I Fonctions récursives

Rappel du principe

Deviner la valeur de retour des lignes suivantes (a est un nombre et n un entier naturel) :

```
def f(a, n):
    if n == 0:
        return 1
    return a * f(a, n - 1)

f(2, 3)
```

Implémentation

Pour $n, k \in \mathbb{N}$ avec $n, k > 0$ et $n \geqslant k$, on a $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$. De plus, $\binom{n}{0} = 1$ et dans le cas $k > n$, $\binom{n}{k} = 0$ Compléter la fonction suivante

```
def coeff(n, k):
    if    :
        return 0
    elif   :
        return 1
    else:
        return coeff(  ) + coeff(  )
```

II Versions récursives

Vers le récursif

Que retourne la fonction suivante quand on l'appelle avec une liste `L` ?

```
def machin(L):
    res = []
    for i in range(len(L)):
        res = [L[i]] + res
    return res
```

Ecrire une fonction récursive qui effectue le même travail.
Rappels :

- Il s'agit de construire le résultat pour une liste de longueur n connaissant le résultat pour une liste de longueur $n-1$. On pourra utiliser la syntaxe `L[a:b]` qui retourne la liste des éléments de L dont les indices sont exactement `range(a, b)` c'est à dire de a jusqu'à $b-1$
- Ne pas oublier le cas initial (ou cas darrêt), c'est à dire le cas que l'on sait traiter, pour une valeur simple de la longueur de `L`

Depuis le récursif

On considère maintenant la fonction qui prend un entier naturel n comme paramètre

```
def truc(n):
    if n <= 9:
        return [n]
    L = truc(n // 10)
    L.append(n % 10)
    return L
```

1. Simuler sur le papier un appel à truc(513). Combien d'appels récursifs ?
2. Que calcule truc(n) ?
3. Ecrire une version itérative, avec un nom raisonnable.

III Plusieurs appels

Un petit dessin

Le but de cet exercice est d'obtenir le graphique représenté sur la figure 1

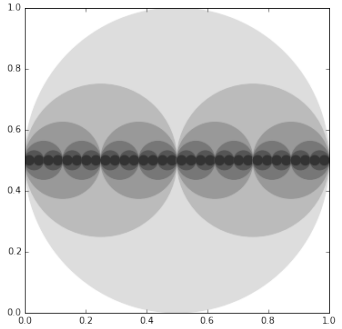

FIGURE 1 – Cercles récursifs

Pour cela vous disposez dans le fichier **cercles.py** de plusieurs fonctions de dessin déjà prête. Votre tâche est de remplir la fonction cercles_rec. Vous pourrez appeler cercles(5) pour tester votre fonction.

REMARQUE : la fonction récursive est en fait une fonction “auxiliaire” qui prend plus d’arguments que la fonction principale. cette astuce est très classique pour calculer récursivement avec plus de données.

Récursion mutuelle

Que calculent ces deux fonctions (d’argument entier naturel) ?

```
def a(n):
    if n == 0:
        return True
    else:
        return b(n - 1)
def b(n):
    if n == 0:
        return False
    else:
        return a(n - 1)
```

IV Retour aux classiques

Ecrire des fonctions récursives pour :

- calculer le maximum d’une liste de nombre. Deux indications : 1) *max* est un mot réservé, utilisez *maxi* comme nom de fonction, 2) ici on pourra utiliser une fonction récursive auxiliaires.
- calculer le nombre d’occurence de l’élément x dans la liste L.
- calculer la liste des k-uplets de $[\![1, n]\!]$.
- Donner la liste des sous-ensemble de $[\![1, n]\!]$, où chaque sous-ensemble est une liste.