Chaînes de Markov (1)

Modélisation de phénomènes aléatoires - simulations de chaînes de Markov

Modélisation de phénomènes aléatoires - simulations de chaînes de Markov

1 Une puce se déplace sur la ligne suivante en sautant au hasard d'une case ou de deux cases vers la droite. Elle part de la case 1 (et elle va toujours vers la droite).
On observe sa position après 6 sauts.

1	2	3	4	5	6	7	8	9	10	11	12	13

1°) Quelles sont les cases qui peuvent être atteintes après 6 sauts ?

2°) Recopier et compléter le programme Python suivant pour faire afficher la position de la puce après 6 sauts :

```
from random import randint

def position():
    position=.....
    for i in range(..............):
        position=position + randint(1,2)
    return position
```

Exécuter le programme 10 fois et compléter le tableau :

Issue							
Fréquence							

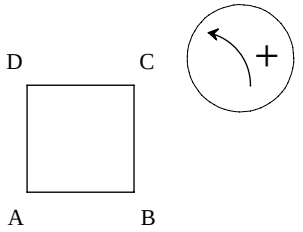
3°) Recopier et compléter le programme suivant pour qu'il répète l'expérience 100 fois et qu'il comptabilise les arrivées à la case 8.

```
from random import randint

def position():
    position=.....
    for _ in range(..............):
        position=position + randint(1,2)
    return position

def arrivee8():
    arrivee=0
    for i in range(100):
        if ..................................... :
            arrivee=arrivee + 1
    return arrivee
```

4°) Modifier la deuxième partie du programme pour pouvoir comptabiliser les arrivées dans n'importe quelle case saisie en paramètre (*n*) :

```
def arrivee(n):
    arrivee=0
    for i in range(100):
```

2 Soit ABCD un carré direct dans le plan orienté.
Une fourmi parcourt les côtés de ce carré en partant du sommet A et met une minute pour parcourir un côté.
Arrivée à un sommet, elle choisit au hasard l'un ou l'autre des deux côtés issus de ce sommet pour poursuivre sa marche.



Quels sont les sommets qui peuvent être atteints après 1 minute ? 2 minutes ? 3 minutes ? 4 minutes ? Généraliser.

Pour simuler les déplacements de la fourmi, on affecte les chiffres 0, 1, 2, 3 au fait d'arriver respectivement aux sommets A, B, C et D.
Si la fourmi se déplace dans le sens direct, on ajoute 1 lors du déplacement. Sinon, on retranche 1 et on additionne au fur et à mesure des déplacements.

Recopier et compléter la fonction Python `position(n)` qui prend pour argument un entier naturel *n* supérieur ou égal à 1 et qui affiche la position de la fourmi après *n* minutes :

```
from random import choice

def position(n):
    s=.....
    for i in range(..............):
        x=choice([- 1,1])
        s=s+ .....
        s=s%4
    return s
```

Exécuter le programme.

Il est possible de généraliser à un déplacement quelconque sur un polygone régulier.

3 Un pion se déplace sur le damier suivant constitué de 9 cases. Au départ, il est sur la case 0.

– 4	– 3	– 2	– 1	0	1	2	3	4

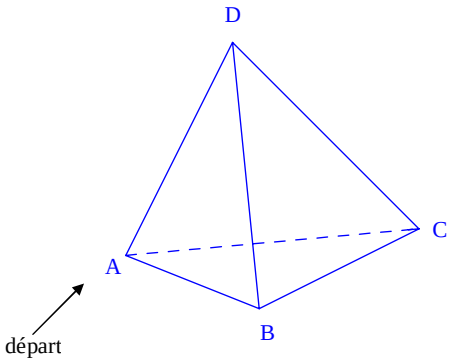
Les déplacements du pion sont déterminés par le lancer d'une pièce de monnaie bien équilibrée.
Si la pièce tombe du côté face, le pion se déplace d'une case vers la droite ; sinon il se déplace d'une case vers la gauche. Si le pion est sur la case 4 ou – 4, il y reste.

1°) On envisage un trajet qui est une succession de 4 déplacements.
Écrire une fonction Python qui simule un trajet, c'est-à-dire qui renvoie la position finale du pion.

2°) On envisage un trajet qui est une succession de *n* déplacements où *n* est un entier naturel supérieur ou égal à 1.
Même question ue précédemment.

4 On promène un pion sur les sommets d'un tétraèdre régulier ABCD. Toutes les minutes, on déplace le pion d'un sommet à un autre, en choisissant au hasard parmi les trois sommets possibles.
On suppose qu'au départ le pion se trouve en A.
On envisage *n* déplacements où *n* est un entier naturel supérieur ou égal à 1.

Écrire une fonction Python qui prend pour argument l'entier naturel *n* et qui renvoie la position où se trouve le pion au bout de *n* déplacements.
Figure :



5 Les urnes d'Ehrenfest à 2 boules

On considère deux urnes A et B. On dispose également d'une pièce de monnaie équilibrée.
Au départ,
- l'urne A contient deux boules indiscernables au toucher, numérotées 1 et 2 ;
- l'urne B est vide.

On effectue des lancers successifs indépendants de la pièce en notant à chaque fois le côté qu'elle présente.
Si elle présente le côté pile (P), on change d'urne la boule numéro 1.
Si elle présente le côté face (F), on change d'urne la boule numéro 2.
On s'intéresse à l'évolution du nombre de boules dans A.
On s'intéresse à l'évolution du nombre de boules de chaque urne.

1°) Dans cette question, on choisit de s'intéresser à une série de 7 lancers de la pièce.
On suppose que les 7 lancers ont donné les résultats suivants : P-P-F-P-F-F-P.
Originellement, les deux boules sont dans l'urne A.
Le premier lancer donne pile donc on change d'urne la boule 1. La boule 1 est donc placée dans l'urne B.

Recopier et compléter le tableau suivant :

		Urne A	Urne B	Nombre de boules dans A
Étape 0	Au départ	1-2		
Étape 1	Après le 1er lancer	2	1	
Étape 2	Après le 2e lancer	1-2		
Étape 3	Après le 3e lancer	1	2	
Étape 4	Après le 4e lancer		1-2	
Étape 5	Après le 5e lancer			
Étape 6	Après le 6e lancer			
Étape 7	Après le 7e lancer			

Représenter sur un graphique le nombre de boules dans l'urne A en fonction du numéro de l'étape (on peut parler de « trajectoire »).
2°) Démontrer qu'au bout de *n* étapes, les deux boules sont dans la même urne si *n* est pair et ne sont pas dans la même urne si *n* est impair.
3°) On considère le programme Python suivant : s : nombre de boules de l'urne A

```
from random import choice

def urne(n):
    s=2
    for i in range(n):
        if s==2:
            s= 1
        elif s==0:
            s=1
        else:
            x=choice([-1,1])
            s=s+x
        print (s)
```

Lire et comprendre les programmes suivants.
Recopier l'un des programmes au choix puis le taper et le faire tourner.

1ère version :

```
from random import randint

def ehrenfest(n):
    A=[1,2]
    for _ in range(n):
        x=randint(1,2)
        if x in A:
           A.remove(x)
        else:
           A.append(x)
    return A # contenu de l'urne A au bout de n étapes
```

2e version :

```
from random import randint
import matplotlib.pyplot as plt

def ehrenfest(n):
    A=[1,2]
    X=[2]   #liste répertoriant le nombre de boules dans l'urne A
    for _ in range(n):
        b=randint(1,2)
        if b in A:
           A.remove(b)
        else:
           A.append(b)
        X.append(len(A))
    return X

def graphique(n):
    plt.xlim(0,n+1)
    plt.ylim(0,3)
    plt.grid(linestyle="-")
    x=[i for i in range(n+1)]
    plt.plot(x,ehrenfest(n),marker='o')
    plt.show()
```

Le 27-3-2023

Proposition Olivier Allard

Programme Python avec un graphique :

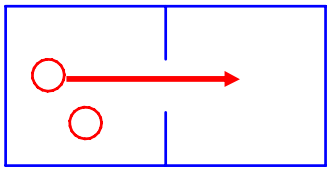
```
from random import choice
import matplotlib.pyplot as plt

def ehr(n) :
    x=2
    M=[x]
    for _ in range(n) :
        if x==2 :
            x=1
        elif x==1 :
            x=choice([0,2])
        else :
            x=1
        M.append(x)
    return M

def traj(n) :
    x=list(range(n+1))
    plt.plot(x,ehr(n))
    plt.show()
```

Utiliser https://www.codabrainy.com/python-compiler/

Ce modèle d'urne a été inventé par deux physiciens, les époux Paul et Tatiana Ehrenfest, en 1907.
Tous deux étaient amis d'Albert Einstein.

On considère l'expérience suivante : une enceinte hermétique est séparée en deux compartiments A et B de taille égale et mis en communication par un trou.
On remplit A d'un gaz. Assez rapidement, les molécules migrent d'un compartiment à l'autre et cela, jusqu'à atteindre une position d'équilibre où il y aura autant de molécules de gaz dans chaque compartiment.
Or, toutes les lois de la mécanique quantique sont réversibles par rapport au temps. Ainsi, le système devrait revenir à son état initial ; or, cela ne se produit pas.
On se propose d'étudier ce paradoxe à l'aide de la modélisation imaginée par les époux Paul et Tatiana Ehrenfest en 1907.
Nous allons nous placer dans le cas où l'on aurait deux molécules de gaz au départ.
Le modèle d'Ehrenfest consiste à considérer deux urnes A et B.

Schéma du dispositif (à refaire) :



Paul Ehrenfest (à gauche) au piano avec le célèbre physicien Einstein.

Paul Ehrenfest (1880-1933) de nationalité hollandaise autrichienne et allemande est un physicien qui fut formé à l'Ecole Polytechnique de Vienne et à l'université de Göttingen. Il étudie à la Technische Hochschule de Vienne à partir d'octobre 1899, où il suit les cours de Boltzmann sur la « théorie mécanique de la chaleur », notre actuelle théorie cinétique des gaz. En 1901, il étudie à Göttingen, où il suit les cours de Klein et Hilbert en mathématiques, d'Abraham en électromagnétisme, ainsi que ceux de Stark, Nernst, Schwarzschild et Zermelo. Il y rencontre également sa future femme, la mathématicienne russe Tatiana Afanassieva, avec qui il écrira en 1911 un article renommé sur les fondements de la mécanique statistique. Il soutient sa thèse, Le *mouvement des corps rigides dans les fluides et la mécanique de Hertz*, sous la supervision de Boltzmann, le 23 juin 1904 à Vienne. Il succède à Lorentz à la chaire de physique théorique de l'université de Leyde. Il eut avec sa femme quatre enfants, dont une fille, Tatiana Pavlovna Ehrenfest, née en 1905 et devenue mathématicienne également.

6 Les urnes d'Ehrenfest à N boules

On considère deux urnes A et B.
Au départ,
- l'urne A contient N boules indiscernables au toucher (N étant un entier naturel supérieur ou égal à 1) numérotées 1, 2, …, N ;
- l'urne B est vide.

À chaque étape, on choisit un entier au hasard entre 1 et N.
La boule portant ce numéro est changée d'urne.
Lire et comprendre les programmes suivants, puis le taper et le faire tourner.

```
from random import randint
import matplotlib.pyplot as plt

def ehrenfest(N,n):
    A=[i for i in range(1,N+1)] ou list(range(1,N+1))
    X=[N]
    for i in range(n):
        b=randint(1,N)
        if b in A:
           A.remove(b)
        else:
           A.append(b)
        X.append(len(A))
    return X

def trajectoire(N,n):
    plt.xlim(0,n)
    plt.ylim(0,N+1)
    plt.grid(linestyle= "–" )
    x =[i for i in range (n+1)]
    plt.plot(x,ehrenfest(N,n),marker='o')
    plt.show()
```

Le 30 avril 2024

```
from random import randint
import matplotlib.pyplot as plt

def ehrenfest(N,n):
    A=[i for i in range(1,N+1)] ou list(range(1,N+1))
    X=[N]
    for i in range(n):
        b=randint(1,N)
        if b in A:
           A.remove(b)
        else:
           A.append(b)
        X.append(len(A))
    return X
```

Le 30 avril 2024

Le programme suivant marche bien :

```
from random import randint
import matplotlib.pyplot as plt

def ehr(N,n):
   A=[i for i in range(1,N+1)]
   X=[N]
   for i in range(n):
      b=randint(1,N)
      if b in A:
        A.remove(b)
      else:
        A.append(b)
      X.append(len(A))
   x=list(range(n+1))
   plt.plot(x,X)
   plt.show()
```

Programme tiré d'un document équipe DREAM Claroline

```
from random import randint

def init(N):
    urnA=list(range(1,N+1))
    urnB=[]

def exp(N):
    init(N)
    r=randint(1,N)
    if r in urnA:
       urnB.append(r)
       urnA.remove(r)
    else:
       urnA.append(r)
       urnB.remove(r)
    return urnA,urnB

def simu(n,N):
    init(N)
    for i in range(N):
        exp(n)
    return len(urnA), len(urnB)
```

Le 30 avril 2024

```
from random import
from random import randint

def exp(N):
    urnA=list(range(1,N+1))
    urnB=[]
    r=randint(1,N)
    if r in urnA:
       urnB.append(r)
       urnA.remove(r)
    else:
       urnA.append(r)
       urnB.remove(r)
    return urnA,urnB
```


IFé 0 1 2 3 1 1 3 1 3 2 3 1 3 1 for i in range(100): print expe_ehrensein(10000,500)

7 Une urne A contient 2 boules blanches et une urne B contient 4 boules noires.
On procède au tirage aléatoire simultané d'une boule de chaque urne et chaque boule extraite est changée d'urne.
On répète cette opération.
On s'intéresse au contenu de l'urne A à chaque étape.
Il y a trois états possibles :
« L'urne A contient 2 boules blanches » (état 1),
« L'urne A contient 1 boule blanche et 1 boule noire » (état 2),
« L'urne A contient 2 boules noires » (état 3).
On notera que le nombre de boules est constant dans chaque urne.
Le contenu de l'urne B se déduisant à chaque fois de celui l'urne A, il n'est pas utile de le préciser pour définir les états.

Écrire une fonction Python qui prend pour argument un entier naturel *n* supérieur ou égal à 1 et qui renvoie une liste donnant le nombre de boules blanches dans l'urne A au cours de *n* tirages successifs.

On utilisera des listes et les fonctions append et remove.

8 Chaque matin, l'allumeur de réverbère du Petit Prince change l'état de sa planète avec une probabilité de 0,75.
On considère le processus aléatoire comportant les deux états suivants :
état A : « Le réverbère est allumé » ;
état E : « Le réverbère est éteint ».
1°) Représenter la situation à l'aide d'un graphe probabiliste.
2°) Recopier et compléter les programmes Python suivants.

```
def suivant(x):
    if x == "A":
       r=random()
       if r<0.75:
          return ......
       else:
          return ......
    else:
       r=random()
       if r<0.75:
          return ......
       else:
          return ......
```

```
def simule_petitprince(n):
    x = "A"
    for i in range(n):
        x = suivant(x)
    print(x)
```

9 Soit N un entier naturel non nul.
Un pion se déplace sur les cases disposées sur le schéma ci-dessous. Au départ, il est sur la case 0.

$-N$	$-(N-1)$	…	-1	0	1	…	$N-1$	N

Les déplacements du pion sont déterminés par le lancer d'une pièce de monnaie bien équilibrée.
Si la pièce tombe du côté face, le pion se déplace d'une case vers la droite. Sinon, il se déplace d'une case vers la gauche. Si le pion est sur la case N ou $-N$, il y reste.
On peut assimiler la situation à une marche aléatoire sur un graphe dont les sommets représentent la position du pion : $-N$, $-(N-1)$, …, -1, 0, 1, …, $N-1$, N.

1°) Écrire la matrice de transition en lignes A pour $N=1$ et $N=2$.
2°) Dans cette question, on prend $N=4$.
Déterminer la probabilité que le pion revienne à la case de départ après 4 lancers.
3°) **Simulation**
On considère la fonction Python ci-dessous qui prend pour pour argument deux entiers naturels *n* et N supérieurs ou égaux à 1 et qui renvoie la position du pion au bout de *n* étapes.

Le 30-3-2023 Il faut écrire une question ….

```
from random import choice

def simu(n,N):
    x=0
    for i in range(n):
        if abs(x)<N:
           r=choice([-1,1])
           x=x+r
     return x
```

10 Un compte bancaire peut être créditeur (il est alors « dans le vert ») ou débiteur (il est alors « dans le rouge »).
L'évolution mensuelle du compte bancaire d'un certain client est telle que :
- lorsque le compte est « dans le vert » à la fin du mois, la probabilité qu'il le soit encore à la fin du mois suivant est 0,4 ;
- lorsque le compte est « dans le rouge » à la fin du mois, la probabilité qu'il le soit encore à la fin du mois suivant est 0,35.

1°) On définit les états V : « Le compte est dans le vert » (état 1) et R : « Le compte est dans le rouge » (état 2).
Représenter le graphe probabiliste traduisant la situation.
2°) Rédiger un programme Python de simulation.

Solutions des exercices

1 Une puce se déplace sur la ligne suivante…

Corrigé :

Après 6 sauts, la fourmi peut atteindre les cases 7, 8, 9, 10, 11, 12, 13.

1°)

```
from random import randint

def position():
    position=1
    for _ in range(1,7):
        position=position+randint(1,2)
    return position
```

À la place de `range(1,7)`, on peut aussi écrire `range(6)`.

On notera que la fonction position est une fonction sans argument.

```
from random import randint

def position():
    position=1
    for _ in range(1,7):
        position=position+randint(1,2)
    return position

def arrivee8():
    arrivee=0
    for i in range(100):
        if position()==8
           arrivee=arrivee + 1
    return arrivee
```

2°) Exécuter le programme 10 fois permet de siuler 10 séries de 6 sauts dans des conditions identiques indépendantes.

Les fréquences sont des nombres compris entre 0 et 1. La somme des fréquences est égale à 1.
Chacun a un tableau de fréquences différent.

Résultats de Maxime Guégan obtenus le mardi 23-4-2024

Issue	7	8	9	10	11	12	13
Fréquence	0,1	0,1	0,2	0,1	0,1	0,4	0

Comme on exécute le programme 10 fois, on divise les résultats par 10.
On pourrait exprimer les fréquences en pourcentage.

3°) `position()==n`

2 Soit ABCD un carré direct dans le plan orienté.
Une fourmi parcourt les côtés de ce carré en partant du sommet A…

Corrigé :

1 minute : sommets A et C
2 minutes : sommets B et D
3 minutes : sommets A et C
4 minutes : sommets B et D

Généralisation :

Au bout de *n* minutes (*n* étant un entier naturel supérieur ou égal à 1) :
La puce peut être en A ou C si *n* est pair.
La puce peut etre en B ou en D si *n* est impair.

```
from random import choice

def position(n):
    s=0
    for _ in range(1,n+1):
        x=choice([- 1,1])
        s=s+x
        s=s%4
    return s
```

On peut utiliser la fonction choice qui permet de choisir un élément au hasard dans une liste.

Variante possible : L'instruction « `2*randint(0, 1) - 1` » fournit un entier aléatoire égal à 1 ou à – 1.

```
from random import randint

def position(n):
    s=.....
    for i in range(..............):
        x = 2*randint(0, 1) - 1
        s=s + .....
        s=s%4
    return s
```

Exécuter le programme.

Il est possible de généraliser à un déplacement quelconque sur un polygone régulier.

3 Un pion se déplace sur le damier suivant constitué de 9 cases. Au départ, il est sur la case 0.

– 4	– 3	– 2	– 1	0	1	2	3	4

1°) On envisage un trajet qui est une succession de 4 déplacements.

```
from random import choice

s=0
for i in range(1,5):
    a=choice([-1,1])
    s=s+a
print (s)
```

2°) On envisage un trajet qui est une succession de *n* déplacements où *n* est un entier naturel supérieur ou égal à 1.
Écrire une fonction Python qui permet de simuler un trajet.

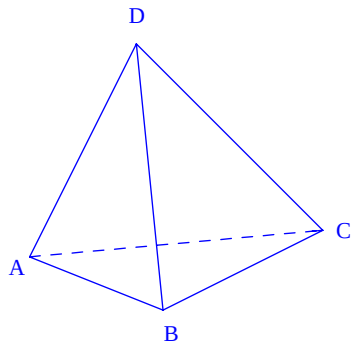
Le 26-4-2024

Programme d'Albane Montanier

```
def trajet(n) :
    position=0
    for _ in range(n):
        p=choice([-1,1])
        position=position +p
        if position==4 or position==-4
           return position
        return position
```

4 On promène un pion sur les sommets d'un tétraèdre régulier ABCD. Toutes les minutes, on déplace le pion…

Solution :



Marche aléatoire sur un graphe connexe sur un graphe complet

On utilise 2 listes.

La variable *x* sert à noter la position du pion.

Le 26-4-2024

Programme d'Albane Montanier

```
def trajet(n) :
    def trajet (n) :
    position='A '
    for _ in range(n) :
       L=['A','B','C','D']
       L.remove(position)
       position=choice(L)
     return position
```

Si on tape `trajet(1)`, on peut obtenir B, C, D.

Programme avec toutes les positions :

```
from random import choice

def simul(n):
    x=1
    M=[x]
    for _ in range (1,n+1):
      L=[1,2,3,4]
      L.remove(x)
      x=choice(L)
      M.append(x)
    return M
```

On peut aussi utiliser une liste `L=['A','B','C','D']`.

Partie à enlever :

Indication : On pourra numéroter respectivement 0, 1, 2, 3 les sommets A, B, C, D.

Mon programme :
On ajoute un nombre aléatoire entre 0 et 3.
On réduit ensuite (division euclidienne par 4).

Le 20-4-2022

Programme Python fait par Vicente Seixas

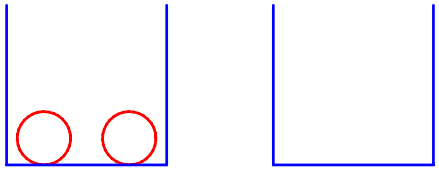
```
from random import randint

L=['A','B','C','D']
p=0
for i in range(n):
    indice=p
    while indice==p:
          p=randint(0,len(L)-1)
    print (L[p])
```

Commentaires :

`L=['A', 'B', 'C', 'D']`	On initialise une liste.
`p=0` :	La fourmi commence en A.
`for i in range (n):`	On répète 10 fois.
`indice=p`	utile pour la suite
`while indice==p:`	On force la fourmi à se déplacer.

5 Les urnes d'Ehrenfest à 2 boules

Schéma des urnes (à refaire) :



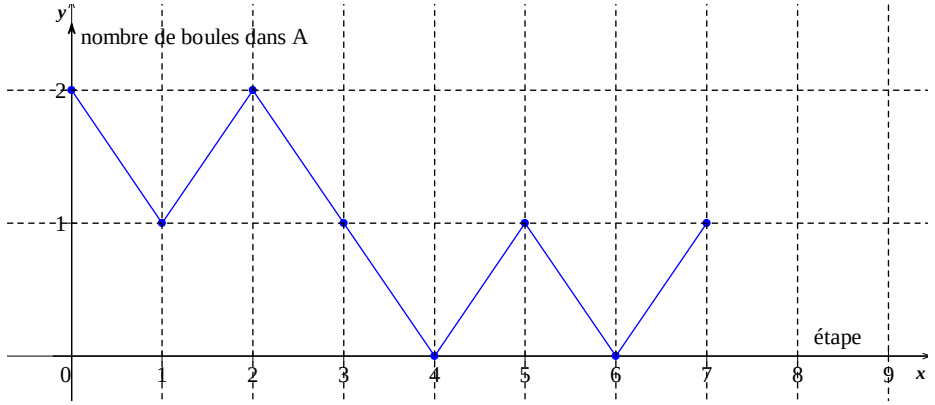
État E_0 : l'urne A contient 2 boules et l'urne B contient 0 boule.
État E_1 : l'urne A contient 1 boule et l'urne B contient 1 boule.
État E_2 : l'urne A contient 0 boule et l'urne B contient 2 boules.

E_0	E_1	E_2
Les 2 dans A	1 dans A 1 dans B	Les 2 dans B

		Urne A	Urne B	Nombre de boules dans A
Étape 0	Au départ	1-2		2
Étape 1	Après le 1er lancer	2	1	1
Étape 2	Après le 2e lancer	1-2		2
Étape 3	Après le 3e lancer	1	2	1
Étape 4	Après le 4e lancer		1-2	0
Étape 5	Après le 5e lancer	2	1	1
Étape 6	Après le 6e lancer		1-2	0
Étape 7	Après le 7e lancer	1	2	1

On observe un retour à l'état initial après le 2e lancer.

La trajectoire conrrespondant à la promenade aléatoire est la suivante.



8 Chaque matin, l'allumeur de réverbère du Petit Prince change l'état de sa planète avec une probabilité de 0,75.

```
def suivant(x):
    if x == "A":
       r=random()
       if r<0.75:
          return "E"
       else:
          return "A"
    else:
       r=random()
       if r<0.75:
          return "A"
       else:
          return "E"
```

```
def simule_petitprince(n):
    x = "A"
    for i in range(n):
        x = suivant(x)
    print(x)
```