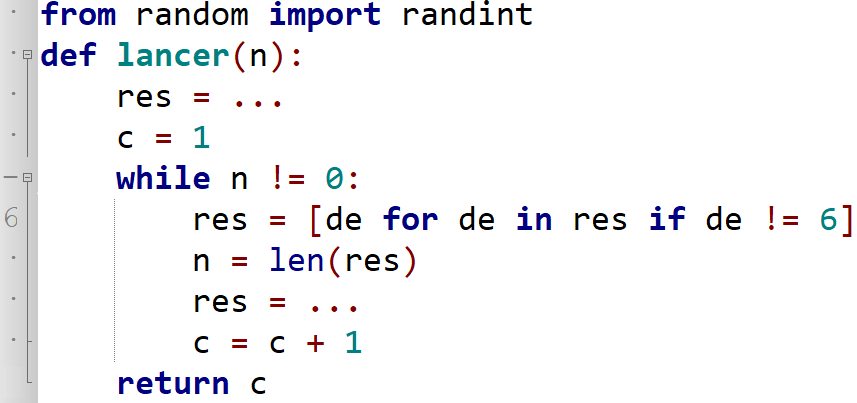
Tous les dés qui affichent 6 sont éliminés.

Puis on relance les dés restant jusqu’à ne plus avoir de dés.

Afin de modéliser ce nouveau jeu, on décide de créer une fonction Python : *lancer()* qui, connaissant le nombre de dés initialement utilisés, renvoie le nombre d’étapes nécessaires à l’élimination des dés.

**1.** Comment peut-on modéliser un lancer de dés en utilisant une liste en compréhension ?

**2.** Compléter le script de la fonction ci-contre afin qu’elle modélise le jeu.

**3.** Expliquer la ligne 6.

**4.** Implanter la fonction dans un éditeur Python et compléter le tableau ci-dessous :

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre n de dés | 1 000 | 10 000 | 100 000 |
| Nombre de lancers nécessaires à l’élimination des n dés |  |  |  |

**5.** Modifier la fonction précédente afin d’obtenir une nouvelle fonction *lancer\_demi()* qui renvoie le nombre d’essais nécessaires pour l’élimination de la moitié des dés.

**6.** Compléter le tableau ci-dessous :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n | 1 000 | 5 000 | 10 000 | 100 000 | 250 000 | 1 000 000 |
| *lancer\_demi(n)* |  |  |  |  |  |  |

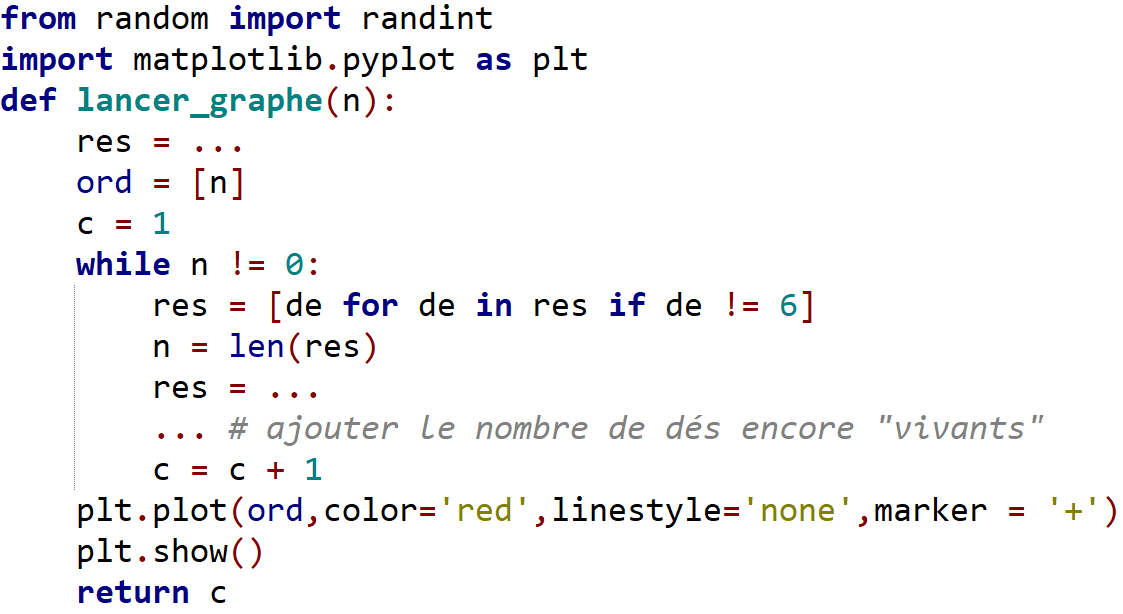
**a)** Que constate-on ?

**b)** Quel résultat peut-on conjecturer à la suite de cette expérience ?

**PARTIE 3 : Modélisation mathématique**

Afin d’identifier la relation qui lie le nombre de dés au nombre d’essais nécessaire à leur élimination, on décide de représenter le nombre d’essais en fonction du nombre de dés restants.

Pour cela on modifie une nouvelle fois la fonction *lancer()* de la partie précédente afin de créer une fonction *lancer\_graphe()* qui non seulement renvoie le nombre d’étapes nécessaires à l’élimination des dés mais aussi affiche le nombre de dés encore présents à chaque lancer.



*Tous les éléments graphiques relatifs à la bibliothèques matplotlib-pyplot sont donnés.*

**1.** Afin d’afficher le nombre de dés restant, on crée une liste *ord*, qui permettra de garder en mémoire le nombre de dés encore « en vie » à chaque étape.

Compléter le script de la fonction ci-contre :

**2.** Implanter cette fonction dans un éditeur Python, la tester avec différentes valeurs de .

**3.** Peut-on relier le nuage de points obtenus à une fonction de référence connue ?

**4.**Afin de caractériser le lien entre le nombre de dés encore vivants et le nombre d’essais, on modifie l’échelle de l’axe des ordonnées en prenant une échelle logarithmique à la place d’une échelle linéaire.

Pour cela on ajoutera avant l’instruction *plt.show()* :



Pour la prochaine fois

**RAPPEL : Papier semi-log**

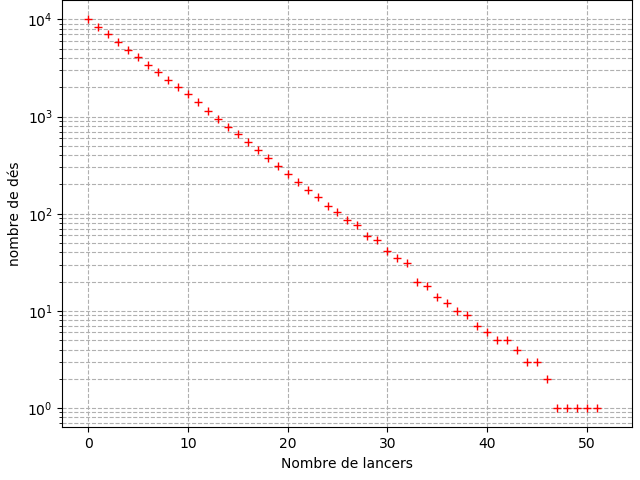
Un papier semi-logarithmique est utilisé lorsque l’une des séries de données possède une très grande amplitude.

Lorsque des points sont alignés sur un papier semi-logarithmique, dont l’échelle logarithmique est donnée en ordonnées, ces points sont liés par une formule du type : où a et b sont des réels.

Comme on en déduit que déterminer une telle relation revient à écrire :

On pourra donc directement modéliser la situation à l’aide de l’équation : où A et B sont des réels.

En faisant *lancer\_graphe(10000)* on a obtenu la représentation graphique ci-dessous :



Tracer une droite ajustant au mieux le nuage de points.

**5. a.** Choisir deux points sur cette droite afin d’établir une relation liant et .

**b.** En déduire une relation liant et .

DEVOIR MAISON Déposer les programmes Python dans l’ENT, les réponses aux questions pourront être données sous forme numériques, déposées dans l’ENT.

**PARTIE 4 : Désintégration radioactive**

Certains noyaux sont instables et se désintègre, c’est le phénomène de radioactivité.

La désintégration radioactive satisfait à deux conditions :

* Un noyau instable ne vieillit pas, sa probabilité de désintégration par unité de temps reste constante au cours du temps.
* La désintégration d’un noyau n’affecte pas celle des autres

**Être informé** : Aller dans « lycée connecté » et rechercher la définition de la radioactivité dans « l’encyclopédie Universalis » via le « Media Centre ».

Ainsi pour modéliser la désintégration radioactive on prendra noyaux. Chaque noyau a la probabilité de se désintégrer. Modéliser ce phénomène à l’aide d’une fonction *désintégration(,)*.

On assimilera un noyau à un dé, et la probabilité de se désintégrer à celle de faire 6.

**1.** Afin de simuler la probabilité qu’un noyau a à se désintégrer, on utilisera la fonction *uniform*(,) de la bibliothèque *random*.

Supposons que = 0,2, comment simuler cette probabilité ?

**2.** Adapter la fonction *lancer()* aux nouvelles conditions afin de créer la fonction *désintégration(,)*.

**3.** Modifier la fonction *désintégration(,)* afin de calculer la demi-vie du noyau dont la désintégration nous intéresse. On nommera *désintégration\_demi(,)* cette fonction. (On pourra reprendre le travail effectué dans la partie 2.)

**4.** Le **tritium** est un isotope radioactif de l’hydrogène. Il est notamment utilisé au LMJ (**L**aser **M**éga **J**oule, Le Barp) afin d’étudier la fusion nucléaire avec le deutérium(autre isotope de l’hydrogène).

En savoir plus : <http://www-lmj.cea.fr/fr/experiences/index.htm>

On sait qu’un noyau de tritium sur 650 disparait tous les 10 jours.

**a.** Déterminer la probabilité qu’un noyau de tritium disparaisse tous les 10 jours.

**b.** Utiliser le programme *désintégration\_demi(,)* pour déterminer la demi vie du tritium. On rappelle que l’on a conjecturé que la demi-vie ne dépendait pas du nombre de noyaux initialement présents. (cf. Partie 2 6°)b).

**c.** Adapter le programme précédent pour déterminer le temps au bout duquel le les trois quarts des noyaux de tritium auront disparu.

**d.** Au bout de combien d’années 90% des noyaux de tritium auront-ils disparu ?

**Pour aller plus loin : (Bonus)**

Afin de déterminer la relation qui lie le nombre de noyaux radioactifs avec le temps écoulé, on pourra créer une fonction *désintégration\_graphe(,)*, et à l’aide d’un ajustement affine, en repère semi-logarithmique, on pourra proposer une approximation de la relation liant les deux quantités.

**Boite à outils :**

* La bibliothèque *random* contient des fonctions qui permettent de simuler le hasard.
* La fonction *randint(,)*, où et sont des entiers relatifs permet de renvoyer un entier compris entre et , ce résultat pouvant être égal à ou à .
* La fonction *uniform(,)* renvoi un « float » de [ ;].
* La nombre d’éléments d’une liste L, ou longueur de L est *len(L)* (len pour lenght)