1 Environnement de travail

De nombreux outils permettent d'écrire des fichiers python. Ces fichiers sont simplement des fichiers textes, ils peuvent donc être écrits avec des éditeurs de texte simples (comme Bloc-Note) ou plus adaptés permettant notamment la coloration syntaxique (comme NotePad++). Des environnements de développements intégrés (dits IDE) permettent de disposer dans une même fenêtre d'un éditeur de texte adapté, d'outils de déboggage et d'un interpréteur (encore appelé console).

Nous utiliserons l'IDE Spyder qui est inclus dans un certain nombre de distribution (et peut-être parfois Pyzo selectionné par certains concours pour les épreuves orales). Pour votre ordinateur personnel, vous pourrez télécharger ces environnements avec les distributions :

- Anaconda pour Windows, Mac, et Linux (qui inclut Spyder) à l'adresse `https://www.anaconda.com/download/`.
- Miniconda + Pyzo (environnement de développement intégré alternatif à Spyder) pour Windows, Mac, et Linux disponibles aux adresses `https://conda.io/miniconda.html` et `http://www.pyzo.org/`
- WinPython pour Windows 64 bits (qui inclut Spyder) à l'adresse `https://winpython.github.io/`

Pour l'utilisation de Spyder, commencer par *enregistrer le fichier vide* automatiquement créé sous un nom pertinent (surtout sans apostrophe!) puis régler les options d'exécution (touche Ctrl + F6 du clavier) en : *Exécuter dans un nouvel interpréteur dédié* avec l'option *Intéragir avec l'interpréteur Python après l'exécution*. L'exécution des programmes se fait avec la touche F5. A chaque fois qu'un fichier est enregistré en changeant de destination, les options d'exécution doivent être configurées à nouveau.

2 Suite de Syracuse

La suite de Syracuse est définie par la relation de récurrence suivante :

$$u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$$

1. Ecrire une fonction qui prend une valeur de u_n en argument et renvoie la valeur de u_{n+1} correspondante.
Indice : utiliser les opérateurs `//` et `%`

2. Le module `pylab` contient les fonctions `plot` et `show` qui permettent d'afficher des graphiques. Importer ces fonctions dans votre programme.

3. Ecrire une fonction qui prend en argument un terme initial et un entier $rangMax$ et qui affiche sur un graphique les premiers termes, jusqu'au rang $rangMax$, de la suite en appelant la fonction `plot` avec pour argument n, u_n et la chaîne de caractères `"ro"`. Appeler également une fois la fonction `show` sans argument juste avant d'interrompre la fonction.

Tester la fonction avec les arguments 17 (terme initial) et 20 (nombre de termes).

On remarque que si la suite atteint la valeur 1 à un certain rang, elle devient alors 3-périodique. La conjecture de Syracuse stipule que quelque soit l'entier naturel utilisé comme terme initial, la suite finit toujours par atteindre cet état.

Par suite, on appelle *longueur de vol* le nombre de termes de la suite avant d'atteindre la valeur 1, et *hauteur de vol* le plus grand terme de la suite.

4. Ecrire une fonction qui prend un argument le terme initial de la suite et qui renvoie la longueur de vol correspondant.

5. Ecrire une fonction qui prend un argument le terme initial de la suite et qui renvoie la hauteur de vol correspondant.

6. Ecrire une fonction qui prend un entier `entierMax` en argument et qui déterme l'entier $1 \leqslant k \leqslant entierMax$ dont la hauteur de vol est maximale.

7. Dans la fonction précédente, ajouter un argument : `a_maximiser` qui sera elle-même une fonction (en l'occurence, soit la hauteur, soit la longueur). Ainsi, dans cette questionn, on généralise la précédente en permettant de choisir quelle fonction on cherche à maximiser.

3 Résolution des équations du second degré

8. Ecrire un programme qui demande les trois coefficients d'un polynôme du second degré à l'utilisateur (penser à bien convertir le résultat de input en une valeur numérique d'un type adapté) puis calcule le discriminant du polynôme et renvoie les racines. On pourra traiter le cas où les racines sont complexes.

4 Manipulation de fonctions mathématiques

9. Commencer un nouveau programme et y définir une constante globale `pas` par exemple égale à 10^{-3}, deux réels `borneInf` et `borneSup` délimitant un intervalle fermé et une fonction `f` définie sur cet intervalle. Programmer ensuite le calcul des extrema *locaux* par un parcours discret de l'intervalle en sautant de `pas` en `pas`.

Améliorer le programme pour effectuer un seul calcul d'image à chaque tour de boucle. (La fonction `f` prend potentiellement du temps à être évaluée). Application à $f : x \mapsto -(x-3)(x-5)(x-7)$ sur $[0;10]$.

10. Ecrire une fonction qui prend en argument une fonction de la variable réelle et qui renvoie une approximation de sa fonction dérivée. On utilisera $\frac{f(x+h)-f(x-h)}{2h}$ comme approximation du nombre dérivé en x avec `h=1e-9`.

11. On s'intéresse à la recherche d'un maximum local d'une fonction de deux variables. On utilise le principe suivant : on se fixe un pas, puis on part d'un couple (x,y) donné et on effectue en boucle les deux actions suivantes :

1. on considère trois possibilités : augmenter x du pas, diminuer x du pas, et ne pas changer x, celle qui donne l'image la plus grande devient la nouvelle valeur de x.
2. on effectue les mêmes considérations pour y.

On arrête le programme dès qu'on a successivement gardé la valeur de x et de y et on renvoie ces valeurs.

Ecrire ce programme. Application à $f : x,y \mapsto \frac{1}{x^2+y^2+1}$ et $(x,y) = (1,1)$ et à $f : x,y \mapsto \cos(x-y) - \cos(x+y)$.

12. Ecrire un programme réalisant une étude de signe sur un intervalle d'une fonction continue, on utilisera à nouveau un pas et un parcours discret. Le résultat devra comporter les zéros de la fonction et le signe entre chaque zéro de la fonction.