Université d'Angers
Département de Mathématiques

DL1 MI et ME

Programmation sous python – 2024-2025

Feuille d'exercices n° 2

Exercice 1 (utilisation de tests logiques)**.** On utilise les structures `if ... elif ... else`.

1) Écrire une fonction `solEquationD2R` prenant en arguments deux réels b et c et rendant la liste des **solutions réelles** de l'équation $x^2+bx+c=0$. Cette liste de sortie sera donc de longueur 0, 1 ou 2, suivant les cas[1]. (*Indication :* La comparaison avec 0 du discriminant devrait être de la forme $|\Delta| < |b|^2 \cdot 10^{-14}$.)

2) Faire des tests avec les réels $a=1$, $b=\frac{1}{10^k}$ et $c=\frac{1}{\gamma \cdot 10^{2k}}$, avec $k=0,\dots,9$ et $\gamma=4$ ou 5. Expliquer les résultats et comparer les avec ceux obtenus en utilisant une condition de la forme $|\Delta| < 10^{-14}$.

Exercice 2 (utilisation des boucles)**.**

1) Écrire une fonction `euler2` prenant en argument un entier n et rendant $\sum_{j=1}^{n} \frac{1}{j^2}$. On introduira une variable S qui vaut initialement 0 et à laquelle on ajoutera successivement les valeurs de $1/j^2$ en utilisant une boucle `for` et le générateur `range`.

2) Écrire deux fonctions prenant en argument un entier n, et rendant respectivement

$$\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{1}{j^2+k^3} \qquad \text{et} \qquad \sum_{k=1}^{n}\sum_{j=1}^{n} \frac{1}{j^2+k^3}.$$

En quoi diffèrent leurs écritures ? Comparez les résultats obtenus par ces deux fonctions et expliquez.

Exercice 3 (construction de listes)**.** Pour un entier n, on veut construire, par différentes méthodes, la liste des j^2, pour j variant de 0 à n.

1) Partir d'une liste vide et lui adjoindre successivement tous les j^2 (utiliser `append`).

2) Utiliser une *compréhension* de liste (similaire à l'écriture mathématique $\{j^2 \mid j=0,\dots,n\}$).

3) Comparer les vitesses d'exécution de ces différentes méthodes. On pourra utiliser `%timeit` dans l'interpréteur `ipython` de Spyder.

Exercice 4 (autopsie d'un programme ; boucle *tant que*)**.** Étudiez ce que fait la fonction suivante :

```python
def F(a, b):
  while b != 0:
    a, b = b, a%b
  return a

```

Testez ce que deviennent les variables en introduisant des commandes `print`.

Exercice 5 (suite de l'exercice 2)**.** On rappelle que $\sum_{j\geq 1} \frac{1}{j^2} = \frac{\pi^2}{6}$.

1) Écrire une fonction `eulerDecimalesExactes` qui prend en argument un entier strictement positif d et qui retourne S et N, où N est le **plus petit entier** pour lequel `S=euler2(N)` a les d premières décimales égales à celles de $\frac{\pi^2}{6}$.

2) Même problème pour `S=euler3(N)` en remplaçant $\frac{\pi^2}{6}$ par la valeur de la somme $\sum_{j\geq 1} \frac{1}{j^3}$. Faudrait-il repenser le programme précédent ? En ne connaissant pas cette somme, il faudrait

[1] python représente les nombres réels en virgule flottante sur 64 de bits. La commande `numpy.finfo(float)` permet de voir les informations principales concernant l'utilisation des nombres réels par python.

trouver une borne pour le reste

$$r_N = \sum_{j=1}^{+\infty} \frac{1}{j^3} - \sum_{j=1}^{N} \frac{1}{j^3} = \sum_{j=N+1}^{+\infty} \frac{1}{j^3}.$$

Par exemple, on pourrait obtenir $r_N < \frac{1}{(N+1)^2}$. Si $d = 4$, alors en prenant $N = 2 \cdot 10^2$, `euler3(N)` a quatre décimales exactes.

Exercice 6 (diviseurs d'un nombre entier)**.** Dans cet exercice, il faudra, au moins dans un deuxième temps, veiller à optimiser le programme afin d'éviter de faire des calculs inutiles.

1) Écrire une fonction `python` `sommeDiviseurs` qui prend en argument un entier n et rend le couple s et L, où L est la liste des diviseurs stricts de n et s la somme des diviseurs stricts. L'entier d est un diviseur strict de n si $1 \leq d < n$ et $d|n$. On pourra tester si $d|n$ avec `n%d == 0`.

2) On dit qu'un nombre est parfait s'il est égal à la somme de ses diviseurs stricts ; ainsi $6 = 1 + 2 + 3$ est un nombre parfait. Écrire une fonction qui prend en argument un entier n et rend la liste des nombres parfaits inférieurs ou égal à n.

3) Deux nombres a et b sont dit amicaux si l'un est égal à la somme des diviseurs stricts de l'autre et réciproquement. Déterminer tous les couples de nombres amicaux inférieurs à 1000.

Exercice 7. Écrire une fonction `python` qui prend en argument une liste L et retourne la liste U obtenue en enlevant tous les doublons de L. On veut que l'ordre des éléments de U soit celui des premières apparitions de ces éléments dans la liste L. Par exemple $[2, 4, 1, 1, 4, 5] \mapsto [2, 4, 1, 5]$.

Exercice 8 (nombres premiers)**.** On rappelle qu'un entier supérieur à 1 est dit *premier* si ses seuls diviseurs positifs sont 1 et lui même.

1) Écrire une fonction python `estPremier` qui prend en argument un entier n et rend `True` ou `False` suivant que n est premier ou pas.

2) Écrire une fonction qui prend en argument un entier n et rend la liste des nombres premiers inférieurs ou égal à n. Écrire aussi une fonction `cribleDErathostene` qui retourne la liste des nombres premiers en la construisant avec l'algorithme d'Ératosthène. Comparer les temps exécutions des deux fonctions.

3) Deux nombres premiers impairs sont dits *jumeaux* si leur différence est égale à 2 ; ainsi $(3, 5)$, $(5, 7)$, $(11, 13)$ sont des couples de nombres premiers jumeaux. Écrire une fonction qui prend en argument un entier n et rend la liste des couples de nombres premiers jumeaux inférieurs à n.

Exercice 9. Une partition de $n \in \mathbb{N}$ est une écriture de la forme $n = a_0 + a_1 + \cdots + a_k$ avec $a_j \geq a_{j+1} \geq 1$. On dit que deux partitions p et p' vérifient $p \prec p'$ si $k \leq k'$ et $a_j \leq a'_j$ pour tout $0 \leq j \leq k$. Dans ce qui suit, on code une partition par la liste décroissante $p = [a_0, \ldots, a_k]$. Écrire une fonction qui prend en argument une partition p de n et qui retourne la liste de toutes les partitions p' de $n + 1$, avec $p \prec p'$.