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

**Éléments mathématiques mobilisés :**

Python : boucle non bornée - boucle bornée - compteur (2nd) listes en compréhension (1ère TC Techno)

Statistiques : simulations - Nuage de points associé à une série statistique à deux variables quantitative - Ajustement affine (TC Terminale Techno)

Modèle discret de décroissance exponentielle (TC 1ère : uniquement modèle croissant)

Loi binomiale (TC Terminale Techno) 🡪 non utilisé dans cette version, peut être proposée en Partie 2- 1°) pour illustrer la proba d’obtenir 6 avec dés.

Papier semi-log (TC Terminale Techno)

Fonction log (TC Terminale Techno), exp (TC Terminale Techno) et ln (Maths/PC TSTI2D TSTL).

**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°)** Compétences : chercher - Calculer

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



**2°)** Compétence : Communiquer

Quel est le rôle de la variable c ?

c est un compteur, il permet de compter le nombre de lancers avant d’obtenir 6.

**3°)** Compétence : Chercher - Calculer

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

Il suffit de saisir >>> c.

**4°)** Compétences : Modéliser - Chercher

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 |  |  |  |  |  |  |



**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°)** Compétence : Représenter - Modéliser

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

*[ randint(1,6) for i in range()]*

****RAPPEL : possibilité de rappeler que *rang()* décrit la liste des entiers de 0 à - 1, donc cas possibles.

**2°)** Compétence : Représenter - Calculer

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



**3°)** Compétence : Communiquer - Modéliser

Expliquer la ligne 6.

la ligne 6 permet de ne garder que les dés qui n’ont pas fait de 6 au premier lancer, puis au lancer précédent lorsque l’on se trouve à l’intérieur de la boucle non bornée.

**4°)** Compétences : Calculer

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°)** Compétences : Raisonner - Modéliser

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°)** Compétence : Calculer

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 ? Compétence : Chercher - Modéliser - Communiquer

On constate que tous les résultats sont identiques.

On pourra préparer l’introduction du vocabulaire demi-« vie », en lien avec les dés encore « vivant ».

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

Le temps au bout duquel la moitié des dés est éliminé est constant, il ne semble pas dépendre du nombre de dés initialement choisi.

**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°)** Compétence : Modéliser - Chercher

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°)** Compétence : Calculer

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

**3°)** Compétence : Raisonner - Communiquer

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

S’attendre à la réponse : une hyperbole…dans ce cas laisser vivre l’erreur jusqu’à la validation via le papier semi-log.

**4°)** Compétence : Modéliser - Représenter

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()* :



**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)** Compétence : Raisonner - Communiquer

Choisir deux points sur cette droite afin d’établir une relation liant ln() et .



Les point semblent alignés, on décide d’ajuster le nuage de point à l’aide d’une droite, on obtient : ln() = a + b

Deux points : (0 ;104) et (40 ;6) : ln(y) - 0,185 + 4ln(10) (TRAVAIL SUR LES CHIFFRES SIGNIFICATIFS ?)

Possibilité de faire une vérification graphique mais attention la représentation ne sera pas tout à fait la même.

 **b)** Compétence : Calculer En déduire une relation liant et .

**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°)** Compétence : Modéliser - Raisonner- Communiquer

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 ?

Il suffit de choisir au hasard un nombre de [0 ;1], si ce nombre est inférieur à 0,2 l’expérience sera considérée comme réussi, s’il est strictement supérieur à 0,2 l’expérience sera considérée comme non réussi.

 p = uniform(0,1)

 if p <= 0.2 : noyau désintégré

 else : noyau vivant

**2°)** Compétence : Chercher - Représenter

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



**3°)** Compétences : Représenter - Chercher

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°)** Compétences : Chercher - Modéliser Représenter - Communiquer- Calculer

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)** Compétence : Chercher - Calculer

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

 **b)** Compétence : Calculer - Représenter - Raisonner

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).



La demi-vie du tritium est de 451 \* 10 /365 12,36 années.(12,32 d’après Universalis).

 **c)** Compétence : Modéliser

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

Remplacer N/2 par N/3 :



904, résultats compris entre 892 et 916. Donc 904 \* 10/365 24,77 ans.

 **d)** Compétences : Modéliser - Représenter - Communiquer

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

1504, résultats compris entre 1485 et 1510. Donc 1504 \* 10/365 41,21 ans.

**Pour aller plus loin :**

Compétences : Chercher - Modéliser - Représenter - Calculer - Raisonner - Communiquer

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)