DS n° 04 – DS04

- Faire tous les exercices dans un même fichier NomPrenom.py à sauvegarder ;
- mettre en commentaire l'exercice traité (ex : # Exercice 1) ;
- il est possible de demander un déblocage pour une question **entre 8h et 14h** sur la classe virtuelle de vos enseignants. Il vous sera envoyé par mail ;
- **vous avez 3h** pour faire le sujet ;
- il faut vérifier avant de téléverser votre fichier que le code peut s'exécuter et qu'il affiche les résultats que vous attendez ;
- téléversez votre fichier réponse .py sur le site de dépôt de fichier de la PTSI.

Exercice 1

Soit la suite $(u_n)_{n\in\mathbb{N}}$ définie par :

$$u_0 = N \text{ (entier naturel non nul)} \quad \text{et} \quad \forall n \in \mathbb{N}, \quad u_{n+1} = \begin{cases} u_n/2 \text{ si } u_n \text{ est pair} \\ 3u_n + 1 \text{ si } u_n \text{ est impair} \end{cases}$$

1. À l'aide notamment des instructions `plot` et `show` de la bibliothèque `matplotlib.pyplot`, représenter graphiquement les 50 premiers termes de la suite (u_n) avec $N = 7$, puis $N = 18$.
2. On conjecture que pour tout entier naturel N non nul, il existe un plus petit entier n tel que $u_n = 1$. Que se passe-t-il si c'est le cas ? (on mettre la réponse en commentaire).
   Si cet entier existe, on l'appelle "durée de vol" de la suite (u_n) de valeur initiale N.
3. Écrire une fonction `vol` d'argument un entier N, renvoyant la durée de vol de la suite (u_n) de valeur initiale N.
4. Représenter la durée de vol en fonction du point de départ N, pour les valeurs de N inférieures à 1000.
5. Représenter la distribution des durées de vol à l'aide d'un histogramme (on pourra utiliser la commande `hist` de `matplotlib.pyplot`).

Exercice 2

Pour un entier naturel $n \geq 2$, on appelle **diviseurs propres de** n les entiers naturels strictement inférieurs à n qui divisent n.
Par exemple la liste des diviseurs propres de 100 est $[1, 2, 4, 5, 10, 20, 25, 50]$.
On va s'intéresser à la somme de ces diviseurs propres. Pour 100, elle vaut par exemple 117.

1. Écrire une fonction `LDP` d'argument un entier naturel n qui renvoie la liste de ses diviseurs propres. La tester pour $n = 100$.
2. Écrire une fonction `SDP` d'argument un entier naturel n qui renvoie la somme de ses diviseurs propres. La tester pour $n = 100$.
3. On dit qu'un entier naturel est **parfait** s'il est égal à la somme de ses diviseurs propres. Écrire une fonction `parfaits` d'argument un entier naturel K qui renvoie la liste des entiers p parfaits inférieurs ou égaux à K. La tester pour $K = 2000$.
4. On dit que deux entiers distincts sont **amicaux** si chacun est égal à la somme des diviseurs propres de l'autre. Écrire une fonction `amicaux` d'argument un entier naturel K qui renvoie la liste de tous les couples (p, q) d'entiers amicaux tels que $p < q \leq K$. Tester `amicaux` pour $K = 300$, $K = 1300$, $K = 5000$.
   Remarque : si le temps de calcul est trop long, il faut peut-être optimiser votre programme.

Exercice 3

1. Télécharger le fichier `algo-pi.txt` qui est avec l'énoncé et enregistrer-le dans le dossier où vous travaillez.
2. Le fichier `algo-pi.txt` contient les premières décimales de π, sur une seule ligne et sans espace entre les chiffres.
   Récupérer le contenu de ce fichier sous forme d'une chaîne de caractères que l'on nommera `decpi`.

3. Faire afficher les 10 premiers caractères de `decpi`, ses `10` derniers, ainsi que le nombre `nbdec` de caractères de `decpi`.
4. Écrire une fonction `estIci` de trois arguments : deux chaînes de caractères P et M et un entier `i` et qui renvoie `True` si `M` est dans `P` à la position `i` et `-1` sinon.
   Par exemple `estIci('le','Bonjour le monde',8)` renvoie `True`.
5. Écrire une fonction `trouve` de deux arguments, deux chaînes de caractères P et M, qui renvoie un entier naturel p si M est une sous-chaîne de P commençant à la position p, et -1 si M n'est pas une sous-chaîne de P. On n'utilisera pas la méthode `find` de la classe `str`.
   Par exemple, `trouve("BanquePT","an")` donne `1`, `trouve("BanquePT","PT")` donne `6`, alors que `trouve("BanquePT","PSI")` donne `-1`. Comparer avec `"BanquePT".find("an")`, *etc.*
6. Les nombres `"14159"`, `"123456"`, `"12345"`, et `"1789"` se trouvent-ils dans `decpi` ?
7. Si les nombres précédents sont dans `decpi`, en combien d'exemplaires et à quelle(s) position(s) sont-ils ?

Exercice 4

Pour tout ensemble non vide, identifié ici à une liste d'éléments deux à deux distincts, $E = [e_0, \ldots, e_{n-1}]$ de taille n, chacune de ses parties A peut être codée sous forme d'une liste `C` à n éléments contenant des zéros et des uns :

`C[i] = 1` si $e_i \in A$, et `C[i] = 0` sinon.

Par exemple, les parties $\emptyset$, $[a]$, $[b]$ et $[a, b]$ de la liste $[a, b]$ sont respectivement codées par les listes $[0, 0]$, $[1, 0]$, $[0, 1]$ et $[1, 1]$.

1. Écrire une fonction `decoder` de deux arguments E et C renvoyant la partie de la liste E codée par le code C.
   Ainsi, `decoder([2,3,5,7],[1,0,0,1])` donne `[2,7]` ; de même `decoder([2,3,5,7],[0,0,0,0])` donne `[]`.
2. Écrire une fonction `coder` de deux arguments E et A renvoyant le code de la partie A de la liste E.
   Ainsi `coder([2,3,5,7],[2,7])` donne `[1,0,0,1]`.
3. Écrire une fonction `incrementer` d'argument une liste C de zéros et de uns de taille n, représentant sur n bits l'écriture en base 2 d'un entier naturel k, et renvoyant la liste représentant sur n bits l'écriture en base 2 de l'entier $(k + 1)$.
   Par exemple, `[0,0,1,0,1,1,1]` est l'écriture en base 2 sur 7 bits de 23 et `[0,0,1,1,0,0,0]`, l'écriture en base 2 sur 7 bits de 24.
   Remarque : on considère que k sera toujours pris inférieur à $2^n - 1$ de sorte que le problème du nombre qui suit `[1,1,1,...,1,1]` ne se pose pas.
4. En déduire la fonction `parties` d'argument E renvoyant la liste des parties de la liste E.