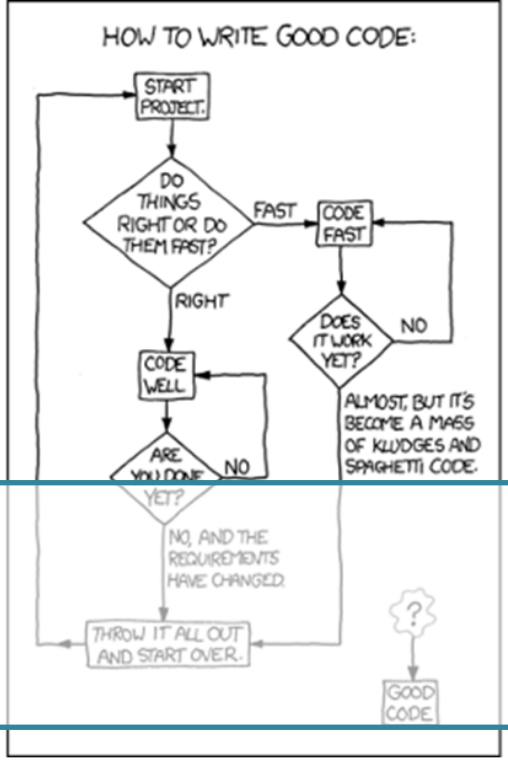

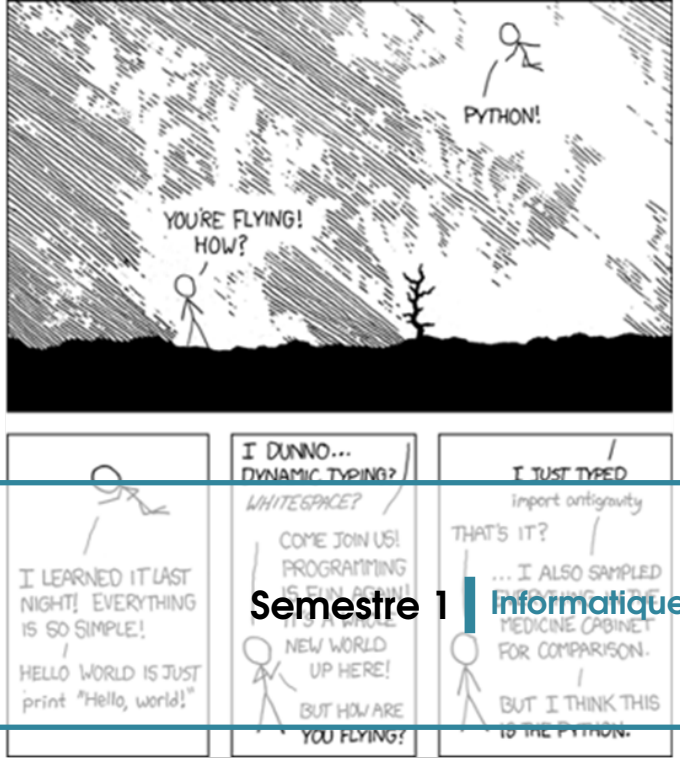

Semestre 1 | Informatique

Thèmes d'étude

Thème : Algorithmes opérant sur une structure séquentielle par boucles imbriquées. Commentaires :
- recherche d'un facteur dans un texte ;
- recherche des deux valeurs les plus proches dans un tableau ;
- tri à bulles ;
- notion de complexité quadratique
- outils pour valider la correction de l'algorithme

1 Parcours d'une liste de listes

Les listes de listes permettent de mettre les données en deux dimensions.

▪ **Exemple**

Grille de mots mêlés.

L	E	S
E	T	E
S	E	C

```
grille = [['L','E','S'],['E'
    ,'T','E'],['S','E','C']]
```

Table de multiplication

×	1	2	3
1	1	2	3
2	2	4	6
3	3	6	9

```
table =
    [[1,2,3],[2,4,6],[3,6,9]]
```

Température en fonction du temps. ▪

T (s)	1	2	3	4
T°C	18	19	21	24

```
data = [[1,18],
    [2,19],[3,21],[4,24]]
```

Pour parcourir les éléments d'un tableau on procède de la même façon que pour une recherche séquentielle. Prenons l'exemple d'un tableau `tab` de `n` lignes et p colonnes.

Utilisation de boucles `while`

```
n = len(tab)
p = len(tab[0])
i,j = 0,0
while i<n :
    while j<p :
        print(tab[i][j])
        j = j+1
    j=0
    i=i+1
```

Utilisation de boucles `for`

```
n = len(tab)
p = len(tab[0])
for i in range(n) :
    for j in range(p) :
        print(tab[i][j])
```

Écriture de boucles `for` en `Python`

```
for t in tab :
    for e in t :
        print(e)
```

R Il est possible de dénombrer le nombre d'itérations réalisées par les algorithmes ci-dessus. Dans chaque cas, la première boucle est réalisée n fois. La seconde boucle, imbriquée dans la première est parcourue p fois. On peut donc dénombrer le nombre de fois que la fonction `print` est appelée : $n \times p$.
Une estimation grossière du nombre d'opérations réalisées en tout est donc $n \times p$. On dit que la complexité dans ces algorithmes, dans le pire des cas est $\mathcal{O}(np)$. Si $n = p$, la complexité est de $\mathcal{O}(n^2)$. On parle de complexité quadratique.

2 Recherche de facteur dans un mot

Rechercher un facteur dans un mot signifie rechercher une (sous-)chaîne de caractères dans une chaîne de caractères (ou encore un mot dans une chaîne).

```
def recherche_01(m:str, s:str) -> bool:
    """Recherche le mot m dans la chaine s
       Préconditions : m et s sont des chaines de caractères"""
    long_s = len(s) # Longueur de s
    long_m = len(m) # Longueur de m
    for i in range(long_s-long_m+1):
        # Invariant : m n'a pas été trouvé dans s[0:i+long_m-1]
        j = 0
        while j < long_m and m[j] == s[i+j]:
            # Invariant : m[:j] == s[i:i+j]
            j = j+1
            # Invariant : m[:j] == s[i:i+j]
        if j == long_m:
            # Invariant précédent : m == s[i:i+long_m]
            return True
    return False
```

Cet algorithme est simplifiable en utilisant le slicing.

```
def recherche_02(m:str, s:str) -> bool:
    """Recherche le mot m dans la chaine s
       Préconditions : m et s sont des chaines de caractères"""
    long_s = len(s) # Longueur de s
    long_m = len(m) # Longueur de m
    for i in range(long_s-long_m+1):
        # Invariant : m n'a pas été trouvé dans s[0:i+long_m-1]
        if s[i:i+long_m] == m: # On a trouvé m
            return True
    return False
```

En utilisant les possibilités de `Python`, il est possible de simplifier encore l'algorithme.

```
def recherche_03(m:str, s:str) -> bool:
    """Recherche le mot m dans la chaine s
       Préconditions : m et s sont des chaines de caractères"""
    return m in s
```

Pour réaliser l'activité associée à ce cours, suivre le lien suivant : `https://bit.ly/3AmRgdH`

Question de cours

Question 1 *Dans l'algorithme de recherche de mot dans une chaîne, expliquer pourquoi la boucle principale (boucle* `for` *est itérée* `long_s-long_m+1` *?*

Question 2 *Pourquoi la condition* `j<long_m` *est une condition d'arrêt de la boucle* `while` *?*

QCM

Question 3 *Quelle est la valeur de la variable* `image` *après exécution du programme* `Python` *suivant?*

```
image = [[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]
for i in range(4) :
    for j in range(4) :
        if (i+j) == 3 :
            image[i][j] = 1
```

1. `[[0,0,0,1],[0,0,1,0],[0,1,0,0],[1,0,0,0]]`.
2. `[[0,0,0,1],[0,0,0,1],[0,0,0,1],[0,0,0,1]]`.
3. `[[0,0,0,1],[0,0,1,1],[0,1,1,1],[1,1,1,1]]`.
4. `[[0,0,0,0],[0,0,0,0],[0,0,0,0],[1,1,1,1]]`.

Question 4 *Quelle est la valeur de la variable* `table` *après exécution du programme* `Python` *suivant?*

```
table = [12, 43, 6, 22, 37]
for i in range(len(table) - 1):
    if table [i] > table [i+1] :
        table [i] ,table [i+1] = table [i+1] ,table [i]
```

1. `[12, 6, 22, 37, 43]`.
2. `[6, 12, 22, 37, 43]`.
3. `[43, 12, 22, 37, 6]`.
4. `[43, 37, 22, 12, 6]`.

Question 5 *On considère le programme suivant. Quelle est la valeur de maxi(L)?*

```
def maxi(tab):
    """
    tab est une liste de couples (nom, note)
      * nom est de type str
      * note est un entier entre 0 et 20.
    """
    m = tab[0]
    for x in tab:
        if x[1] >= m[1]:
            m = x
    return m
L = [('Adrien', 17), ('Barnabe', 17), ('Casimir', 17), ('Dorian', 17), ('Emilien', 16), ('Fabien
    ', 16)]
```

1. (’Adrien’,17).
2. (’Dorian’,17).
3. (’Fabien’,16).
4. (’Emilien’,16).

Question 6 *Que contient la variable compteur à la fin de l'exécution de ce script?*

```
liste = [0, 1, 2, 3]
compteur = 0
for i in range(len(liste)-1) :
    for j in range(i,len(liste)) :
        compteur += 1
```

1. 4.
2. 8.
3. 9.

4. 10.

Question 7

On considère la liste de p-uplets suivante :

```
table = [ ('Grace', 'Hopper', 'F', 1906), ('Tim', 'Berners-Lee', 'H', 1955), ('Ada', 'Lovelace',
     'F', 1815), ('Alan', 'Turing', 'H', 1912) ]
```

où chaque p-uplet représente un informaticien ou une informaticienne célèbre ; le premier élément est son prénom, le deuxième élément son nom, le troisième élément son sexe ('H' pour un homme, 'F' pour une femme) et le quatrième élément son année de naissance (un nombre entier entre 1000 et 2000).

On définit une fonction :

```
def fonctionMystere(table):
    mystere = []
    for ligne in table:
        if ligne[2] == 'F':
            mystere.append(ligne[1])
    return mystere
```

Que vaut fonctionMystere(table) ?

1. ['Grace', 'Ada'].
2. [('Grace', 'Hopper', 'F', 1906), ('Ada', 'Lovelace', 'F', 1815)].
3. ['Hopper', 'Lovelace'].
4. [].

Question 8 *Quelle est la valeur de la variable table à la fin de l'exécution du script suivant ?*

```
table = [[1, 2, 3], [1, 2, 3], [1, 2, 3], [1, 2, 3]]
table [1][2] = 5
```

1. [[1, 5, 3], [1, 2, 3], [1, 2, 3], [1, 2, 3]].
2. [[1, 2, 3], [5, 2, 3], [1, 2, 3], [1, 2, 3]].
3. [[1, 2, 3], [1, 2, 5], [1, 2, 3], [1, 2, 3]].
4. [[1, 2, 3], [1, 2, 3], [1, 2, 3], [1, 5, 3]].

Question 9 *Soit le tableau défini de la manière suivante :* `tableau = [[1,3,4],[2,7,8],[9,10,6],[12,11,5]]`
On souhaite accéder à la valeur 12.

1. tableau[4][1].
2. tableau[1][4].
3. tableau[3][0].
4. tableau[0][3].

Question 10 *Une erreur s'est glissée dans le tableau, car le symbole du Fluor est F et non Fl. Quelle instruction permet de rectifier ce tableau ?*

```
mendeleiev = [ ['H','.', '.','.','.','.','.','He'],
               ['Li','Be','B','C','N','O','Fl','Ne'],
               ['Na','Mg','Al','Si','P','S','Cl','Ar'],
                 ..... ]
```

1. mendeleiev.append('F').
2. mendeleiev[1][6] = 'F'.
3. mendeleiev[6][1] = 'F'.
4. mendeleiev[-1][-1] = 'F'.

Question 11 *Quelle est la valeur de la variable t1 à la fin de l'exécution du script suivant ?*

```
t1 = [['Valenciennes', 24],['Lille', 23],['Laon', 31],['Arras', 18]]
t2 = [['Lille', 62],['Arras', 53],['Valenciennes', 67],['Laon', 48]]
for i in range(len(t1)):
    for v in t2:
        if v[0] == t1[i][0]:
            t1[i].append(v[1])
```

1. [['Valenciennes', 67], ['Lille', 62], ['Laon', 48], ['Arras', 53]].
2. [['Valenciennes', 24, 67], ['Lille', 23, 62], ['Laon', 31, 48], ['Arras', 18, 53]].
3. [['Arras', 18, 53],['Laon', 31, 48], ['Lille', 23, 62], ['Valenciennes', 24, 67]].
4. [['Valenciennes', 67, 24], ['Lille', 62,23], ['Laon', 48, 31], ['Arras', 53, 18]].

Question 12 *Que vaut asso à la fin de l'exécution ?*

```
asso = []
L = [ ['marc','marie'], ['marie','jean'], ['paul','marie'], ['marie','marie'], ['marc','anne'] ]
for c in L :
    if c[1]=='marie':
        asso.append(c[0])
```

1. ['marc', 'jean', 'paul'].
2. [['marc','marie'], ['paul','marie'], ['marie','marie']].
3. ['marc', 'paul', 'marie'] .
4. ['marie', 'anne'].

Question 13 *Quelle est la valeur de x après exécution du programme ci-dessous ?*

```
t = [[3,4,5,1],[33,6,1,2]]
x = t[0][0]
for i in range(len(t)):
    for j in range(len(t[i])):
        if x < t[i][j]:
            x = t[i][j]
```

1. 3.
2. 5.
3. 6.
4. 33.

Exercice 1 – Percolation

Pour ce TP, on utilisera le fichier `Percolation_Sujet.py` que vous renommerez en utilisant la convetion suivante : `TP_n_Nom1_Nom2.py` où `n` désigne le numéro du TP et `Nom1` et `Nom2` vos noms.

La percolation [1] désigne le passage d'un fluide à travers un solide poreux. Ce terme fait bien entendu référence au café produit par le passage de l'eau à travers une poudre de café comprimée, mais dans un sens plus large peut aussi bien s'appliquer à l'infiltration des eaux de pluie jusqu'aux nappes phréatiques ou encore à la propagation des feux de forêt par contact entre les feuillages des arbres voisins.

L'étude scientifique des modèles de percolation s'est développée à partir du milieu du XXe siècle et touche aujourd'hui de nombreuses disciplines, allant des mathématiques à l'économie en passant par la physique et la géologie.

Choix d'un modèle

Nous allons aborder certains phénomènes propres à la percolation par l'intermédiaire d'un modèle très simple : une grille carrée $n \times n$, chaque case pouvant être ouverte (avec une probabilité p) ou fermée (avec une probabilité $1-p$). La question à laquelle nous allons essayer de répondre est la suivante : est-il possible de joindre le haut et le bas de la grille par une succession de cases ouvertes adjacentes ?

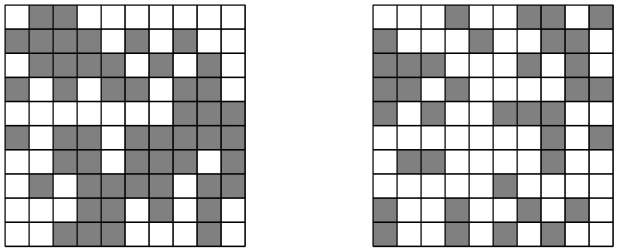

FIGURE 1 – Deux exemples de grilles 10×10. La percolation n'est possible que dans le second cas (les cases ouvertes sont les cases blanches).

Création et visualisation de la grille

La grille de percolation sera représentée par le type `list`. Cette grille sera elle-même composée de `n` listes de `n` flottants (tableau $n \times n$). Une fois la grille `grille` créée, la case d'indice (i, j) est référencée par `grille[i][j]`.

Afin de remplir la grille, on dispose de la fonction `rd.random()` permettant de renvoyer un nombre aléatoire compris dans l'intervalle $[0, 1[$.

Afin de visualiser la grille et de sauvegarder la grille on dispose des fonctions `afficher_grille(grille : list) -> None` et `sauvegarder_grille(grille : list, nom_de_fichier : str) -> None`.

Question 1 *Définir une fonction* `Python`, *de signature* `def creation_grille(p: float, n: int) -> list` : *à deux paramètres : un nombre réel p (qu'on supposera dans l'intervalle $[0, 1[$ et un entier naturel n, qui renvoie un tableau (n, n) dans lequel chaque case sera ouverte avec la probabilité p et fermée sinon.*

Pour afficher la grille, vous pourrrez utiliser les instructions suivantes :

```
>>> grille = creation_grille(0.6, 10)
>>> afficher_grille(grille)
```

Percolation

Une fois la grille crée, les cases ouvertes de la première ligne sont remplies par un fluide, ce qui sera représenté par la valeur 0.5 dans les cases correspondantes. Le fluide pourra ensuite être diffusé à chacune des cases ouvertes voisines d'une case contenant déjà le fluide jusqu'à remplir toutes les cases ouvertes possibles.

Question 2 *Écrire une fonction* `percolation(grille : list) -> None` *qui prend en argument une grille et qui remplit de fluide celle-ci, en appliquant l'algorithme exposé ci-dessous :*

1. *Créer une liste contenant initialement les coordonnées des cases ouvertes de la première ligne de la grille et remplir ces cases de liquide.*
2. *Puis, tant que cette liste n'est pas vide, effectuer les opérations suivantes :*
   (a) *extraire de cette liste les coordonnées d'une case quelconque ;*
   (b) *ajouter à la liste les coordonnées des cases voisines qui sont encore vides, et les remplir de liquide.*

L'algorithme se termine quand la liste est vide.

Question 3 *Rédiger un script vous permettant de visualiser une grille avant et après remplissage, et faire l'expérience avec quelques valeurs de p pour une grille de taille raisonnable (commencer avec $n = 10$ pour vérifier visuellement que votre algorithme est correct, puis augmenter la taille de la grille, par exemple avec $n = 64$). On pourra exporter et l'enregistrer sous le nom* `tp_n_q03_vos_noms.png`.

On dit que la percolation est réussie lorsqu'à la fin du processus au moins une des cases de la dernière ligne est remplie du fluide.

Question 4 *Écrire une fonction* `teste_percolation(p : float, n : int) -> bool` *qui prend en argument un réel $p \in [0, 1[$ et un entier $n \in \mathbb{N}^*$, crée une grille, effectue la percolation et renvoie :*

- `True` lorsque la percolation est réussie, c'est-à-dire lorsque le bas de la grille est atteint par le fluide ;
- `False` dans le cas contraire.

1. du latin *percolare* : couler à travers.

Seuil critique

Nous allons désormais travailler avec des grilles de taille 128×128 [2]

Faire quelques essais de percolation avec différentes valeurs de p. Vous observerez assez vite qu'il semble exister un seuil p_0 en deçà duquel la percolation échoue presque à chaque fois, et au delà duquel celle-ci réussit presque à chaque fois. Plus précisément, il est possible de montrer que pour une grille de taille infinie, il existe un seuil critique p_0 en deçà duquel la percolation échoue toujours, et au delà duquel la percolation réussit toujours. Bien évidemment, plus la grille est grande, plus le comportement de la percolation tend à se rapprocher du cas de la grille théorique infinie.

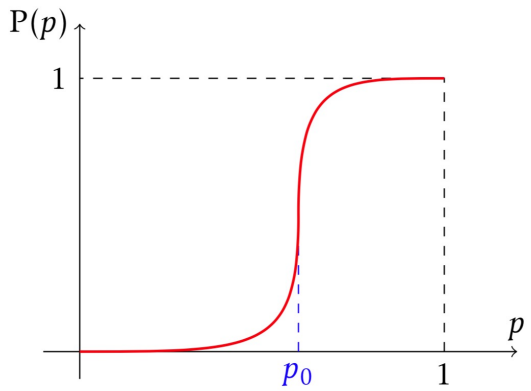

FIGURE 3 – L'allure théorique du graphe de la fonction $P(p)$.

Notons $P(p)$ la probabilité pour le fluide de traverser la grille. Pour déterminer une valeur approchée de la probabilité de traverser la grille, on se contente d'effectuer k essais pour une valeur de p puis de renvoyer le nombre moyen de fois où le test de percolation est vérifié.

Question 5 *Rédiger la fonction* `proba(p : float, k : int, n : int) -> float` *qui prend en argument le nombre d'essai k, la variable p ainsi que le nombre de cases n sur la largeur de la grille et qui renvoie* $P(p)$.

Question 6 *Écrire une fonction* `tracer_proba(n : int ,nom_de_fichier : str) -> None` *qui prend en argument une taille n ne renvoyant rien mais enregistrant dans* `nom_de_fichier` *le graphe obtenu. On pourra traiter le cas d'une grille de* 128×128 *cases et enregistrer la figure obtenue sous le nom* `tp_n_q06_vos_noms.png`.

2. Baisser cette valeur si le temps de calcul sur votre ordinateur est trop long.