Introduction
Exercices - Enoncés

Pour les exercices de ce document, on utilisera le langage Python et, si besoin est, les bibliothèques et/ou modules requis (`math`, `random`, `matplotlib`…).

Les niveaux sont fournis à titre indicatif.

Pour certains exercices, il est suggéré d'étudier une version récursive des fonctions demandées. Ces études sont à priori réservées aux 5/2.

Niveau I

Exercice I-1

a) A la console, obtenir la liste des 20 premiers entiers naturels pairs (respectivement impairs) :
   - à l'aide de définitions explicites.
   - à l'aide de sous-listes.

b) Ecrire une fonction `ArithTermes` (respectivement `GeomTermes`) qui reçoit en argument un entier naturel non nul n et deux réels a et b et renvoyant les n premiers termes de la suite arithmétique (respectivement géométrique) de premier terme a et de raison b.

Exercice I-2

Soit n un entier naturel non nul.
Ecrire une fonction `ValidPermut` qui reçoit n en argument et une liste L d'entiers et renvoie `True` si L est une permutation de $[\![1\,;n]\!]$ et `False` sinon.

Exercice I-3

Ecrire une fonction `ASC_List` (respectivement `DESC_List`) qui reçoit en argument une liste non vide d'entiers et renvoie `True` si la liste est croissante (respectivement décroissante) et `False` sinon.
Remarques :

- On pourra étudier une version récursive de la fonction `ASC_List`.
- On pourra écrire une fonction `DESC_List` faisant appel à la fonction `ASC_List`.

Exercice I-4

Ecrire une fonction `PermCirc` qui reçoit en argument une liste L non vide d'entiers et effectue une permutation circulaire sur L. Par exemple, si `L=[1,2,3,4]`, `PermCirc(L)` renverra `[4,1,2,3]`.

Exercice I-5

Ecrire une fonction `LenStrings` qui reçoit en argument une liste non vide de chaînes de caractères et renvoie la liste des longueurs de ces chaînes.

Exercice I-6

Ecrire une fonction `IntTrap` renvoyant une valeur approchée de l'intégrale $\int_a^b f(x)\mathrm{d}x$ par la méthode des trapèzes. La fonction recevra comme arguments :

- `f` : le nom de la fonction à intégrer.
- `a` et `b` : les bornes d'intégration.
- `N` : le nombre de pas de la subdivision de l'intervalle sur lequel on intègre.

On rappelle que dans ces conditions, on a :

$$\int_a^b f(x)\mathrm{d}x \simeq h\times\left(\frac{f(a)}{2}+f(a+h)+f(a+2h)+\ldots+f\left(a+(N-1)h\right)+\frac{f(b)}{2}\right)$$

où : $h=\dfrac{b-a}{N}$.

Niveau II

Exercice II-1

Ecrire une fonction `IsArithSeq` (respectivement `IsGeomSeq`) qui reçoit en argument une liste numérique comportant n éléments ($n \geq 3$) et renvoie `True` si ces éléments sont les termes consécutifs d'une suite arithmétique (respectivement géométrique).

Exercice II-2

Ecrire une fonction `IsPalindrome` qui reçoit en argument une liste non vide et renvoie `True` s'il s'agit d'un palindrome.

On pourra étudier une version récursive de la fonction `IsPalindrome`.

Exercice II-3

Ecrire une fonction `StringDecompo` recevant en argument une chaîne de caractères `cc` (ne comportant que des lettres minuscules non accentuées) et renvoyant deux chaînes de caractères :

- La première, `vo`, contenant uniquement (et dans le même ordre) les voyelles de `cc`.
- La seconde, `co`, contenant uniquement (et dans le même ordre) les consonnes de `cc`.

Exercice II-4. Constante d'Euler

On rappelle que la suite $\left(u_n\right)_{n\in\mathbb{N}^*} = \left(\sum_{k=1}^{n} \frac{1}{k} - \ln\left(n\right)\right)_{n\in\mathbb{N}^*}$ converge vers la constante d'Euler notée γ et dont une valeur approchée à 10^{-10} est : 0,577 215 664 9.

Ecrire une fonction `Euler` qui reçoit comme argument un flottant `epsilon` et renvoie le plus petit entier naturel non nul `N` tel que $\left|u_{N+1} - u_N\right| < \text{epsilon}$.

On s'intéressera aux fonctions `log` et `log1p` du module `math` de `Python`.

Exercice II-5. Algorithme de Heron

Soit $a>0$ et $(u_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} u_0 > 0 \\ u_{n+1} = \frac{1}{2}\left(u_n + \frac{a}{u_n}\right) \end{cases}$$

On rappelle que pour toute valeur de $u_0 > 0$, la suite (u_n) converge vers $\sqrt{a}$.

Ecrire une fonction `Heron` qui reçoit trois réels strictement positifs `a`, `b` (correspondant à a et u_0), `epsilon` et qui renvoie N et u_N où N est le plus petit entier naturel tel que $|u_{N+1} - u_N| <$ epsilon.

Exercice II-6

Ecrire une fonction `EcartMax` qui reçoit une liste d'entiers naturels (comportant au moins deux éléments) et qui renvoie la plus grande différence (en valeur absolue) entre deux éléments de la liste ainsi que les indices de deux éléments correspondant à cette différence maximale.

Exercice II-7

Ecrire une fonction `nPlotRac` qui reçoit en argument un entier naturel `n` non nul et trace le polygone régulier ayant pour sommets les points d'affixes les racines nièmes de l'unité.

On pourra utiliser le module `turtle` de Python ou le module `pyplot` de la bibliothèque `matplotlib`.

Exercice II-8

Ecrire une fonction Python qui reçoit en argument un tuple de tuples (ces derniers étant au moins au nombre de 4) correspondant aux coordonnées des sommets consécutifs d'un polygone et retournant le booléen `True` ou `False` selon que le polygone considéré est ou non convexe.

La fonction recevra un second argument optionnel, également booléen, conduisant au tracé du polygone lorsque positionné à `True` (ce sera également la valeur par défaut). Pour ce faire, on utilisera le module `pyplot` de la bibliothèque `matplotlib`.

Exercice II-9. La conjecture de Syracuse

On s'intéresse aux suites dites « de Syracuse » (la suite de l'énoncé vous donnera quelques éclaircissements sur ce nom apparemment… étrange ! ☺) d'entiers naturels non nuls définies de la façon suivante :

$$\begin{cases} u_0 \in \mathbb{N}^* \\ \forall n \in \mathbb{N}, u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases} \end{cases}$$

1. Que peut-on dire de la suite lorsque $u_0 \in \{1;2;4\}$?

2. Ecrire une fonction `Syracuse` qui reçoit en argument un entier naturel non nul (variable `u`) et affiche tous les termes de $(u_n)_{n\in\mathbb{N}}$ jusqu'au premier terme valant 1.
   Par exemple, avec `u=8`, la fonction affichera :

```
8
4
2
1
```

3. Tester la fonction `Syracuse` pour diverses valeurs de `u` et émettre une conjecture.

Cette conjecture a été émise au milieu du 20ème siècle après que le problème a été diffusé aux Etats-Unis en 1952 à l'université de Syracuse (d'où le nom…). La conjecture a été vérifiée pour tout $u_0 < 1,25 \times 2^{62}$ et ces calculs « poussent » de nombreux mathématiciens à penser qu'elle soit vraie… Pour autant, son éventuelle démonstration est toujours attendue ! ☺

Pour une suite de Syracuse $(u_n)_{n\in\mathbb{N}}$ donnée, on définit :

- Le temps de vol : c'est le plus petit indice *n* tel que $u_n = 1$.
- Le temps de vol en altitude : c'est le plus petit indice *n* tel que $u_{n+1} \leq u_0$.
- L'altitude maximale : c'est la valeur maximale de la suite.

4. Compléter la fonction `Syracuse` ci-dessus pour écrire une fonction `Syracuse2` fournissant, en plus de l'affichage de `Syracuse`, les éléments suivants :
   - Une phrase avertissant que le cycle limite a été atteint.
   - Le temps de vol.
   - Le temps de vol en altitude.
   - L'altitude maximale.

   Pour tester `Syracuse2` :

u_0	19	121	1515
Temps de vol	20	95	140
Temps de vol en altitude	5	2	7
Altitude maximale	88	9232	65608

Niveau III

Exercice III-1 (d'après oral PSI 2016)

Ecrire une fonction qui reçoit une liste numérique non vide et en renvoie la (ou une des) plus grande(s) sous-liste(s) croissante(s).

Exercice III-2

a) Ecrire une fonction `ListBin` qui reçoit comme argument un entier naturel *n* non nul et génère une liste aléatoire de *n* éléments valant équiprobablement 0 ou 1 (liste binaire aléatoire).

b) Ecrire une fonction `MaxBin` qui reçoit comme argument un entier naturel `n` non nul et un booléen. La fonction `MaxBin` devra :

- générer et afficher une liste binaire aléatoire `L` de `n` éléments.
- renvoyer l'indice de début et la longueur de la sous-liste de `L` contenant un nombre maximal de 1 ou de 0 selon que le booléen vaut `True` ou `False`.

Exercice III-3. Algorithme de Newton-Raphson

On se donne une fonction *f* définie sur un intervalle $[a\,;b]$, dérivable sur $]a\,;b[$ (de dérivée non nulle sur cet intervalle) et s'annulant une seule fois sur $]a\,;b[$.
L'unique solution sur $]a\,;b[$ de l'équation $f(x)=0$ est notée α.

L'algorithme de Newton-Raphson (ou méthode de la tangente) correspond à la suite $(x_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} x_0 \in]a\,;b[\\ \forall n \in \mathbb{N},\ x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)} \end{cases}$$

On suppose enfin que l'on a : $\forall n \in \mathbb{N},\ x_n \in]a\,;b[$.
Nous ne revenons pas ici sur l'interprétation géométrique de cet algorithme (mais il convient de l'avoir clairement présente à l'esprit).

Pour la mise en œuvre de cet algorithme, on utilise trois critères d'arrêt (interruption des itérations dès lors qu'au moins l'un de ces critères est satisfait) :

- $\left|f\left(x_n\right)\right| < \varepsilon_1$
- $\left|x_{n+1} - x_n\right| < \varepsilon_2$
- $n \leq N$

ε_1 et ε_2 sont des réels strictement positifs et *N* est un entier naturel non nul.

1. Dans cette question, on suppose que la fonction dérivée f' est donnée (fonction `df`). Ecrire une fonction `NR_base` qui reçoit comme argument :
   - `e1` correspondant à ε_1.
   - `e2` correspondant à ε_2.
   - `N` correspondant à *N*.
   - `x` correspondant à x_0.

   La fonction `NR_base` devra afficher les deux dernières valeurs calculées de la suite $\left(x_n\right)_{n\in\mathbb{N}}$ et les images correspondantes par la fonction *f*.

2. Dans cette question, la fonction dérivée f' n'est pas donnée et on utilisera l'approximation :

$$f'\left(x_n\right) \simeq \frac{f\left(x_n + h\right) - f\left(x_n - h\right)}{2h}$$

   où *h* est un réel non nul.

   - Ecrire une nouvelle fonction `NR_dapprox` qui recevra comme arguments les mêmes arguments que la fonction `NR_base` et un réel non nul *h* comme cinquième argument. La fonction `NR_dapprox` devra afficher les mêmes valeurs que la fonction `NR_base`.

   - L'approximation précédente peut s'avérer coûteuse en temps de calcul. Au lieu d'évaluer le rapport $\frac{f\left(x_n + h\right) - f\left(x_n - h\right)}{2h}$ à chaque itération, on peut décider de n'effectuer ce calcul que toutes les *m* itérations, la valeur courante étant donc utilisée *m* fois.

     Ecrire une nouvelle fonction `NR_dapprox2` qui recevra comme arguments les mêmes arguments que la fonction `NR_dapprox` et un entier *m* strictement supérieur à 1 comme sixième argument. La fonction `NR_dapprox2` devra afficher les mêmes valeurs que la fonction `NR_dapprox`.

Exercice III-4. Accélération de la convergence de $\sum \frac{1}{n^2}$ (méthode de Stirling)

On pose : $\forall n \in \mathbb{N}^*, S_n = \sum_{k=1}^{n} \frac{1}{k^2}$ et on admet : $S = \sum_{k=1}^{+\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$

Par ailleurs, pour tout entier naturel *k*, on pose :

$$f_k : \begin{cases} \mathbb{R}_+^* \to \mathbb{R} \\ x \mapsto f_k(x) = \dfrac{1}{x(x+1)(x+2)...(x+k)} \end{cases}$$

(on a donc, pour tout réel *x* strictement positif : $f_0(x) = \frac{1}{x}$)

On définit enfin :

$$\Delta : \begin{cases} \mathscr{C}^0(\mathbb{R}_+^*, \mathbb{R}) \to \mathscr{C}^0(\mathbb{R}_+^*, \mathbb{R}) \\ f \to \Delta(f) / \Delta(f)(x) = f(x+1) - f(x) \end{cases}$$

1. Pour tout entier naturel *k* non nul, exprimer $\Delta(f_{k-1})$ en fonction de *k* et de f_k.
2. Pour tout entier naturel *k* non nul, établir la convergence de la série $\sum f_k(p)$.
3. Vérifier que l'on a : $\forall k \in \mathbb{N}^*, \sum_{p=n+1}^{+\infty} f_k(p) = \frac{1}{k} \times \frac{1}{(n+1)(n+2)...(n+k)}$.
4. Etablir, pour tous *p* et *q*, entiers naturels non nuls, l'égalité suivante :
$$\frac{1}{p^2} - \sum_{k=1}^{q} (k-1)!\, f_k(p) = \frac{q!}{p} f_q(p)$$
5. Pour tous *n* et *q*, entiers naturels non nuls, on pose :
$$S'_n = S_n + \sum_{k=1}^{q} \frac{(k-1)!}{k \times (n+1)(n+2)...(n+k)}$$
Montrer que l'on a : $0 \le S - S'_n \le \dfrac{(q-1)!}{(n+1)^2 (n+2)...(n+q)}$

Dans la suite de l'exercice, on prend $q = 2$.

6. Donner S'_n et l'encadrement précédent de $S - S'_n$.
7. Soit ε un réel strictement positif.
Ecrire une fonction Python `SIC_Stirling` recevant en argument la variable (flottant) `epsilon` et renvoyant S'_n telle que $0 \le S - S'_n \le \varepsilon$.