Simulation de la dérive génétique

L'objectif de cette activité est de simuler la dérive génétique à partir d'une population initiale d'individus. Chaque individu est assimilé à son génotype. On s'intéressera à un seul gène représenté sous la forme de deux allèles 'm' et 'v'.
On suppose qu'il n'y a pas de mutation au cours des générations, ni de pression de l'environnement (sélection naturelle).
Par ailleurs, on considère que la fréquence des allèles dans la population ne dépend pas du sexe des individus. Les couples seront donc formés par le choix aléatoire de deux individus de cette population.

TP à réaliser sur notebook

A. Mise en place de quelques fonctions pour faciliter l'écriture du programme en python

1. La fonction Pop

La fonction Pop définie ci-dessous génère aléatoirement une population initiale formée de 2×n individus afin de pouvoir former n couples d'individus pour la reproduction.
Chaque individu est caractérisé par un couple d'allèles pour le gène étudié.

```
from random import *

def Pop(n):
    U = ['m','v']
    P = []
    for i in range(1,2*n+1):
        I = [choice(U),choice(U)]
        P.append(I)
    return P
```

Commentaires :

*On importe la bibliothèque **random**.*

n est un entier naturel représentant le nombre de couples.
*U est une **liste** dans laquelle on choisira au hasard les allèles pour constituer les individus.*
La population est au départ une liste vide.
*La boucle **"pour"** répète 2n fois les deux instructions indentées.*
***"choice"** renvoie un allèle choisi au hasard dans U.*
I est un individu, c'est-à-dire une liste de deux allèles.
***"append"** ajoute l'individu I à la fin de la liste P.*

a. Exécuter cette fonction. Expliquer comment est défini l'individu I et ce que contient la variable P en fin d'exécution.
b. Saisir : **Ma_population** = Pop(3), puis afficher **Ma_population** (saisir à nouveau **Ma_population** puis exécuter la cellule).
c. Combien y a-t-il d'individus dans **Ma_population** ? Est-ce bien la valeur attendue ?

2. La fonction Dist

La fonction Dist définie ci-dessous renvoie la distribution des fréquences des allèles 'm' et 'v' dans une population P.

```
def Dist(P):
    N_m = 0
    N_v = 0
    for I in P:
        N_m = N_m + I.count('m')
        N_v = N_v + I.count('v')
    return [N_m/(2*len(P)),N_v/(2*len(P))]
```

Commentaires :

P est une liste d'individus.
N_m représente le nombre d'allèles 'm'
N_v représente le nombre d'allèles 'v'
*Ici la boucle **"pour"** fait **parcourir** I **dans la liste** P.*
***"count"** compte le nombre d'allèles 'm'et d'allèles 'v' dans la liste I.*
***"len"** renvoie le nombre d'éléments de la liste P.*

a. Exécuter cette fonction. Expliquer comment est calculée la fréquence des allèles 'm' et 'v'.
b. Saisir : Dist(**Ma_population**), puis exécuter la cellule.
c. En observant **Ma_population**, vérifier que les fréquences renvoyées ci-dessus sont bien celles attendues. Effectuer les calculs nécessaires.

3. La fonction Pop_suivante

La fonction Pop_suivante définie ci-dessous renvoie, à partir d'une population P d'effectif pair, la génération suivante de même effectif.

```
def Pop_suivante(P):
    P_suivant = []
    while P != []:
        I_1 = choice(P)
        P.remove(I_1)
        I_2 = choice(P)
        P.remove(I_2)
        E_1 = [choice(I_1),choice(I_2)]
        E_2 = [choice(I_1),choice(I_2)]
        P_suivant.append(E_1)
        P_suivant.append(E_2)
    return P_suivant
```

Commentaires :

"tant que" *P n'est pas vide*

"remove" *retire l'individu I_1 de la liste P.*

E_1 et E_2 sont les descendants du couple (I_1 , I_2)

a. Exécuter cette fonction. Interpréter comment sont définis les descendants E_1 et E_2.
b. Saisir : P_s = Pop_suivante(**Ma_population**), puis afficher P_s. Commenter.
c. Saisir : len(P_s). La valeur renvoyée est-elle bien la valeur attendue ?
d. Comment calculer la fréquence des allèles 'm' et 'v' dans P_s ? Donner ces fréquences.

B. Le programme

Le programme ci-dessous est à compléter et doit représenter graphiquement l'évolution des fréquences alléliques d'une population à effectif constant au cours des générations.
On utilisera les fonctions définies précédemment.

```
import matplotlib.pyplot as plt

n = int(input("Nombre de couples dans la population : "))
g = int(input("Nombre de générations : "))

plt.axis([0,g,0,1])
plt.xlabel('Générations')
plt.ylabel('Fréquences')

P = Pop(n)
L = Dist(P)
plt.plot(0,L[0],'m.')
plt.plot(0,L[1],'gx')

for i in range(1,g+1):
    P = _ _ _ _ _ _ _ _ _ _ _
    L = Dist(P)
    plt.plot(_ _,L[0],'m.')
    plt.plot(_ _,L[1],'gx')

plt.show()
```

Commentaires :

On importe la bibliothèque graphique ***matplotlib.pyplot*** *avec le raccourci* ***plt****.*
"input" *invite l'utilisateur à saisir une valeur n avec le texte " . . . . . " et* ***"int"*** *convertit la saisie en nombre entier.*
"plt.axis" *règle la fenêtre graphique* [*Xmin,Xmax,Ymin,Ymax*].
"plt.xlabel" *et* ***"plt.ylabel"*** *donnent un titre aux axes.*

"plt.plot" *trace ici les points de coordonnées* (0 , *L*[0]) *et* (0 , *L*[1]) *avec respectivement un point mauve et une croix verte. L*[0] *est le* 1^{er} *élément de la liste L, L*[1] *en est le* $2^{ème}$.

A Compléter

"plt.show" *affiche le repère.*

a. Une fois complété, exécuter le programme ci-dessus.
b. On fixe g = 100. Exécuter ce programme avec n = 5, n = 50, n = 500, n = 5 000 et n = 50 000.
On collera les graphiques obtenus dans un fichier texte.
c. Commenter ces graphiques et expliquer ce que l'on appelle "dérive génétique".