Travaux pratiques sur les bases du langage

Exercice 1 : (difficulté:)

Ecrire un programme qui affiche la suite de 12 nombres dont chaque terme est égal au triple du précédent. On fera l'exercice avec une boucle `for` puis une boucle `while`.

Exercice 2 : (difficulté:)

Ecrire un programme qui affiche

```
*
**
***
****
*****
******
```

Exercice 3 : (difficulté:)

Soit la chaîne de caractères suivante :

```
Chaine='Ceci est une chaine.'
```

Question 1.
Est-ce que `Chaine` est un objet ? Quelles sont ses méthodes ?

Question 2.
Mettre dans `CHAINE` cette chaîne en majuscules.

Question 3.
Mettre dans `nb_mot` le nombre de mots de la phrase.

Question 4.
Compter le nombre d'occurrences de chaque lettre (on utilisera un dictionnaire et la fonction *get*).

Exercice 4 : (difficulté:)

Ecrire une fonction qui renvoie les approximations des dérivées première et seconde d'une fonction f en un point x en utilisant des différences finies :

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h},$$

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2},$$

où le pas h aura comme valeur par défaut 0.0001. Faire en sorte que l'on puisse appeler des fonctions ayant des paramètres supplémentaires (par exemple $f(x, a, b, ...)$).

Exercice 5 : (difficulté:)

Un triangle quelconque peut être caractérisé par ses trois sommets (x_1, y_1), (x_2, y_2) et (x_3, y_3). L'aire de ce triangle peut être calculé à l'aide de la formule suivante

$$aire = \frac{1}{2} \left| (x_1 - x_3)(y_2 - y_1) - (x_1 - x_2)(y_3 - y1) \right| .$$

Ecrire une fonction qui renvoie l'aire d'un triangle donné en paramètre sous la forme d'une liste ayant les coordonnées des trois sommets (par exemple `[[0, 0], [0, 1], [1, 0]]`).

Exercice 6 : (difficulté:)

Définir une matrice et un vecteur numpy de la manière suivante

```
import numpy as np
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
b = np.array([-3, -2, -1])
```

Question 1.
Quels sont les types des données de A et de b ?

Question 2.
Faire le produit matrice-vecteur et afficher le résultat.

Exercice 7 : (difficulté:)

Extraire la matrice 2×2 qui se situe dans le coin en bas à droite de la matrice A de l'exercice précédent. Ajouter à cette matrice une autre matrice 2×2 et multiplier le résultat par une autre matrice 2×2. Insérer le résultat final dans le coin en haut à gauche de la matrice A.

Exercice 8 : (difficulté:)

Ecrire le script suivant

```
from numpy import linspace

x = linspace(0, 1, 3)
# y = 2*x + 1
y = x; y *= 2; y += 1
print x, y
```

Question 1.
Expliquer pourquoi x et y ont les mêmes valeurs. Comment modifier ce script pour conserver la variable x ?

Question 2.
Sauvegarder x et y dans un fichier *npz*.

Exercice 9 : (difficulté:)

La fonction suivante

```
def initial_condition(x):
    return 3.0
```

ne fonctionne pas correctement si x est un tableau numpy. Dans ce cas, la fonction devrait retourner un tableau de la même taille que le tableau x avec toutes ses valeurs à 3.
Effectuer les modifications nécessaires pour que cette fonction fonctionne correctement pour les scalaires et les tableaux.

Exercice 10 : (difficulté:)

La seconde loi de Newton nous dit que la position verticale d'une balle varie au cours du temps t selon la formule

$$y(t) = v_0 t - \frac{1}{2} g t^2,$$

où
– v_0 est la vitesse initiale,
– g est la force de gravité.
La balle part de la position $y(0) = 0$ et on négligera les forces liées à la résistance de l'air. En observant que la balle reviendra après $2v_0/g$ secondes en $y = 0$, écrire une fonction qui renvoie un tableau avec les $y(t)$ pour $t \in [0, 2v_0/g]$. On choisira un découpage uniforme.

Exercice 11 : (difficulté:)

Le troisième *la* d'un piano est le son émis par le diapason. La frappe de ce *la* fait vibrer la corde correspondante du piano à une fréquence de $440Hz$: c'est le nombre d'oscillations par seconde de la corde. Mathématiquement, cette oscillation peut être représentée par un sinus évoluant dans le temps

$$s(t) = Asin(2\pi f t),$$

où A est l'amplitude, f est la fréquence. La qualité d'un CD est donnée par 44100 échantillons par seconde. Il est possible d'avoir d'autres taux d'échantillonnage. Nous noterons r le nombre d'échantillons. On peut donc calculer de manière discrète $s(t)$

$$s_r = Asin\left(2\pi f \frac{n}{r}\right) \text{ pour } n = 0, \cdots, r \cdot length,$$

où $length$ est le nombre de secondes où le son est émis.

Question 1.
Ecrire une fonction *note* qui renvoie le sinus échantillonné. La fonction prendra en paramètres la fréquence, la durée de la note en seconde, l'amplitude et le nombre d'échantillons qui seront mis par défaut à 1 et à 44100.

Question 2.
Soit un piano virtuel avec 88 touches. La 49eme touche est le *la* à $440Hz$. La fréquence des autres touches peut être calculée à l'aide de la formule suivante

$$K_n = K_{49} 2^{\frac{49-n}{12}},$$

où $K_{49} = 440$ et n est le numéro de la touche. Ecrire une fonction qui retourne K_n.

Question 3.
Ecrire une fonction qui joue « au clair de la lune »sachant que les touches pour la mélodie sont

$$[52, 52, 52, 54, 56, 54, 52, 56, 54, 54, 52, 52, 52, 52, 54, 56, 54, 52, 56, 54, 54, 52,$$
$$54, 54, 54, 54, 49, 49, 54, 52, 51, 49, 47, 52, 52, 52, 54, 56, 54, 52, 56, 54, 54, 52].$$

On concaténera les notes en utilisant la fonction `concatenate` de *numpy* et on sauvegardera la mélodie au format wav en utilisant le module `wavefile` de *scipy* dont voici un exemple d'utilisation

```
import numpy as np
import scipy.io.wavfile as wav

song = note(440, 1.)
scaled = np.int16(song/np.max(np.abs(song))*32767)
wav.write('test.wav', 44100, scaled)
```

La carte de l'ordinateur ne peut lire les données que si ce sont des séquences de 2 octets. C'est pourquoi nous refaisons une mise à l'échelle avec la variable `scaled`.

Exercice 12 : (difficulté:)

Un nombre premier est un nombre qui n'est divisible que par un et par lui-même. Écrivez un programme qui établisse la liste de tous les nombres premiers compris entre 1 et 1000, en utilisant la méthode du **crible d'Eratosthène** :
- Créer une liste de 1000 éléments, chacun initialisé à la valeur 1.
- Parcourir cette liste à partir de l'élément d'indice 2 : si l'élément analysé possède la valeur 1, mettez à zéro tous les autres éléments de la liste, dont les indices sont des multiples entiers de l'indice auquel vous êtes arrivé.

Lorsque vous aurez parcouru ainsi toute la liste, les indices des éléments qui seront restés à 1 seront les nombres premiers recherchés.
En effet : A partir de l'indice 2, vous annulez tous les éléments d'indices pairs : 4, 6, 8, 10, etc. Avec l'indice 3, vous annulez les éléments d'indices 6, 9, 12, 15, etc., et ainsi de suite.
Seuls resteront à 1 les éléments dont les indices sont effectivement des nombres premiers.

Exercice 13 : (difficulté:)

L'intégrale d'une fonction $f(x)$ entre $x \in [a, b]$ peut être calculée numériquement à l'aide de la méthode des trapèzes

$$\int_a^b f(x)dx \approx \frac{h}{2}f(a) + \frac{h}{2}f(b) + h\sum_{i=1}^{n-1} f(a + ih), \quad h = \frac{b-a}{n}.$$

Implémenter cette méthode à l'aide d'une fonction Python et d'une boucle.
Ce code est très lent comparé à une version vectorielle. Implémenter la fonction vectorielle puis introduire des mesures de temps à l'aide du module *time* afin de mesurer les gains.
On pourra utiliser la fonction

$$f(x) = 1 + 2x$$

comme fonction test.
Si ce n'est déjà fait, faire en sorte que l'on puisse appliquer la méthode des trapèzes sur n'importe quelle fonction.

Exercice 14 : (difficulté:)

On considère la fonction suivante

$$f(x) = \frac{n}{n+1} \begin{cases} 0.5^{1+1/n} - (0.5-x)^{1+1/n}, & 0 \leq x \leq 0.5 \\ 0.5^{1+1/n} - (x-0.5)^{1+1/n}, & 0.5 < x \leq 1 \end{cases}$$

avec un réel $0 < n \leq 1$. Implémenter une fonction Python vectorielle pour évaluer $f(x)$ pour un vecteur x de taille m ayant ses valeurs entre 0 et 1.

Exercice 15 : (difficulté:)

On souhaite créer une classe *Mesh* permettant de construire des maillages 2D à partir d'un maillage généré par FreeFem ou d'un ensemble d'intervalles. Dans le premier cas, nous aurons des triangles et dans l'autre cas, nous aurons des quadrangles. Il faut donc bien penser la classe pour gérer ces différents éléments. Il faut également noter que ces éléments peuvent appartenir à différentes régions.
On propose la structure de données suivante :

Un maillage est constitué de

1. *sa dimension*
2. *un tableau de points*
3. *un dictionnaire d'éléments où les clés seront*
   - *1 : ligne*
   - *2 : triangle*
   - *3 : quadrangle*

   Chaque valeur des clés de ce dictionnaire est encore un dictionnaire où les clés représenteront les régions et les valeurs les tableaux de connectivités.

Question 1.
Créer une fonction *__init__* permettant d'initialiser le tableau de points à *None* et le dictionnaire des éléments.

Question 2.
Créer une fonction *fromCoords* qui initialise un maillage de quadrangles à partir d'un intervalle suivant l'axe des x et un suivant l'axe y. On aura donc pour l'exemple suivant

```
m = Mesh()
m.fromCoords([0, 1], [0, 1])
```

les données suivantes

```
m.dim = 2

m.nodes = [[0, 0],
           [1, 0],
           [0, 1],
           [1, 1]]

m.elements[1][1] = [[0, 1],
                    [1, 2]]
m.elements[1][2] = [[2, 5],
```

```
                    [5 8]]
m.elements[1][3] = [[8, 7],
                    [7 6]]
m.elements[1][4] = [[6, 3],
                    [3 0]]

m.elements[3][1] = [[0, 1, 4, 3],
                    [1, 2, 5, 4],
                    [3, 4, 7, 6],
                    [4, 5, 8, 7]]
```

Question 3.
Créer une fonction *fromFreeFem* qui initialise un maillage de triangles à partir d'un maillage généré par FreeFem qui a le format suivant

```
ns nt na
x_1 y_1 r_1
x_2 y_2 r_2
...
x_ns y_ns r_ns
t_11 t_12 t_13 rt_1
t_21 t_22 t_23 rt_2
...
t_nt1 t_nt2 t_nt3 rt_nt
a_11 a_12 ra_1
a_21 a_22 ra_2
...
a_na1 a_na2 ra_na
```

On pourra tester cette fonction avec le fichier *membrane.msh*.

Question 4.
Créer une fonction *numberOfNodes* qui renvoie le nombre de points.

Question 5.
Créer une fonction *numberOfElements* qui renvoie le nombre d'éléments total si il n'y a pas de type d'éléments spécifié, le nombre d'éléments d'un certain type sinon.

Question 6.
Créer une fonction *__str__* permettant de dire quel est le comportement lorsque l'on utilise la mot clé *print*. On souhaite avoir l'affichage suivant

```
Mesh
        Number of nodes: 1360
        Number of line elements: 150
        Number of triangle elements: 2568
        Number of quadrangle elements: 0
```

Question 7.
Créer une fonction *__add__* permettant de fusionner deux maillages.