TD 2

Exercices d'application

Savoirs et compétences :
❒ *Alg – C15 : Récursivité : avantages et inconvénients.*

Exercice 1 – Fonction mystère

D'après ressources de C. Lambert. On donne la fonction suivante :

▪ Python
```
def mystere(L):
    """
    Ceci est la fonction mystère, saurez-vous
    trouver son but ?
    Entrée :
        * L( list ) :  liste  de nombres entiers
    ou réels
    Sortie  :
        * ???
    """
    n = len(L)
    if  n==0 :
        return (None)
    elif  n==1 :
        return (L[0])
    else :
        x = mystere(L[0:n-1])
        if  x<=L[-1] :
            return (x)
        else :
            return (L[-1])
```

Question 1 *Sans coder la fonction, déterminer le résultat de l'instruction* ***print(mystere([14, 20, 3, 16]))*** *? Vous pourrez représenter de façon graphique l'empilement et le dépilement de la pile d'exécution.*

Question 2 *D'après vous quel est le but de cette fonction ?*

Question 3 *Programmer la fonction et tester l'instruction précédente. Sur plusieurs exemples, vérifiez la conjecture faite à la question précédente.*

Question 4 *Question subsidiaire. – Montrer que la propriété suivante est une propriété d'invariance :*
$\mathscr{P}(k)$ *: l'algorithme retourne le plus petit élément de toute liste de taille* k*.*

R

- Il faudra montrer que l'algorithme se termine au moyen d'un variant de boucle.
- Il faudra montrer $\mathscr{P}$ par récurrence.

Exercice 2 – Palindrome...

D'après ressources de C. Lambert.

On souhaite réaliser une fonction `miroir` dont le but est de retourner le «miroir» d'une chaîne de caractères. Par exemple le résultat de `miroir("miroir")` serait *"riorim"*.

Question 1 *Programmer la fonction* `miroir_it` *permettant de répondre au problème de manière itérative.*

Question 2 *Programmer la fonction* `miroir_rec` *permettant de répondre au problème de manière récursive.*

Question 3 *Que renvoie la fonction si la chaîne de caractère est "Eh ! ça va la vache" ?*

Question 4 *Évaluer la complexité algorithmique de chacune des deux fonctions.*

Exercice 3 – Suite de Fibonacci

D'après ressources de C. Lambert. On définit la suite de Fibonacci de la façon suivante :

$$\forall n \in \mathbb{N}, \begin{cases} u_0 = 0, u_1 = 1 \\ u_{n+2} = u_n + u_{n+1} \end{cases}$$

Question 1 *Définir la fonction* `fibonacci_it` *permettant de calculer* u_n *par une méthode itérative. Évaluer la complexité algorithmique de l'algorithme.*

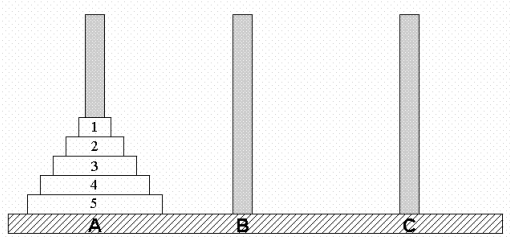
Question 2 *Définir la fonction* `fibonacci_rec` *permettant de calculer* u_n *par une méthode récursive « intuitive». Évaluer la complexité algorithmique de l'algorithme.*

Question 3 *Observer comment passer du couple* (u_n, u_{n+1}) *au couple* (u_{n+1}, u_{n+2})*. En déduire une autre*

méthode récursive pour calculer le n^e terme de la suite de Fibonacci. Évaluer la complexité algorithmique de l'algorithme.

Exercice 4 – Les tours de Hanoï

On considère le problème suivant, connu sous le nom des *tours de Hanoï.*



On dispose de n disques de diamètres tous différents, qui sont empilés sur un piquet A, par ordre croissant de diamètre, en les regardant de haut en bas. On dispose de deux autres piquets vides B et C, et le but est de déplacer tous les disques sur le piquet C, en respectant quelques règles :

- on a le droit d'utiliser tous les piquets ;
- à la fin, les disques doivent toujours être rangés par ordre croissant de diamètre, en les regardant de haut en bas ;
- on ne peut déplacer les disques que un par un ;
- on ne peut empiler un disque que sur un disque de diamètre inférieur.

Ce problème est très compliqué à résoudre de manière directe.
Il est par contre très simple à résoudre de manière récursive.

Question 1 *Résoudre le problème pour $n = 1$.*

Question 2 *On suppose que l'on sait résoudre le problème pour $n-1$ disques. Donner alors une résolution du problème pour n disques.*

Question 3 *Écrivez maintenant un programme **récursif** résolvant ce problème. Il s'agit d'écrire un programme* `hanoi` *ayant quatre arguments A, B, C et n :*

- *le premier, A, est le piquet sur lequel se trouvent les disques au départ ;*
- *le second, B, est le piquet "de transition" ;*
- *le troisième, C, est le piquet sur lequel on veut récupérer les disques à la fin ;*
- *et enfin le quatrième et dernier argument, n, est le nombre de disques.*

On lui demande également d'afficher tous les déplacements effectués sous forme de chaînes : la chaîne `'a->b'` *signifie par exemple que le disque du dessus du piquet A est déplacé sur le piquet B. On utilisera le programme d'impression des déplacements suivant :*

```
def deplace (x, y) :
    print (x + ''->'' + y + ''\n'')
```

Attention : A, B, C, x et y sont ici de type `string`.

Écrire le programme `hanoi`. *Attention : il s'agit d'un programme récursif, qui va donc être très court ! En aucun cas le programme en lui-même ne fait apparaître les opérations effectuées.*

Exercice 5 – Faisons des Bulles

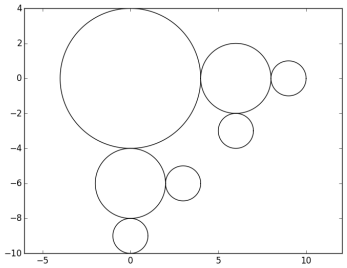
D'après les ressources de Mmes SEMBELY et VERDIER
Les fractales sont des objets mathématiques fondés sur des figures géométriques se répétant à l'infini via un processus itératif.

"Une fractale est un objet irrégulier, dont l'irrégularité est la même à toutes les échelles et en tous les points"
Adrien DOUADY - mathématicien français

Ce type de structure se retrouve également dans la nature : architecture des côtes maritimes, ramifications nerveuses, nuages, galaxies ...
Etant donné leur structure, il est très intéressant d'utiliser la récursivité pour visualiser des ensembles fractals.

Question 1 *Ecrire une fonction* `cercle` *d'arguments* `x`, `y` *et* `r` *qui trace le cercle de centre le point $A(x, y)$ et de rayon* `r` *(supposé strictement positif).*

Question 2 *Ecrire une fonction récursive* `bubble` *d'arguments* `x`, `y`, `r` *et* `n`. *Elle effectuera la construction pour un nombre* `n` *d'étapes en ayant pour figure de base le cercle de centre $A(x, y)$ et de rayon* `r` *(supposé strictement positif). À chaque étape, le rayon du cercle est divisé par 2.*



Question 3 *Ecrire une fonction récursive* `bubbleComplet` *d'arguments* `x`, `y`, `r`, `n` *et une chaîne de caractères* `position`. *Elle effectuera la construction ci-dessous pour un nombre* `n` *d'étapes en ayant pour figure de base le cercle de centre $A(x, y)$ et de rayon* `r` *(supposé strictement positif).*