TP : Simulation d'un phénomène physique : la percolation.

> "Un mathématicien est une machine qui transforme le café en théorèmes."
>
> 
>
> Paul Erdos

1. La percolation.

> "Le concept mathématique de percolation a été formulé par le mathématicien anglais J. M. Hammersley, en 1957. Il cherchait à décrire le passage d'un fluide à travers un milieu poreux. Peu à peu, le concept de percolation s'est répandu dans de nombreux domaines. Généralement, il cherche à décrire un phénomène critique (crucial). Avant le seuil de percolation, il n'y a pas d'écoulement. Au-delà de ce seuil, l'écoulement est très large..." http://www.matierevolution.fr/spip.php?article788

Initialement, Hammersley s'est intéressé à ce sujet pour comprendre pourquoi les masques à gaz des soldats pouvaient devenir inefficaces.

Le terme de percolation vient du latin "percolare" : couler à travers.

2. Faire du bon café.

> *Pour obtenir du café, il faut qu'il y ait suffisamment de passage entre les grains pour laisser l'eau filtrer. L'eau peut ne pas passer, soit parce que des pores sont bouchés, soit parce que les connexions entre les pores sont bloquées. Pour avoir du café, il faut que l'eau puisse « percoler ». Il n'est pas si facile de faire du bon café. Vous pourriez penser qu'il n'y a qu'à diluer les grains et avoir des pores grands ouverts. Mais si les pores sont trop grands et contiennent trop d'eau, on extraira bien les arômes, mais le café sera trop dilué. Au contraire, si la poudre est trop serrée, on bouchera aléatoirement trop de pores et... plus de café. (...)*
>
> http://www.matierevolution.fr/spip.php?article788

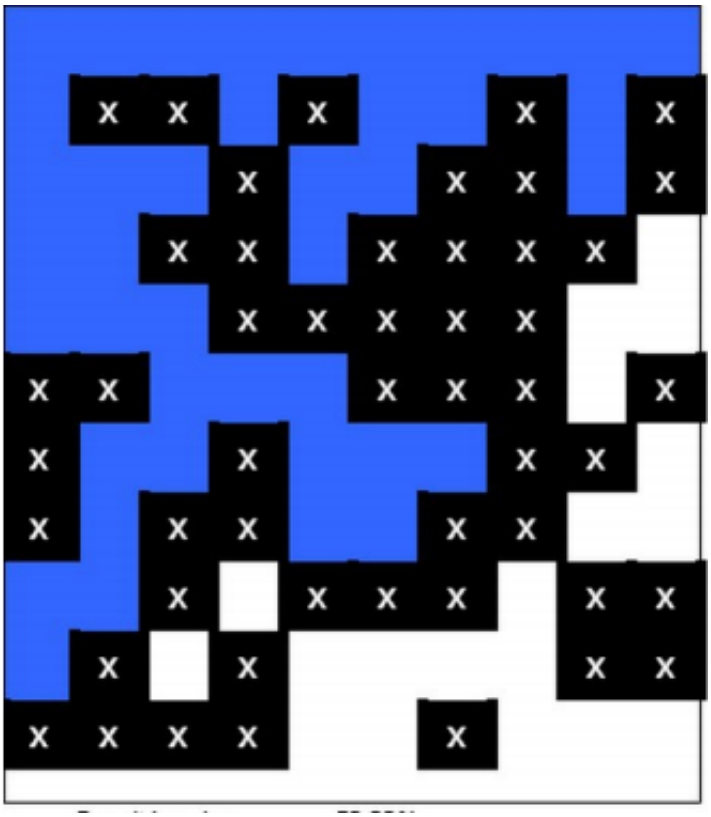
On considère un milieu fermé (ici une grille de dimensions 10 sur 10) contenant du café avec une densité de 50%.

Pour simuler ce milieu, on a rempli aléatoirement une grille de dimensions fixées.

Les carrés en noirs représentent le café, les blancs l'espace libre entre ceux-ci.

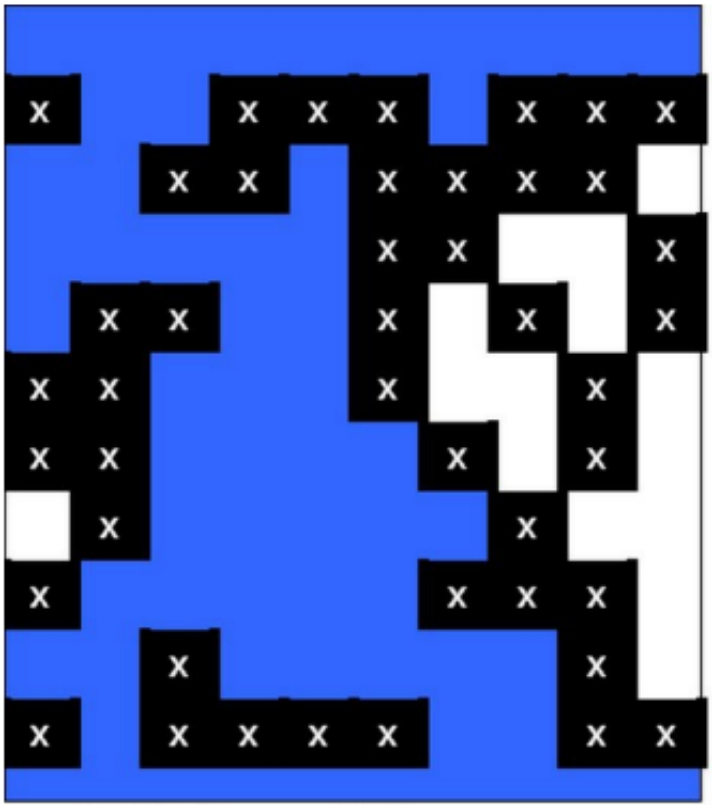
On considère à présent, de l'eau que l'on verse en haut de notre milieu.

On suppose qu'elle ne peut s'écouler que dans les cases vides horizontalement adjacentes ou situées verticalement juste en dessous.



On dira qu'il y a percolation si il existe un chemin permettant à l'eau de s'écouler jusqu'en bas de notre milieu, sinon il n'y a pas percolation (le milieu est alors imperméable).

Dans ce premier cas, il n'y a pas percolation.



Ici, il y a percolation.

Ce modèle peut-être étendu à d'autres domaines de la physique (conductivité d'un matériaux si l'on remplace le café par des conducteurs électriques), de la médecine (propagation d'un virus), de la sécurité (modèle de simulation de propagation d'incendies) ….

3. Construction de nos milieux et un peu de statistiques et de probabilités en mathématiques.

On va considérer qu'un milieu est défini par ses dimensions (une grille de largeur l et de hauteur h donnée), et une densité d (un nombre compris entre 0 et 1).

Pour déterminer si une case de notre grille constitue est rempli par du café, on va tirer un nombre aléatoire entre 0 et 1, s'il est inférieur ou égal à la densité, on considérera qu'il est rempli.

Dans le cas contraire, la case sera considérée comme vide.

a) Premières fonctions : on génère aléatoirement une simulation de notre milieu.

Une structure de données satisfaisante pour représenter notre milieu est une grille sous la forme d'un tuple de tuples.

Par exemple :

```
grille =(
(0,0,0,0,0,0,0,0,0,0),
(1,0,0,1,1,1,0,1,1,1),
...
...
...
...
...
...
...
(0,0,0,0,0,0,0,0,0,0)
)
```

Pour générer notre grille aléatoire on aura besoin d'importer la méthode random.

Pour l'affichage, on importera des éléments des bibliothèques matplotlib et numpy.

Compléter la fonction grille pour qu'elle réponde à sa docstring :

```
from random import *
import matplotlib.pyplot as plt
import numpy as np


def grille(largeur,hauteur,densite):
    '''
    génère une grille sous la forme d'un tuple de tuples
    de largeur et hauteur données
    chaque élément de la grille a la probabilité densité
    d'être plein
    largeur : int
    hauteur : int
    densite : float
    return : tuple
    exemple : grille(10,10,0.1) retournera une grille aléatoire
    du type
    ((0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
    (0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
    (0, 1, 0, 0, 0, 0, 0, 0, 0, 0),
    (0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
    (0, 0, 0, 0, 0, 1, 0, 0, 0, 1),
    (0, 0, 0, 0, 0, 0, 0, 0, 1, 0),
    (0, 0, 1, 0, 0, 0, 0, 1, 0, 0),
    (0, 0, 0, 0, 0, 0, 0, 0, 1, 0),
    (0, 0, 0, 1, 0, 1, 0, 0, 0, 1),
    (0, 1, 0, 0, 0, 0, 0, 1, 0, 0))
    '''


```

```
31
32
33
34
35
36
37
38
39
40
```

La fonction affichage :

```
1  def afficher(grille):
2      '''
3      affichage de la grille
4      '''
5      hauteur=len(grille)
6      largeur=len(grille[0])
7      grille_a_afficher=np.zeros((largeur,hauteur))
8      for i in range(hauteur):
9          for j in range(largeur):
10             grille_a_afficher[j,i]=grille[j][i]
11     plt.spy(grille_a_afficher)
12     plt.show()
```

Notre grille est définie avec une densité théorique fixée. Le contenu de chaque élément étant défini aléatoirement en fonction de cette densité, la fréquence des cases de notre grille ne correspond par exactement à la densité. On peut le vérifier en implémentant une fonction permettant de déterminer cette fréquence pour une grille donnée :

```
1  def frequence(grille):
2      '''
3      retourne la frequence des 1 présents dans la grille
4      grille : tuple de tuples
5      return : float
6      >>> g=((0, 0, 0, 0, 0, 0, 1, 0, 0, 0),
7      (0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
8      (0, 0, 0, 0, 0, 0, 0, 1, 0, 1),
9      (0, 0, 0, 1, 1, 0, 0, 0, 0, 0),
10     (0, 0, 0, 0, 0, 0, 0, 1, 0, 0),
11     (0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
12     (0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
13     (0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
14     (0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
15     (0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
16     >>>print(frequence(g))
17     0.06
```

```
    '''
```

Compléter cette fonction pour qu'elle correspondent à sa docstring.

Faites quelques essais avec différentes tailles de grille.

Que peut-on dire pour la fréquence relevée par rapport à la densité fixée en fonction des dimensions de la grille ?

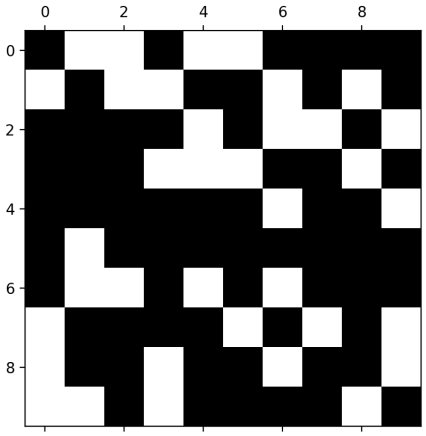
.....

.....

b) On cherche à déterminer si l'eau peut passer du haut vers le bas.

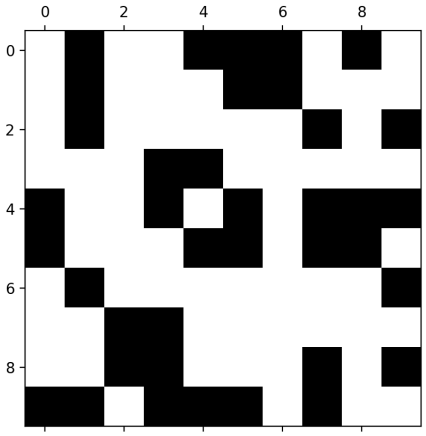
En percolation, on appelle amas ou cluster un ensemble de cellules de même type qui sont adjacentes horizontalement ou verticalement.

Nous allons donc chercher des amas de cases vides en partant du haut de notre grille.



Colorier les 2 amas de cases vides en partant du haut dans cet exemple de grille.

Combien il y a-t-il d'amas répondant à notre problématique dans l'exemple de milieu ci-dessous ?

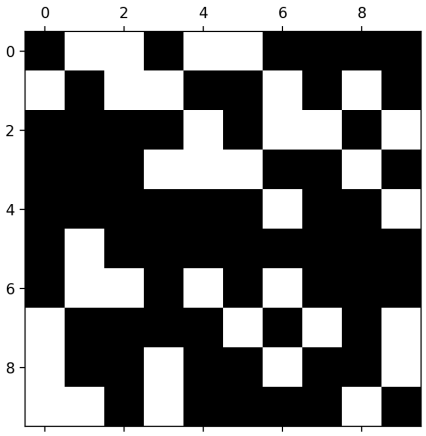


Il faut donc à présent créer une fonction permettant de déterminer les amas de cases vides en partant du haut pour une grille donnée.

Chaque amas sera sous la forme d'un tuple contenant les coordonnées des cases formant cet amas.

Notre fonction amas nous renverra un tuple formé de ces différents amas.

Pour l'exemple ci-dessous :

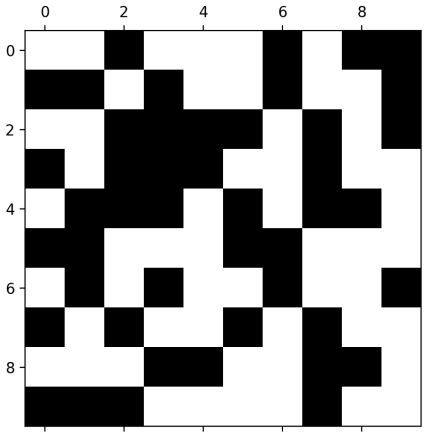


Que doit nous renvoyer notre future fonction amas ?

.....

.....

Notre fonction amas devra répondre à sa docstring dont l'exemple correspond à la grille :




```
def amas(grille):
    '''
    détermine les amas de cases vides de la grille
    en partant du haut et de manière ordonnée
    grille : tuple de tuples contenant des 0 et des 1
    return : tuple constitué de tuples contenant les coordonnées
    des cases formant un même amas
    >>>g=((0, 0, 1, 0, 0, 0, 1, 0, 1, 1),
    (1, 1, 0, 1, 0, 0, 1, 0, 0, 1),
    (0, 0, 1, 1, 1, 1, 0, 1, 0, 1),
    (1, 0, 1, 1, 1, 0, 0, 1, 0, 0),
    (0, 1, 1, 1, 0, 1, 0, 1, 1, 0),
    (1, 1, 0, 0, 0, 1, 1, 0, 0, 0),
    (0, 1, 0, 1, 0, 0, 1, 0, 0, 1),
    (1, 0, 1, 0, 0, 1, 0, 1, 0, 0),
    (0, 0, 0, 1, 1, 0, 0, 1, 1, 0),
    (1, 1, 1, 0, 0, 0, 0, 1, 0, 0))
    >>> amas(grille)==(((0,0),(1,0)),
    ((3,0),(4,0),(5,0),(4,1),(5,1)),
    ((7,0),(7,1),(8,1),(8,2),(8,3),(9,3),
    (9,4),(7,5),(8,5),(9,5),(7,6),(8,6),
    (8,7),(9,7),(9,8),(8,9),(9,9)))
    True
    '''
```

A vous de compléter la fonction (on pourra recourir à plusieurs autres fonctions pour décomposer le problème).

c) Il ne reste plus qu'à déterminer si il y a percolation

A vous de déterminer une fonction qui permettra de déterminer si pour une grille donnée, il y a ou non percolation.