TD PYTHON 07

Autour des séries entières

❑ **Méthode 0.1.— Rappel : Comment manipuler des entiers avec Python**

- `range(n)` donne les entiers de 0 à $n-1$.
- `range(a,b)` donne les entiers compris entre a (compris) et b (non compris).
- `range(a,b,p)` donne les entiers compris entre a et b avec un pas de p.
- `range(n,i,-1)` pour $i < n$, donne les entiers de $i+1$ à n dans l'ordre décroissant.
- `a // b` donne le quotient dans la division euclidienne de a par b.
  Ainsi `a // n` donne $\lfloor \frac{a}{n} \rfloor$.
- `a % b` donne le reste dans la division euclidienne de a par b.
  Ainsi `a % b == 0` signifie que a est un multiple de b.
- Pour écrire $S = \sum_{k=1}^{n} a_k$, on peut **utiliser une boucle**.
  Ou alors on utilise la **fonction prédéfinie** `sum`. Dans ce cas, on tape :

```
S = sum(k**2 for k in range(1,6));
```

  Rappelons au passage que si les a_k sont directement donnés sous forme d'une liste L, la commande `sum(L)` donne S.
- $n!$ s'écrit `factorial(n)` et préalablement on tape `from math import factorial`
- Pour écrire $P = \prod_{k=1}^{n} a_k$, on peut utiliser encore une boucle comparable à celle de la somme mais on remplace `S +=` par `P *=`

Exercice 01 : étude d'une somme de série entière

On pose :

$$f : x \mapsto \sum_{n=0}^{+\infty} x^{n^2}$$

et $(a_n)_{n\in\mathbb{N}}$ la suite définie par $a_n = 1$ s'il existe p entier tel que $n = p^2$ et $a_n = 0$ sinon.

1. Écrire une liste d'instruction pour créer la suite (a_n) jusqu'à 1000 inclus.

2. Déterminer le domaine de définition de f.

3. Écrire une fonction `Somme` qui prend en argument x et n et renvoie $\sum_{k=0}^{n} a_k x^k$.

Tracer cette fonction pour $x \in [-1, 1]$. Que peut-on conjecturer ?

4. Écrire une fonction `NewSomme` qui prend en argument x et n et renvoie $\sum_{k=0}^{n} (-1)^k a_k x^k$.

Quel semble être sa limite en $x = 1$ quand n tend vers $+\infty$?

Indications : *On pourra aller voir préalablement la* **méthode 0.1** *et la méthode du Who's Who pour tracer des courbes.*
En particulier on tape `import math as mt` *et* `mt.floor` *donne la partie entière d'un réel.*
De même, on tapera `import numpy as np` *et* `np.sqrt` *fournit la racine carrée d'un nombre.*

Cours : produit de Cauchy de deux séries entières

Dans tout le paragraphe, R_a (respectivement R_b) est le rayon de convergence de la série entière $\sum_{n\geqslant 0} a_n z^n$ (respectivement $\sum_{n\geqslant 0} b_n z^n$). Leurs sommes respectives sont f et g.

box 0.90 setgray fill 1 setlinewidth

Proposition 0.1.— Produit —. Alors fg est définie par la somme de la série entière $\sum_{n\geqslant 0} c_n z^n$, où

$$c_n = \sum_{k=0}^{n} a_k b_{n-k}$$

(**produit de Cauchy** des suites (a_n) et (b_n)). Cette série entière produit a un rayon de convergence $R \geqslant \inf(R_a, R_b)$.

Exercice 02 : À propos d'une série de Taylor

Soit $f : x \mapsto \dfrac{1}{2-e^x}$ et on note pour tout $n \in \mathbb{N}$, $a(n) = \dfrac{f^{(n)}(0)}{n!}$.

1. Montrer, en utilisant $x \mapsto (2-e^x)f(x)$, que pour tout $n \in \mathbb{N}$, $a(n) = \sum_{k=1}^{n} \dfrac{a(n-k)}{k!}$.

2. Écrire avec Python une procédure pour calculer $a(n)$ pour $n \in [\![0, 10]\!]$.

3. Tracer (pour $n \in [\![0, 10]\!]$) les courbes :

$$n \mapsto (n,\, a(n))\,,\ x \mapsto \left(x,\, \frac{1}{(\ln 2)^x}\right) \text{ et } x \mapsto \left(x,\, \frac{1}{2(\ln 2)^x}\right).$$

En déduire le rayon de convergence de $\sum_{n\geqslant 0} a(n)x^n$. On note S sa somme.

4. Tracer l'approximation de la courbe S grâce aux sommes partielles pour $n \in [\![0, 6]\!]$.
Tracer la courbe de f sur $[0; 0.5]$. Que peut-on en déduire ?

***Indications :* 1.** *On applique la* **proposition 0.1** *aux fonctions $x \mapsto (2-e^x)$ et $x \mapsto f(x)$.*
2. *On tapera* `import math as mt` *pour introduire* `mt.factorial`.
Puis on pourra aller voir la **méthode 0.1**.
3. *Dans cette question et la suivante, on ira revoir la méthode du Whos's Who pour tracer une courbe ou pour tracer ensemble plusieurs courbes. Dans la question* **3**, *$n \mapsto (n, a(n))$ est une fonction à variable discrète et le graphe sera composé de 11 points donc on prendra* `X = [ k for k in range(0,11)]` *pour intervalle des abscisses et par contre les fonctions $x \mapsto \left(x, \dfrac{1}{(\ln 2)^x}\right)$ et $x \mapsto \left(x, \dfrac{1}{2(\ln 2)^x}\right)$ sont continues et on se placera par exemple sur* `XX = np.linspace(0,10,500)` *avec* `np` *l'alias de* `numpy` *comme d'hab ! Enfin, pour appliquer à une liste, on utilisera* `np.e**x` *plutôt que* `mt.exp(x)` *pour taper e^x.*