Fonctions récursives — TP7

Exercice 1 — Prudemment récursif

Dans chacun des cas suivants, écrire une fonction récursive prenant en argument $n \in \mathbb{N}$ et renvoyant u_n. On vérifiera la valeur de u_{10} donnée.

Q1 $u_0 = 2$ et $\forall n \in \mathbb{N},\ u_{n+1} = \dfrac{1}{u_n + 2}$. On a $u_{10} \simeq 0,41421$.

Q2 $u_1 = 3$ et $\forall n \in \mathbb{N}^*,\ u_{n+1} = \dfrac{u_n - 2n}{3}$. On a $u_{10} \simeq -8,4998$.

Q3 $u_1 = 2$ et $\forall n \in \mathbb{N}^*,\ u_n = u_{n-1} + \sqrt{u_{n-1}}$. On a $u_{10} \simeq 31,051$.

> **Remarque**
> Combien d'appels récursifs à la fonction avez-vous demandés dans votre code précédent ? Reprenez votre fonction si vous en avez trop. Un code faisant trop d'appels récursifs sera pénalisé au DS.

Q4 $u_0 = u_1 = 1$ et $\forall n \in \mathbb{N},\ u_{n+2} = u_n + u_{n+1}$. On a $u_{10} = 89$.

> **Remarque**
> Combien d'appels récursifs à la fonction avez-vous demandés dans votre code précédent ? Est-ce judicieux ? Écrire une fonction non récursive plus efficace.

Q5 $u_n = \displaystyle\sum_{k=2}^{n} \frac{\sqrt{k}}{1+k}$. On a $u_{10} \simeq 3,2594$.

Q6 $u_n = \displaystyle\prod_{k=1}^{n+1} (\sqrt{k} + 1)$. On a $u_{10} \simeq 443129$.

Exercice 2 — Fortement récursif

Q1 Soit (u_n) la suite définie par $u_0 = 2$ et : $\forall n \in \mathbb{N}^*,\ u_n = \begin{cases} \dfrac{u_{n-1}+1}{2u_{n-1}+3} & \text{si } n \text{ est impair} \\ u_{\frac{n}{2}} + u_{\frac{n}{2}}^2 - n & \text{si } n \text{ est pair.} \end{cases}$

Écrire une fonction récursive prenant en argument $n \in \mathbb{N}$ et renvoyant u_n. On vérifiera que $u_{10} \simeq -8.9789$ et qu'on n'a pas fait trop d'appels récursifs.

Q2 Soit (v_n) la suite définie par : $v_0 = 0$ et : $\forall n \geq 0,\ v_{n+1} = \begin{cases} v_{\frac{n}{2}} + 2n + 1 & \text{si } n+1 \text{ est impair} \\ 4v_{\frac{n+1}{2}} & \text{si } n+1 \text{ est pair.} \end{cases}$

Écrire une fonction récursive prenant en argument $n \in \mathbb{N}$ et renvoyant v_n. On vérifiera que $v_{10} = 52$ et qu'on n'a pas fait trop d'appels récursifs.

Exercice 3 — Valuation p-adique

Soient $p, n \in \mathbb{N}^*$, on appelle *valuation p-adique de n* et on note $v_p(n)$ le nombre de fois que n est divisible par p. En d'autres termes, $v_p(n) = k$ lorsque p^k divise n mais que p^{k+1} ne divise pas n, ou encore lorsque $n = p^k m$ avec m non divisible par p.
On a par exemple $v_2(40) = 3$ car $40 = 2^3 \times 5$; $v_5(75) = 2$ car $75 = 5^2 \times 3$. Le but de cet exercice est d'écrire une fonction `valuation` prenant en argument p et n et renvoyant $v_p(n)$.

Q1 Combien vaut $v_3(18)$? $v_3(12)$? $v_3(11)$?

Q2 À quelle condition sur p et n l'entier $v_p(n)$ est-il nul ? Écrire cette condition en Python.

Q3 Si n est divisible par p, quelle relation y a-t-il entre $v_p(n)$ et $v_p(\frac{n}{p})$?

Q4 Écrire la fonction `valuation`.

Exercice 4 — Triangle de Pascal

Q1 Rappeler la formule du triangle de Pascal.

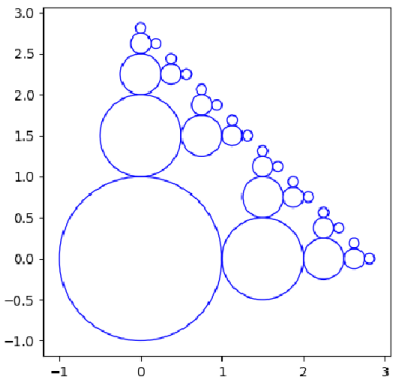
Q2 Expliquer pourquoi cette formule est encore valable lorsqu'elle fait intervenir des coefficients binomiaux de la forme $\binom{b}{a}$ avec $a \notin [\![0, b]\!]$.

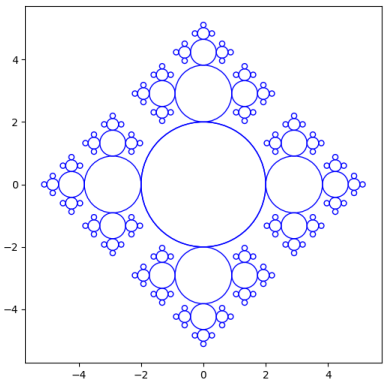
Q3 En déduire une fonction récursive prenant en argument deux entiers k et n et renvoyant le coefficient binomial $\binom{n}{k}$.

Q4 Écrire un programme permettant d'afficher les N premières lignes du triangle de Pascal. On utilisera la fonction `print`. Pour pouvoir afficher plusieurs coefficients binomiaux sur la même ligne, on utilisera la syntaxe : `print(x, end="")` (où `x` est la quantité à afficher). Pour changer de ligne, on pourra utiliser la fonction `print` sans argument.

Exercice 5 — Dessine-moi un chou

Dans la nature, certains végétaux présentent une structure *fractale* c'est-à-dire se répétant à toutes les échelles. C'est le cas par exemple du chou romanesco, première image ci-dessous. Le but de cet exercice est de dessiner des figures fractales pouvant faire penser (avec un peu d'imagination) à des modèles simplifiés de végétaux. On commencera par dessiner la figure du milieu, puis enfin la figure de droite.





Pour dessiner ces figures, commencez par recopier (ou télécharger depuis cahier de prépa, fichier `TP6-exo5-code.py`) le code suivant :

```
1  import matplotlib.pyplot as plt
2  axe = plt.gca()
3
4  def cercle(x,y,r):
5      c = plt.Circle((x, y), r, color='b',fill=False)
6      axe.add_patch(c)
7
8  def montre():
9      plt.axis('scaled')
10     plt.show()
```

Dans ce code, la fonction `cercle` permet de dessiner un cercle dont on précise les coordonnées du centre (arguments `x` et `y`) et le rayon (argument `r`). La fonction `montre` quant à elle ne prend rien en argument et doit seulement être utilisée en fin de code (écrire `montre()`) pour que Python affiche le dessin réalisé.

Q1 On souhaite produire le dessin du milieu. Sur cette figure, le cercle le plus grand est de centre $(0, 0)$ et de rayon 1. Puis s'articulent autour de lui des cercles plus petits dont les centres sont toujours alignés, soit horizontalement soit verticalement, avec le centre du cercle auquel ils sont tangents. Il y a un facteur 2 entre les rayons de deux cercles tangents.
Écrire une fonction récursive `figure1` prenant en argument un entier n, des coordonnées x, y et un rayon r, et réalisant cette figure. La figure montrée ici correspond au cas $n = 4, x = y = 0$ et $r = 1$.

Q2 On souhaite produire le dessin de droite. Sur cette figure, le cercle le plus grand est de centre $(0, 0)$ et de rayon 1 et il y a un facteur $2,2$ entre les rayons de deux cercles tangents.
Écrire un programme permettant de dessiner cette figure. On pourra utiliser une fonction récursive `figure2` prenant en argument un entier n, des coordonnées x, y, un rayon r et éventuellement un autre argument.