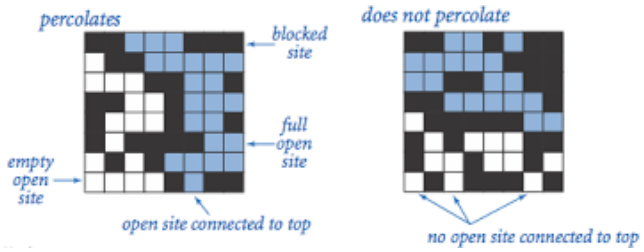
Percolation Problem with Numpy

In this assignment, you will be modeling the process of percolation, the process of a fluid slowly traveling through a porous material.
Example: If a liquid is poured on top of a porous material, such as a porous rock or a layer of soil, will it reach the bottom of the material?



- We will create an nxn square matrix (2D numpy array) to model the open and blocked sites of the porous material.
- Each site (or element of the array) has the same site vacancy p, or the probability of each site to be open. (For eg, if p = .7 in a 10x10 matrix, each site will have a 70% chance to be open and 30% to be closed when the matrix is randomly created)
- Your job is to create a program percolation.py that prompts the user to input:

  1) n: the dimension of the matrix (= #elements in one row or column)
  2) value of p: site vacancy probability

- The program will then randomly generate a nxn matrix with site percolation p. It will print out three things:
  1) randomly generated nxn matrix
  2) A statement about whether or not it percolates
  3) A fluid flow matrix or visual grid (your choice) depicting all full sites. (see below for example)

e.g.
1) Suppose I randomly generated the following matrix (1 for open, 0 for closed)
[[1 0 1 1 1 0 0]
 [0 1 0 1 1 1 0]
 [0 0 1 1 0 0 1]
 [1 1 1 1 1 1 1]
 [1 1 1 1 1 1 0]
 [1 0 1 1 0 1 0]
 [1 1 0 1 1 0 0]]

2) Then I would print out the fluid flow matrix (0 for closed, 1 for open and empty, **2 for open and full**)

[[**2** 0 **2 2 2** 0 0]
 [0 1 0 **2 2 2** 0]
 [0 0 **2 2** 0 0 1]
 [**2 2 2 2 2 2 2**]
 [**2 2 2 2 2 2** 0]
 [**2** 0 **2 2** 0 **2** 0]
 [**2 2** 0 **2 2** 0 0]]
** note that any **open and full site** can be reached from other open and full sites in the first row by a combination of steps going left, right, and down (diagonals don't count)

3. Then I see there are open and full sites in the last row which means the system percolates → print statement to user

Final Goal: Complete and submit the **percolation.py** file available on the course website. Do not modify the functions that are already there; those are the crucial parts of the program. You can add more functions as needed.

Suggested step by step process (Perhaps use different .py files for each step as you don't want to worry about errors in the other sections)

1) Hardcode any matrix of 0s and 1s of a certain size. Start small! Come up with an algorithm to create a fluid flow matrix. This is the essential part of the show_flow function (hint: recursion worked for me; maybe there are other ways!)
2) In a separate .py file, code for an algorithm that creates a matrix of site vacancy p.
   - Note this does not mean that if there are 100 sites and p = .7, exactly 70 of them should be randomly assigned a 1.
   - Instead, each site should be individually, randomly determined as either a 1 (with probability p) or 0 (with probability 1-p).
   - There are many ways to do this! Consider using arithmetic operations or Boolean operations on the numpy arrays. Hint: something similar was seen during class (look at lecture notes!)
3) Put everything together on percolation.py. By now, you should be able to write the percolates function and main function.

**Tip: When you test your code, the console will show you an abbreviated version of your matrix when it gets too large (over 6*6). To avoid this and see the entire array,
Write:

```
import numpy as np
np.set_printoptions(threshold=np.inf)
```

at the top of your code to make the entire array print.