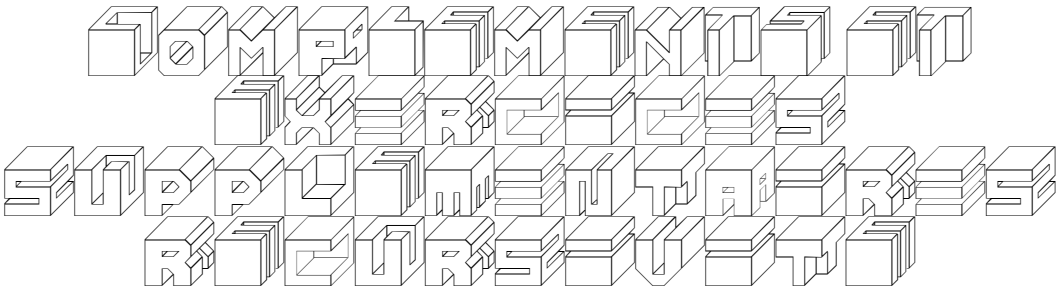


Micro-cours.

Définition : une fonction récursive est une fonction qui s'appelle elle-même.

Indispensables :

- Une fonction récursive doit comporter une ou des condition(s) d'arrêt ;
- Les appels de la fonction à l'intérieur d'elle-même doivent avoir des paramètres différents ;
- Et ces paramètres doivent permettre de se « rapprocher » de la (des) condition(s) d'arrêt.

Contre-exemples :

- ```
  Fonction fact_sans_arrêt(n entier ≥ 0) :
        renvoyer n × fact_sans_arrêt(n - 1)
  ```
- ```
  Fonction fact_même_appel(n entier ≥ 0) :
        si n = 0 :
              renvoyer 1
        sinon :
              renvoyer n × fact_même_appel(n)
  ```
- ```
  Fonction fact_eloigne_arret(n entier ≥ 0) :
        si n = 0 :
              renvoyer 1
        sinon :
              renvoyer n × fact_eloigne_arrêt(n + 1)
  ```

Complexité :

Certains calculs de complexité d'une fonction récursive peuvent être donnés par le tableau suivant, qui n'est pas exhaustif :

Lien entre l'appel de la fonction T_n avec une donnée de taille n et le ou les appel(s) T_{n-1} récursif, avec une donnée de taille $n-1$	Complexité de la fonction
La fonction est de complexité constante et il y a un seul appel récursif : $T_n = T_{n-1} + O(1)$	On ajoute une opération (complexité constante) par appel d'où : $O(n)$
La fonction est de complexité linéaire et il y a un seul appel récursif : $T_n = T_{n-1} + O(n)$	On ajoute n opérations (complexité linéaire) par appel d'où : $O(n^2)$
La fonction est de complexité constante et il y a deux appels récursif : $T_n = 2T_{n-1} + O(1)$	On double les appels lors de chaque passage dans la fonction récursive d'où : $O(2^n)$
La fonction est de complexité constante et la taille des données est divisée par deux : $T_n = T_{n/2} + O(1)$	On divise par 2 la taille des données lors de chaque passage dans la fonction récursive d'où : $O(\log_2 n)$

Exemples :

- ```
  Fonction fact (n entier ≥ 0) :
        si n = 0 :
              renvoyer 1
        sinon :
              renvoyer n × fact (n - 1)
  ```

  La factorielle récursive est de complexité $O(n)$

- ```
  Fonction fibonacci (n entier ≥ 0) :
        si n = 0 ou n = 1:
              renvoyer 1
        sinon :
              renvoyer fibonacci (n - 1) + fibonacci (n - 2)
  ```

  Le calcul d'un terme de la suite de Fibonacci par récursivité est de complexité $O\left(2^n\right)$

Exercice 1.

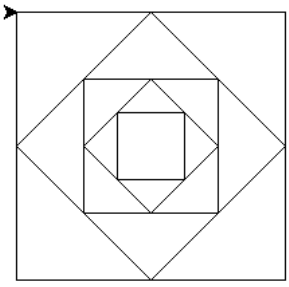
On dispose de tableaux « enchâssés » les uns dans les autres, sous la forme [[[[[[[.....[].....]]]]]]]. Écrire une fonction récursive et une fonction itérative donnant le nombre total de tableaux.
On suppose maintenant qu'il y a soit 0 soit 2 tableaux à l'intérieur d'un autre tableau. Écrire une fonction récursive donnant le nombre de tableaux. Est-il facile d'écrire un programme itératif donnant ce nombre ? Quelques exemples en annexe.
Complément/remarque : Vous pouvez aussi récupérer le programme de génération auprès du professeur, mais seulement après résolu l'exercice. En effet ce programme vous donnera trop d'idées pour faire l'exercice. D'ailleurs vous pouvez aussi faire un programme de génération, notamment pour la deuxième version. Dans ce cas une méthode est de ne pas créer de tableau supplémentaire avec une chance sur deux, et d'en créer 2 avec une chance sur deux. Que pensez-vous de la terminaison de cet algorithme ? Que remarquez-vous après avoir fait plusieurs tests ?

Exercice 2.

Récupérer le programme « affiche_demineur.py ». Ce programme affiche une grille de démineur. Chaque case est un dictionnaire de la forme {"mine": True/False , "cache": True/False , "bombes_v": entier ≥0 }. La case de coordonnées (0 ; 0) n'a pas de mine par principe.
Le compléter en écrivant la fonction « devoiler(i , j) ».
Cette fonction prend en entrée une case de coordonnées i et j, a priori cachée (le booléen est à False)
S'il y a une bombe sur la case, la dévoile (booléen caché à True) et s'arrête ;
S'il y a au moins un voisin avec une bombe, la dévoile (booléen caché à True) et se termine ;
Sinon dévoile toutes les cases voisines cachées (booléen caché à True) et itère le processus sur ces cases.
La fonction « afficheGrille(grille) » permet d'avoir l'affichage console avec les cases dévoilées.
Des exemples de grilles sont donnés dans le programme.

Exercice 3.

1. Activité préliminaire sur papier : dessiner sans lever le stylo et sans passer deux fois sur le même trait la figure suivante



2. Proposer une fonction récursive, à 2 paramètres (la longueur du plus grand carré, le nombre de carrés imbriqués) qui réalise le tracé.

Exercice 4. *difficile*

On dispose de n inégalités disposées dans un tableau `inegs`.
On dispose de $n+1$ nombres entiers triés dans un tableau `nbs`.
Écrire un algorithme (en Python ou en langage naturel) qui permet d'obtenir un tableau `nbs_ok` contenant tous les nombres de `nbs` dans un ordre tel qu'ils respectent les inégalités de `inegs`. Voici un exemple :

```
inegs = [ '<', '<', '>', '>', '<']
nbs = [ 3, 13, 13, 17, 19, 23]
```

`nbs_ok = [13, 17, 19, 13, 3, 23]` convient puisqu'on a bien : $13 < 17 < 19 > 13 > 3 < 19$.
```

Exemples exercice 1.

1ère version

[[]]]   [[[[[[[[[[[[[[[[[[[[[]]]]]]]]]]]]]]]]]]]]]

2ème version (le défi est de les compter sur le papier, et ceux qui ne sont pas sages auront comme punition de recopier le dernier tableau 500 fois à la main ☺)

[[], [[], [[], []]] [[[[], []], []], [[[], [[], []]], []]]   [[[[], [[[], []], [[], []]]], [[[], [[[], []], []]], []]], []]

[[[], [[], []]], [[[], []], [[[[], []], [[], []]], [[[[], [[], []]], [[], []]], [[], []]]]]]

[[[], [[], []]], [[[], []], [[[[], []], [[], []]], [[[[], [[], []]], [[[[], [[], []]], [[[], []], [[[[], []], [[[[], [[], []]], [[[], []], [[[[], []], [[],
[]]], [[[[], [[], []]], [[[[], [[], []]], [[[], []], [[[[],[[[[], [[[], []], [[], []]]], [[[], [[[], []], []]], []]], []]], [[], []]], [[[[], [[], []]],
[[], []]], [[], []]]]]], []]], [[[[[], []], []], [[[], [[], []]],[[[[], []], []], [[[], [[], []]], []]]]], []]]]]], []]], [[[[], [[], []]], [[], []]], [[],
[]]]]]], []]], [[[[[], []], []], [[[], [[], []]],[[[[], []], []], [[[], [[], []]], []]]]], []]]]]]

[[[[[], [[[], []], [[], [[[], [[[[], []], []], [[], []]]], [[], [[], []]]]]]], []], [[], [[], [[[], [[], []]], [[], [[[], [[[[[[], []], [[], [[], []]]],
[[[[], []], [[], [[[], [[[[], []], []], [[[], []], [[[], []], []]]]], [[[[], [[[], []], []]], []], [[[[[], []], [[], []]], []], [[], [[], [[[], [[], []]],
[[], [[[], [[], []]], []]]]]]], [[], [[], [[], []]]]]]]]], [[], []]]], [[[[], [[[], [[], []]], []]], [[], [[[[], [[[[[], []], [[[[], []], [[[[[[[[[[[],
[]], [[[], [[[], []], [[[[[], [[[], []], [[[[[], []], []], []], []]]], []], []], [[[[], [[[[[], [[], []]], []], [[], [[], []]]], []]], [[], []]], []]]]],
[[], []]]], []], []], [[[], [[[], [[[[[[[], []], [[[[[], [[[[], []], [[], [[], []]]], [[], []]]], []], []], [[[], []], [[], [[], [[[[[], [[], []]], []],
[]], [[[], [[], [[[], []], []]]], [[[[], [[], []]], []], [[], [[], []]]]]]]]]]]], [[], [[], [[[], [[[[[[], []], [[[], []], [[], []]]], [[], []]], []], []]],
[]]]]], []], []], []]], [[], [[[], []], [[], [[[[[[], [[], [[], [[[], [[[], [[], []]], []]], []]]]], [[[[[], []], [[[[[], [[[], [[[], [[], [[], []]]], [[],
[]]]], [[], [[[], []], []]]]], [[], []]], [[[[[[], []], [[[[], [[], [[], [[[], [[[[[[], []], []], []], [[[], []], [[], [[[[[[], []], []], [[], []]], [[[[],
[[[], []], []]], []], []]], []]]]], [[], []]]], []]]]], []], [[], []]]], []], []], [[[], [[], [[], []]]], []]]], [[], []]]], [[], [[], [[], []]]]], [[],
[]]]], [[[], []], [[], [[], []]]]], [[[[], [[], []]], []], [[[[[], []], []], [[], []]], []]]], [[], [[[[[[[[[], [[], [[[[[[[[[], [[], []]], []], [[],
[]]], []], []], [[], []]], []], [[[[[], [[], []]], [[[], []], [[], [[[], []], [[], [[], [[], []]]]]]]], []], [[], [[], []]]]]]], [[[], []], [[[], []],
[[[[], []], []], []]]]], [[], [[[[[], [[[[], []], []], []]], []], [[], []]], []]]], [[], [[], []]]], []], [[[[[], []], [[[], []], [[], []]]], [[], []]],
[]]], [[[[[], []], []], []], [[[[], [[[[[[[[], []], [[[[], [[], [[], []]]], [[[], [[[], []], [[[], []], []]]], [[], []]]], [[], []]]], [[], []]], [[[[],
[]], [[], []]], [[[[], []], [[], [[], [[[[[], []], []], [[], [[], [[[], []], []]]]], [[[[[[], [[], []]], []], [[], [[[[], []], []], [[], [[], [[], [[],
[[[], [[[[[[], []], []], []], []], [[], [[], []]]]], [[[[], []], [[], [[[[], []], []], []]]], [[], []]]]]]]]]]], [[], []]], []]]]]], [[[[[], []], [[[[[],
[]], []], [[], []]], []]], []], []]]]], [[], []]], []], [[[], []], []]]], []], [[], []]]]], []]]]]]]]], [[[[[], []], [[[], [[], []]], []]], []], [[[[],
[]], [[[[[], [[], []]], [[], [[], []]]], [[[[], []], [[], [[[[[[], []], []], [[], []]], []], []]]], []]], []]], []]]]], []], [[[], []], []]], []], []],
[]]], [[], [[], [[[[[[], []], [[[], [[], []]], [[[[], [[[], []], [[], [[[[], [[[[[], []], []], []], [[[[[[], [[[], []], [[[[[], [[], []]], []], [[], []]],
[]]]], [[], [[], [[[[], [[[], []], []]], [[[[], []], [[], []]], []]], []]]]], [[[], [[[[[[], [[], [[[[], [[[], [[], []]], [[], [[], [[], []]]]]], []],
[]]]], [[[], [[[[], [[], [[[[], []], [[], [[], []]]], []]]], []], [[[], [[], []]], [[], [[], []]]]]], []]], [[], [[], []]]], []], []]], [[[[], [[], []]],
[]], []]]], [[], []]], [[[[], []], []], []]]]], []], [[[[[[], [[], [[[[], [[], []]], []], [[[], []], [[], []]]]]], []], []], [[], []]], []]]]]], [[], []]],
[[], []]]]], []], [[], [[], [[[[], []], [[], []]], []]]]], [[[[], []], [[], [[[[[[[[[[[], [[[], []], []]], []], []], []], []], []], [[], []]], [[[[],
[[[[[], []], [[], [[], []]]], []], [[], [[[[], [[[[[], []], []], []], [[[], [[], [[], [[], []]]]], [[], []]]]], [[], []]], [[[], []], [[], [[[[[[[[], []],
[]], [[], []]], [[[], [[[], []], [[], [[], []]]]], []]], [[[], []], []]], [[[], []], []]], [[], [[[], [[[[[[], [[], []]], [[[], []], [[[], []], []]]], []],
[[[], [[[], []], [[[], []], []]]], [[], [[], []]]]], [[[], [[[[[[], [[[[], []], [[[], [[], []]], []]], []]], [[], [[], []]]], [[], []]], [[], [[[], [[],
[[[[[[], [[[], []], [[[], []], []]]], [[[[], [[], []]], []], [[], [[[], [[], [[], []]]], [[], []]]]]], []], [[], [[], [[], [[], [[[], [[[], [[], [[], [[],
[]]]]], [[], []]]], []]]]]]], [[], []]]]], [[], [[], []]]]]], []]], []]]], [[], [[], [[], [[[[], [[], [[[[], []], [[[], [[], []]], []]], [[], [[[], [[[],
[]], [[], [[], []]]]], [[[[], []], []], []]]]]]], [[], [[[], []], [[[], [[], []]], [[], [[[[[[], [[[[[], []], [[], []]], [[[], [[[], [[[[], [[], [[], [[[],
[[], [[], [[], [[], []]]]]], [[], []]]]]], [[[[], [[[], [[[[[], []], [[], [[], []]]], [[], [[[], [[], []]], [[], [[], [[[], []], []]]]]]], [[[], [[], []]],
[[[], [[[[], []], []], [[], [[], []]]]], []]]]], []]], []], [[], []]]], [[], [[], []]]]], [[], [[], []]]]], []]], []]], [[[], [[[], []], [[], []]]], []]],
[]], [[[[], [[[], [[[], [[[], [[], [[], []]]], []]], [[], []]]], [[], []]]], [[], [[], []]]], []]], [[], []]]]]]]], []]]]]]]]]]]]], [[[[], [[[], []],
[]]], [[[[], [[], []]], [[], []]], []]], []]], [[[[[[[[[[[], [[[[], [[[], []], []]], [[], [[], []]]], [[], []]]], []], [[[], [[], []]], [[], []]]], [[],
[[], [[], [[[[[], [[], []]], [[[], []], [[], []]]], [[], [[[], [[[[[], [[], [[[], [[], []]], []]]], []], [[[[], []], []], [[], [[], []]]]], []]], [[[[[[],
[[[], []], []]], []], []], []], []]]]], [[], [[[[], [[[], []], []]], [[], [[], []]]], []]]]]]]], []], [[[[], []], []], [[], []]]], []], [[], []]], []],
[[[], []], [[], []]]]]], [[], []]], [[], [[[], []], [[], []]]]]]], []]]]]], [[], []]], []]], [[], []]], []]]], [[[], []], [[], [[[], []], [[[[[], [[],
[[], [[[], []], []]]]], []], []], []]]]]]], [[[], []], []]], [[[], []], []]]], [[[], []], [[], [[[[[[[[], [[[[], []], []], [[], []]]], []], [[], [[],
[[[[[], [[], [[[], []], []]]], []], [[[[], [[[[], [[], []]], [[[], [[], [[], []]]], [[], [[[], []], [[[], []], [[[[[], [[], []]], []], []], []]]]]]], [[],
[]]]], []], [[], [[], []]]]], []]]]], []], [[], [[], []]]], [[[], []], []]], [[[], []], []]]]]]]]]]], [[]]