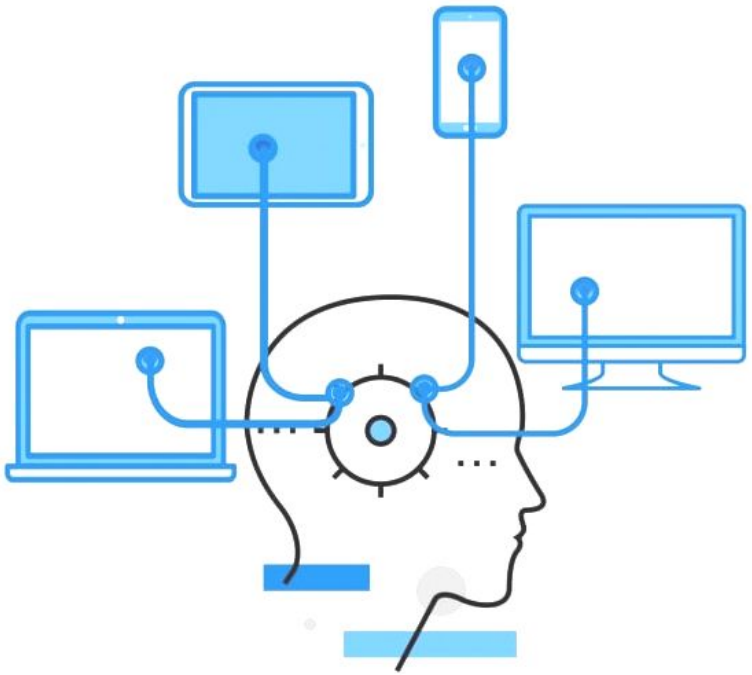


Image: pngtree

Option
IA & Programmation

Cycle 1 & 2: Bonus pour les plus avancés

Alexandre Mazel

Année 2022-2023

alexandre.zelma@gmail.com

Nombre premier 1/2

1. Développer une méthode isPrime(n) qui retourne vrai si n c'est un nombre premier (ou faux sinon).
   [tester le reste de la division de n par 1 a n-1]
2. Développer une méthode countPrime(n), utilisant isPrime, qui retourne le nombre de nombre premier entre 1 et n inclus.
3. Mesurer le temps d'exécution de countPrime(100000)
4. Pourrais on optimiser isPrime?

Nombre premier 2/2

1. Redévelopper countPrime(n) de manière plus optimal:
    - Sans utiliser isPrime
    - Réutilisant les nombres premiers précédemment trouvés dans le comptage.
2. Mesurer le temps d'exécution de countPrime(100000)

La récursion

On dit qu'une fonction est récursive si elle s'appelle elle-même.

Par exemple:

```
def func_debile(n):
    return func_debile(n*2)
```

Attention: toujours penser à la sortie de récursion.

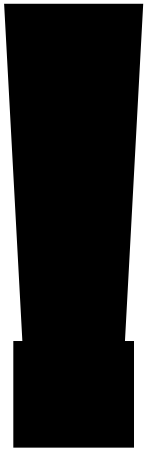
l'exemple ci-dessus est une fonction qui ne se termine jamais. Ella va donner l'impression de "planter": ne jamais rendre la main et prendre des ressources croissantes à la machine.

La récursion

Voici un bon exemple:

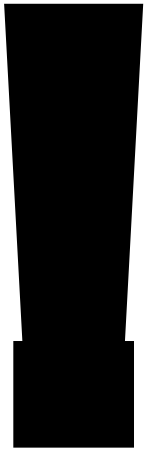
```
def pgcd(a,b):
    if b == 0:
        return 0 # ERROR
    if a%b == 0:
        return b
    return pgcd(b,a%b)
```

Factoriel 1/2



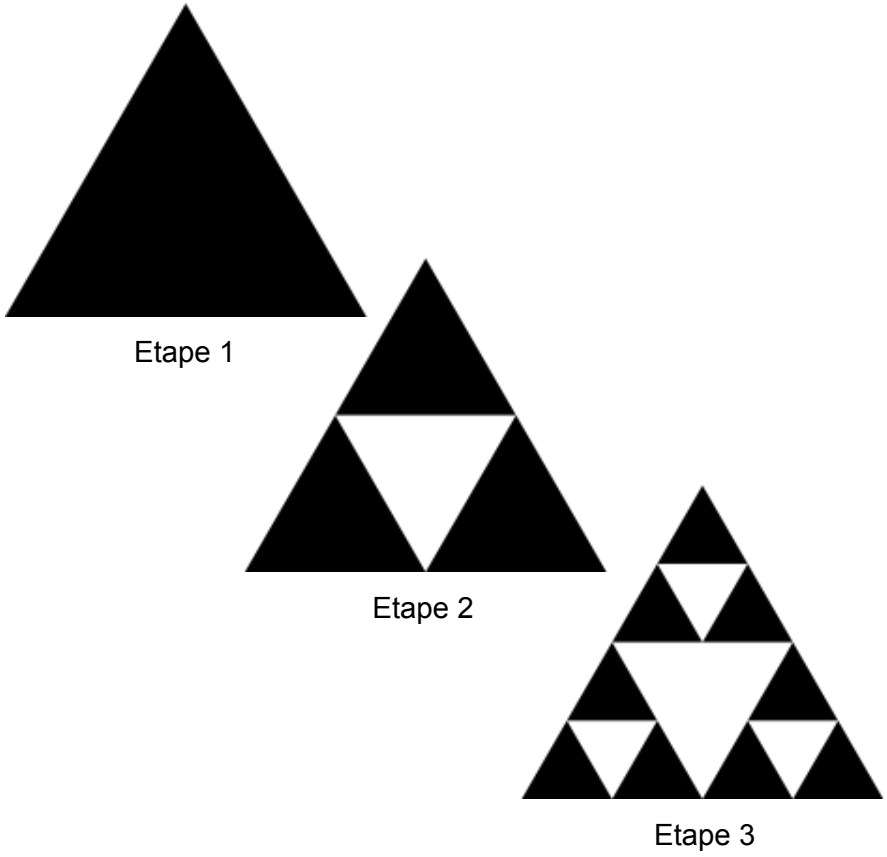
1. Développer la fonction factorial(n) qui retourne la factoriel de n.
2. Paresseusement, de manière récurrente en utilisant la propriété: n! = n*(n-1)!
3. Mesurer le temps d'exécution de factorial(10000)

Factoriel 2/2



1. Développer la fonction factorial(n) qui retourne la factoriel de n de manière itérative sans récursion.
2. Mesurer le temps d'exécution de cette nouvelle factorial(10000)

Triangle de Sierpinsky



credits: wikipedia.org

Faire un programme qui affiche un triangle de Sierpinsky.

1. Tracer un triangle noir
2. Tracer un triangle blanc ayant pour sommet le milieu de chaque arete du précédent.
3. Recommencer pour chacun des 3 nouveaux triangles.
4. Faire un rendu après chaque itération pour voir la progression.

Hint: dessiner un triangle en cv2 (untested):

```
triangle_cnt = np.array( [pt1, pt2, pt3] )

cv2.drawContours(image, [triangle_cnt], 0, (0,255,0), -1)
```