TP 1 : Révisions

I Bases

I A Range

■ **Exercice 1.** Sans utiliser la machine, indiquer le nombre de termes contenus dans les séquences suivantes :

1. `range(0,10)`.
2. `range(1,11)`.
3. `range(1,12,-1)`.
4. `range(12,2,-1)`.

■ **Exercice 2.** Générer les séquences de nombres suivantes à l'aide de `range` (on pourra vérifier que la solution proposée est correcte en programmant une boucle affichant les termes successifs du range) :

1. `0,1,2,...,10`
2. `1,2,3,...,14`
3. `38,39,40,...,57`
4. La séquence des 57 entiers consécutifs à partir de `41` (inclus).
5. La séquence des 33 entiers consécutifs à partir de `-21` (inclus).
6. La séquence des 73 entiers consécutifs jusqu'à `84` inclus.
7. La séquence des 11 entiers consécutifs jusqu'à `104` exclu.
8. La séquence des entiers pairs de `0` à `44`.
9. La séquence des entiers impairs de `0` à `44`.
10. `12,15,18,..., 93`.
11. `12,19,26,...170,177,184`
12. La séquence de 44 entiers commençant par `12,19,26,...`
13. `18,17,...,1`
14. `18,16,...,2`
15. `18,15,12,...,6`

I B Programmation de fonctions basiques

■ **Exercice 3.** Programmer les fonctions suivantes :

1. $f_1 : x \mapsto x^2 - 2x - 1$.
2. $f_2 : x \mapsto \begin{cases} x^2+1 & \text{si} \quad x<0 \\ x^3 & \text{si} \quad x \in [0,2] \\ \ln(x) & \text{sinon.} \end{cases}$
3. $f_3 : x \mapsto \dfrac{1}{1+\lfloor x^2 \rfloor}$
   Rappel : $\lfloor t \rfloor$ désigne la partie entière de t. Elle s'obtient par la commande `floor` issue soit du module `math`, soit du module `numpy` (préférable au premier).
4. $f_4 : x \mapsto \sqrt{x}\ln x$

I C Boucles

■ **Exercice 4.** Dans chaque cas, dire si la programmation du schéma itératif demandée est plus adaptée avec **a)** une boucle `while`, **b)** une boucle `for` avec interruption de boucle, ou **c)** une boucle `for` sans interruption et dire pourquoi.

1. Le calcul des 30 premiers termes d'une suite donnée.
2. La simulation de 10 lancers d'une pièce de monnaie.
3. La recherche du premier terme d'une suite égal à 10.
4. La recherche du premier terme négatif ou nul d'une liste.
5. La recherche du premier caractère égal à `w` d'une chaine de caractères donnée.
6. La simulation d'un tirage successif avec remise d'une boule dans une urne jusqu'à l'obtention d'une boule blanche.
7. Le calcul du plus grand terme d'une liste.

■ **Exercice 5.** Écrire en python la condition de la boucle `while` dans chacun des cas suivants :

1. Les termes consécutifs de la suite (u_n) sont calculés et stockés dans une variable `u`. On cherche le premier terme u_n pour lequel $u_n > 100$.
2. Les termes consécutifs de la suite (u_n) sont calculés et stockés dans une variable `u`. On cherche le premier terme u_n pour lequel $u_n > 100$ parmi $u_1, \dots, u_{50}$.
3. Les termes consécutifs de la suite (u_n) sont calculés et stockés dans une variable `u`. On cherche le premier terme u_n de rang $n > 1000$ pour lequel $u_n > 100$.
4. On simule une suite de tirages successifs avec remise de boules numérotées. Les numéros des deux derniers tirages successifs sont stockés dans l'ordre dans des variables `s` et `t`. On s'arrête quand la suite des numéros tirés n'est plus croissante.
5. On simule une suite de tirages successifs avec remise de boules numérotées. Les numéros des deux derniers tirages successifs sont stockés dans l'ordre dans des variables `x,y,z`. On s'arrête quand la suite des numéros tirés n'est plus monotone.

I D Calcul de sommes

■ **Exercice 6.** À l'aide d'une boucle `for`, programmer des fonctions prenant en entrée un entier `n`, et renvoyant sortie la valeur de S_n dans les cas suivants :

1. $S_n = \sum_{k=1}^{n} k$.
2. $S_n = \sum_{k=0}^{n} 2^k$.
3. $S_n = \sum_{k=0}^{2n} \frac{k(k-1)}{2}$.
4. $S_n = \sum_{k=0}^{n} u_k$ où $u_n = \frac{1}{n}$ si n est impair, et $u_n = e^{-n}$ sinon.

II Suites

■ **Exercice 7.** Écrire une fonction `u(n)` prenant en entrée un entier `n` et renvoyant en sortie la valeur du terme u_n de la suite définie par :

$$u_1 = 3 \quad \forall n \geq 1 \quad u_{n+1} = \frac{nu_n - 1}{2}$$

Si vous ne vous êtes pas trompés, vous devez obtenir $u_{15} =$`-1035072.125`.

■ **Exercice 8.**
Écrire une fonction `liste(n)` prenant en entrée un entier n et renvoyant en sortie la liste des termes $[u_0, \dots, u_n]$ de la suite récurrente définie par :

$$u_0 = 0 \quad \forall n \geq 0 \quad u_{n+1} = \ln(2 + u_n).$$

■ **Exercice 9.**
Écrire une fonction `Fibo(n)` qui prend en entrée un entier n et qui renvoie en sortie le terme F_n de la suite de Fibonacci. Pour rappel, la suite de Fibonacci est définie par :

$$F_0 = F_1 = 1 \quad \forall n \geq 0 \quad F_{n+2} = F_{n+1} + F_n.$$

■ **Exercice 10.** On pose $H_n = \sum_{k=1}^{n} \frac{1}{k}$. Écrire une fonction `rang(M)` prenant en entrée un flottant M et qui renvoie en sortie le plus petit entier n tel que $H_n > M$. Si vous ne vous êtes pas trompés, `rang(10)` devrait renvoyer `12367`.

III Listes

III A Opérations sur les listes - Slicing

■ **Exercice 11.** Recopier le script suivant :

```
n = 15
L = [k**2 for k in range(1,n+1)]
```

1. Extraire de la liste `L` les sous-listes suivantes :
   a) La sous-liste `L1` des 3 premiers termes de `L`.
   b) La sous-liste `L2` des 5 derniers termes de `L`.
   c) La sous-liste `L3` contenant les termes d'indice impair de `L`.
   d) La sous-liste `L4` contenant les termes d'indice pair de `L`.
   e) La sous-liste `L5` contenant un terme sur trois de `L` en partant du premier.
   f) La sous-liste `L6` contenant un terme sur cinq de `L` en partant du deuxième.
2. Construire la liste `M` obtenue en insérant dans `L` un `0` en indice `8`.

■ **Exercice 12.**

1. Écrire une fonction `reverse(L)` prenant en entrée une liste `L` et renvoyant en sortie la liste dans laquelle les items de `L` sont rangés en sens inverse.
2. Même question en utilisant les opérations sur les listes.

■ **Exercice 13.**
Écrire une fonction `saute(L,p)` qui prend en entrée une liste `L`, une entier `p` et qui renvoie en sortie la sous-liste constitué d'un terme sur deux de `L` à partir de son terme d'indice `p`.

III B Listes en compréhension

■ **Exercice 14.** Construire les listes suivantes en compréhension contenant les mêmes objets que les ensembles mathématiques suivants (les p-listes mathématiques seront représentées en python par des listes ici) :

1. $E_1 = \{a_k \quad k \in \{1,\dots,15\}\}$ où on a posé $a_n = \dfrac{1}{n}$.
2. $E_2 = \{b_k \quad k \in \{1,\dots,15\}\}$ où on a posé $b_n = 2^n$.
3. $E_3 = \left\{\dfrac{1}{n(n+1)} \quad n \in \{1,\dots,10\}\right\}$.
4. $E_4 = \{|\cos n - \sin n| \quad n \in \{1,\dots,10\}\}$
5. $E_5 = E_1 \setminus \{1/3, 1/11, 1/8\}$.
6. $E_6 = \{(a_k, b_k) \quad k \in \{1,\dots,15\}\}$.
7. $E_7 = \{(2,4,8,16),(3,9,27,81),(4,16,64,256),(5,25,125,625)\}$.
8. $E_8 = \{(1),(2,3),(3,4,5),(4,5,6,7),(5,6,7,8,9)\}$.
9. $E_9 = \{(i,j) \quad i \in \{-1,0,1\}, j \in \{-1,0,1\}\}$.
10. $E_{10} = E_9 \setminus \{(0,0)\}$.

III C Exercices complémentaires sur les listes

■ **Exercice 15.**
Écrire une fonction `diff(L)` prenant en entrée une liste de flottants `L` et renvoyant en sortie la liste des `L[i+1]-L[i]`.

■ **Exercice 16.** Écrire une fonction `cumsum(L)` prenant en entrée une liste de flottants `L` et renvoyant en sortie la liste `S` dont le k-ème item vaut $\sum_{i=0}^{k} \texttt{L[i]}$.

■ **Exercice 17.**
Écrire une fonction `est_dans(L1,L2)` prenant en entrée deux listes `L1` et `L2` et renvoyant en sortie `True` si tous les éléments de `L2` sont dans `L1` et `False` sinon.

■ **Exercice 18.**
Écrire une fonction `maxi(L)` prenant en entrée une liste de flottants et renvoyant en sortie la valeur de son plus grand élément

■ **Exercice 19.** Écrire une fonction `pos_max(L)` prenant en entrée une liste de flottants et renvoyant en sortie la liste des indices en lesquels se trouve son plus grand élément. Utiliser l'exercice précédent.

■ **Exercice 20.**
Écrire une fonction `sans_rep(L)` prenant en entrée une liste `L` et renvoyant en sortie la liste des valeurs distinctes de `L`. Par exemple, si `L=[1,3,'toto',0,2,1,2,'2','toto']`, la sortie est `[1,3,'toto',0,2,'2']`.
Indication : Partir d'une liste `M` initialement vide, parcourir `L` et ajouter l'élément courant dans `M` si il n'est pas déjà dans `M`. À la fin, `M` contient tous les objets de `L` sans répétition.

■ **Exercice 21.**
Écrire une fonction `multiplicite(L)` prenant un entrée une liste `L` et renvoyant en sortie le tuple `(L1,M)` où `L1` est la liste des items distincts de la liste `L` et `M` est la liste contenant en position `i` le nombre d'occurrence dans `L` de `L1[i]`. Par exemple, si `L=[1,3,'toto',0,2,1,2,'2','toto']`, la sortie est `([1,3,'toto',0,2,'2'],[2,1,2,1,2,1])`.