Corrigé Sujet Mines_Ponts Automate cellulaire

essai_pixel

```
import pixel as px

px.initialiser(10,10,20)

px.marquer(0,0,0)
px.marquer(0, 9, 0)
px.marquer(9,9,0)
px.marquer(2, 10, 0)
px.afficher()
```

automate

```
## Sujet mines ponts zéro
############################

## importation des modules
############################

from random import random
import matplotlib.pyplot as plt

## définition des fonctions
############################

def calcul_n(h):
    '''renvoie le nombre de grains qui vont tomber sur la pile suivante,
pour h entier naturel.'''
    if h > 1:
        return( int( (h + 2.0) / 2 * random() ) + 1 )
    else :
        return(0)

def initialisation(P):
    '''renvoie une variable piles de type liste
contenant P piles de hauteur 0.'''
    return( [0] * P )

def actualise (piles, perdus):
    '''revoie la variable piles acutalisée selon les règles d'évolution
ainsi que le nombre de grain perdus.'''
    for i in range(len(piles) - 1):
        n = calcul_n( piles[i] - piles[i + 1])
        piles[i] -= n
        piles[i + 1] += n
    np = calcul_n(piles[-1])
    perdus += np
    piles[-1] -= np
```

```
    return(piles, perdus)

def automate(P):
    '''gère l'automate et renvoie les piles à l'état final.'''
    piles = initialisation(P)
    perdus = 0
    while True :
        for i in range(10):
            piles, perdus = actualise(piles, perdus)
            if perdus == 1000 :
                # permet de s'arrêter dès que 1000 grains sont sortis
                return(piles)
        piles[0] += 1

## programme principal
#########################

piles_finales = automate( int(input("Taille du support ? ")) )

x = [i for i in range(len(piles_finales))]

plt.plot(x, piles_finales)

plt.show()
```

automate_evolution

```
## Sujet mines ponts zéro
############################

## importation des modules
############################

from random import random
import matplotlib.pyplot as plt
import pixel as px

## définition des fonctions
############################

def calcul_n(h):
    '''renvoie le nombre de grains qui vont tomber sur la pile suivante,
pour h entier naturel.'''
    if h > 1:
        return( int( (h + 2.0) / 2 * random() ) + 1 )
    else :
        return(0)

def initialisation(P):
    '''renvoie une variable piles de type liste
contenant P piles de hauteur 0.'''
    return( [0] * P )
```

```
def actualise (piles, perdus):
    '''revoie la variable piles acutalisée selon les règles d'évolution
ainsi que le nombre de grain perdus.'''
    for i in range(len(piles) - 1):
        n = calcul_n( piles[i] - piles[i + 1])
        piles[i] -= n
        for k in range(n):
            px.marquer(i, 2 * len(piles) - (piles[i] + k + 1), 1)
        piles[i + 1] += n
        for k in range(n):
            px.marquer(i + 1, 2 * len(piles) - (piles[i + 1] - k), 0)
    np = calcul_n(piles[-1])
    perdus += np
    piles[-1] -= np
    for k in range(np):
        px.marquer((len(piles) - 1),  \
                2 * len(piles) - (piles[i + 1] + k + 1), 1)
    return(piles, perdus)

def automate(P):
    '''gère l'automate et renvoie les piles à l'état final.'''
    piles = initialisation(P)
    perdus = 0
    px.initialiser(P, 2 * P, 20)
    while True :
        for i in range(10):
            piles, perdus = actualise(piles, perdus)
            if perdus == 1000 :
                # permet de s'arrêter dès que 1000 grains sont sortis
                return(piles)
            px.afficher(0.1)
        piles[0] += 1
        px.marquer(0, (2 * len(piles)) - (piles[0]), 0)


## programme principal
#########################

piles_finales = automate( int(input("Taille du support ? ")) )

px.afficher()
```