Introduction à Python

Bonus 01

Dans cette feuille, seuls les « pour » et les « si » sont autorisés (pas de boucle « tant que »).

Exercice 1

Écrire une fonction `somme_carres(n)` qui prend en entrée un entier `n` et renvoie la somme des carrés des entiers compris entre 1 et `n` inclus.
En appelant la fonction précédente, écrire une fonction `prod_somme_carres(n)` qui renvoie le produit des sommes des carrés de 1 à `k` pour `k` compris entre 1 et `n`. Par exemple, pour `n` = 4, la fonction renverra 2100 car :

$$(1^2) \times (1^2 + 2^2) \times (1^2 + 2^2 + 3^2) \times (1^2 + 2^2 + 3^2 + 4^2) = 2100$$

Exercice 2

1. Écrire une fonction `somme_inverses(n)` qui renvoie la somme des inverses des entiers compris entre 1 et `n` sous forme d'un flottant.
2. La somme faite précédemment étant une somme de rationnels, le réel calculé précédemment est un rationnel : on va chercher à déterminer la fraction en question.
   a) Supposons que $\sum_{k=1}^{n-1} \frac{1}{k} = \frac{a}{b}$. Montrer que $\sum_{k=1}^{n} \frac{1}{k} = \frac{an+b}{nb}$.
   b) Écrire une fonction `somme_inverses_rationnel(n)` qui renvoie la somme des inverses des entiers compris entre 1 et `n` sous forme (a, b) avec a et b entiers, représentant la fraction $\frac{a}{b}$ (on n'exige pas que la fraction soit irréductible).

Exercice 3

1. Écrire une fonction `somme_inverses_factorielles(n)` qui renvoie la somme des inverses des factorielles de 1 à `n` compris. Par exemple :

$$\frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} \simeq 2,666666$$

2. De nouveau, le résultat de la fonction précédente étant une somme de rationnels, il est possible de l'écrire comme une fraction. Écrire une fonction `somme_invfact_rationnel(n)` qui renvoie le résultat précédent sous la forme (a, b) avec a et b entiers tels que $\frac{a}{b}$ soit le résultat attendu.
3. Pour des raisons que vous verrez plus tard en mathématiques :

$$\frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!} \xrightarrow[n\to\infty]{} e$$

   La constante d'Euler e est connue de la bibliothèque `math`. Écrire une fonction `taux_erreur(n)` qui renvoie le pourcentage d'erreur de `somme_inverses_factorielles(n)` par rapport au calcul de e. Par exemple, pour `n` = 3, le taux d'erreur est de 1.9% : 2,666666 est proche de la vraie valeur de e.

Exercice 4

1. Écrire une fonction `nb_sol_reelles(a,b,c)` qui renvoie le nombre de solutions réelles de l'équation $ax^2 + bx + c = 0$.

2. Écrire une fonction `solutions_reelles(a,b,c)` qui renvoie toutes les solutions réelles de l'équation $ax^2 + bx + c = 0$. Par exemple :
   - pour `(1,8/3,-1)`, la fonction renverra `(0,33333333333,-3)` ;
   - pour `(1,-2,1)`, la fonction renverra `(1)` (et pas `(1,1)`);
   - pour `(1,0,1)`, la fonction renverra `()`.

Exercice 5

On cherche à écrire des fonctions qui permettent de tester la primalité d'un entier.

1. Pour tester si `n` est premier, on parcourt tous les entiers entre 2 et `n-1` : si l'un d'eux divise `n`, on renvoie `False`. Si à la fin de la boucle, on n'a rien renvoyé, c'est que notre nombre est premier : on renvoie `True`. Implémenter cet algorithme dans une fonction `est_premier(n)`.
2. On peut améliorer l'algorithme précédent : plutôt que d'aller de 2 à `n-1`, on peut aller de 2 à $\lfloor\sqrt{n}\rfloor$.
   a) Pourquoi est-ce suffisant ?
   b) Pourquoi est-ce mieux que d'aller de 2 à `n-1` ?
   c) Implémenter le nouvel algorithme dans une fonction `est_premier2(n)`.

Exercice 6

La suite de Fibonacci est définie par $f_0 = 0$, $f_1 = 1$ et $f_{n+2} = f_{n+1} + f_n$ pour tout $n \in \mathbb{N}$.

1. Écrire une fonction `fibo(n)` qui permet de calculer le `n`-ème terme de la suite de Fibonacci.
2. Écrire une fonction `somme_fibo(n)` qui permet de calculer la somme des termes de la suite de Fibonacci jusqu'au `n`-ème inclus.
3. Calculer `somme_fibo(100 000)`.