Exercices 1 à 10

Exercice 1 (correction de code). Chacune des expressions suivantes comporte une erreur. Expliquer pourquoi, et proposer une version corrigée de l'expression.

```
1  z = (1+2j)**2; real(z)
2  x = (1); len(x)
3  x = (1,)+(2,); x.append(3)
4  L = (1,2,3); del L[0]
5  L = [10,3,20]; sort(L)
6  A = frozenset({1,2,3}); A.add(4)
7  E = {1,{2,3}}
8  D = dict( 'a'=1, 'b'=2, 'c'=1 )
9  D = dict( 'a':1, 'b':2, 'c':1 )
10 D = { 'premier':23, 'deuxième':41 }; D[0]
11 D = dict(); D[{1,4}] = -.27
```

Exercice 2 (racine carrée complexe à coefficients entiers).

Dans cet exercice, on se propose de construire une fonction `racine_entiere(z)`, qui prend en entrée un nombre complexe z, et renvoie un nombre complexe w de type $w = a + bi$ avec a, b entiers tels que $w^2 = z$ si un tel nombre existe, et `None` sinon.

(1) Donner une implémentation Python de cette fonction à partir de l'algorithme suivant:

> - calculer dans w une racine carré de z (utiliser la fonction puissance `**`)
> - arrondir les parties réelles et imaginaires de w à l'entier le plus proche
> - renvoyer w ou `None`, selon que $w^2 = z$ ou non.

(2) **(TP)** Implémenter la fonction précédente, et construire un dictionnaire qui associe, à chaque nombre complexe de type $x + iy$ avec $x, y \in \{1, 2, \ldots, 9\}$ qui admet une racine carrée complexe à composantes entières, l'ensemble de ses deux racines carrées.

(3) **(TP)** Écrire une boucle qui affiche le contenu de ce dictionnaire de manière la plus lisible possible.

Exercice 3 (p-valuation et nombre de zéros terminaux de $n!$).

Si p est un nombre premier (c'est-à-dire un nombre entier $\geqslant 2$ divisible seulement par 1 et lui-même), on appelle p-valuation d'un entier $N \in \mathbb{N}^*$ le plus grand entier $k \in \mathbb{N}$ tel que p^k divise N, et l'on note $v_p(N)$ ce nombre. On a par exemple $v_5(1500) = 3$ car $1500 = 2^2 \times 3 \times 5^3$.

(1) Calculer $v_2(100)$, $v_5(10!)$, puis $v_5(25!)$ (avec la notation usuelle pour la factorielle, $n! = 1 \times 2 \times \ldots \times n$).

(2) Soit p le nombre de zéros par lesquels termine l'écriture décimale de N.
Expliquer pourquoi $p = \min(v_2(N), v_5(N))$.

(3) Écrire en pseudo-code un algorithme permettant de calculer $v_p(n!)$ pour un entier n donné.

(4) **(TP)** Implémenter l'algorithme précédent en Python sous la forme d'une fonction `valfact(p,n)` qui retourne $v_p(n!)$.

(5) **(TP)** En déduire le nombre de zéros terminant l'écriture décimale de $N = 200000!$, et vérifier le résultat en analysant par une expression Python la chaîne de caractère obtenue par `str(N)`.

Exercice 4 (équation du second degré).

Écrire une fonction Python `racines_complexes(a,b,c)`, qui renvoie les racines (réelles ou complexes) du trinôme $aX^2 + bX + c$ sous la forme d'un ensemble de type `set`, les arguments `a,b,c` étant des nombres complexes quelconques.

Exercice 5 (écriture binaire d'un `float`).

En Python, les nombres de type `float` sont codés de telle sorte que leur écriture en base 2 est finie. Autrement dit, un tel nombre x (supposé positif pour simplifier) peut s'écrire en base 2 sous la forme

$$x = a_p \dots a_2 a_1 a_0, b_1 b_2 \dots b_q$$

où les $(a_k)_{0 \leqslant k \leqslant p}$ et les $(b_k)_{1 \leqslant k \leqslant q}$ sont tous des chiffres binaires, donc éléments de $\{0, 1\}$. Cette écriture correspond à l'égalité

$$x = \sum_{k=0}^{p} a_k 2^k + \sum_{k=1}^{q} b_k 2^{-k}.$$

Dans toute la suite, on supposera que $x \geqslant 1$, de sorte que les (a_k) ne sont pas tous nuls.

(1) Donner l'écriture en base 2 de $2x$, puis de $2^q x$

(2) Si s est une chaîne de caractères de taille n qui code $2^q x$ en base 2, comment constuit-on à partir de s la chaîne de caractères qui code x en base 2 ?

(3) Expliquer ce que retourne le pseudo-code suivant :

```
entrée : x
a ← 0
tant que x n'est pas entier:
        x ← 2x
        a ← a + 1
retourner a
```

(4) Proposer un algorithme qui à partir d'un nombre $x \geqslant 1$ de type `float`, retourne dans une chaîne de caractères son écriture binaire. On supposera que l'on dispose déjà d'une fonction qui calcule l'écriture binaire d'un entier (question: quel est le nom de cette fonction en Python ?)

(5) **(TP)** Implémenter l'algorithme de la question précédente en Python, puis le tester sur les nombres suivants: 1.1, π, e, 4/3

(6) **(*)** Déterminer expérimentalement le nombre maximal de chiffres nécessaires à l'écriture binaire d'un nombre x de type `float` tel que $1 \leqslant x < 2$.

(7) **(*)** Généraliser le programme de la question 5 à des nombres quelconques de type `float`, sans la condition $x \geqslant 1$.

Exercice 6 (anagrammes).

L'objectif de cet exercice est de déterminer les mots de la langue française qui possèdent le plus d'anagrames (sans tenir compte des accents). On rappelle que deux mots sont anagrammes l'un de l'autre s'ils se déduisent l'un de l'autre par simple permutation des lettres. Par exemple, les mots 'maree' et 'amere' sont anagrammes l'un de l'autre.

(1) Écrire un programme Python, qui à partir d'une liste `M` de mots, construit un dictionnaire `D` dont les clés sont des séquences (`tuple`) de lettres triées dans l'ordre alphabétique, et tel que `D[T]` est le nombre de mots de la liste `M` dont la séquence triée des lettres produit exactement `T`. On pourra construire `D` incrémentalement en parcourant la liste `M` et en calculant le `tuple` `T` associé à chaque mot parcouru.

(2) **(TP)** Déterminer la valeur maximale `m` présente dans `D` quand `M` est la liste obtenue par

```
M = [mot[:-1] for mot in open('mots.txt')]
```

où le fichier `mots.txt` est disponible sur la page Moodle du cours.

(3) **(TP)** Afficher, pour toutes les clés `T` de valeur `m`, les `m` mots de `M` qui sont obtenus à partir de `T`.

Exercice 7 (racine carrée entière).

(1) Proposer une fonction Python `intsqrt(n)`, qui prend en entrée un entier `n` $\geqslant 0$ et renvoie la partie entière de $\sqrt{n}$, calculée à l'aide des fonction `int()` et de l'opérateur puissance `**`.

(2) **(TP)** Vérifier (en calculant le carré du résultat) que la fonction précédente fonctionne correctement pour $n = 90$ et $n = 3^{20}$.

(3) **(TP)** Tester de même la valeur retournée par `intsqrt(n)` pour $n = 3^{1000}$. Pouvez-vous expliquer ce que vous observez ? Expliquer de même le phénomène observé pour $n = 10^{1000}$.

(4) **(TP) (*)** Proposer une nouvelle implémentation de la fonction `intsqrt(n)`, qui produit un résultat correct pour tout entier `n`$\geqslant 0$, en procédant par dichotomie.

Exercice 8 (réalisation d'un ensemble de lettres).

Soit E un sous-ensemble des 26 lettres de l'alphabet. Dans cet exercice, on dira qu'un mot m de la langue française est une *réalisation* de E si :

- le mot m n'est constitué que de lettres éléments de E, et
- chaque lettre de E est présente au moins une fois dans m.

Par exemple, le mot `'miroir'` est une réalisation de l'ensemble `E={'m','i','r','o'}`, mais pas de l'ensemble `F={'m','i','r','o','s'}`.

(1) Écrire un programme Python, qui à partir d'une liste `M` de mots, construit un dictionnaire `D` dont les clés sont des ensembles de lettres, et tel que `D[E]` est le nombre de mots de la liste `M` qui sont des réalisations de `E` (on se limitera bien sûr aux clés `E` qui admettent au moins une réalisation).

(2) **(TP)** Déterminer la valeur maximale `m` présente dans `D` quand `M` est la liste obtenue par

```
M = [mot[:-1] for mot in open('mots.txt')]
```

où le fichier `mots.txt` est disponible sur la page Moodle du cours.

(3) **(TP)** Afficher, pour toutes les clés `E` de valeur `m`, les `m` mots de `M` qui sont une réalisation de `E`.

Exercice 9 (calcul de limite).

Écrire une fonction Python `limite(a,n)` qui prend en entré un réel `a` et un entier `n` $\in \mathbb{Z}$, et retourne le réel L, ou la valeur `inf`, ou `-inf`, ou `None` selon que la limite

$$\lim_{x \to 0} ax^n$$

vaut L, $+\infty$, $-\infty$, ou n'existe pas. On veillera à bien traiter tous les cas (valeurs particulières de `a` et `n`).

Exercice 10 (précision machine).

(1) Proposer des codes Python permettant de déterminer les réels x et y (de type `float`) caractérisés par les propriétés suivantes. On admettra que x et y sont des puissances de 2.

(a) x est le plus petit réel entier positif tel que $x + 1$ soit numériquement égal à x (`x+1==x`)

(b) y est le plus petit réel strictement positif tel que $1 + y$ soit numériquement différent de 1.

(2) **(TP)** Calculer numériquement x et y. Quel est le lien entre les deux ?