DEVOIR DE SYNTHESE SEMESTRE 2

Matière : INFORMATIQUE

Classes : MP1, PC1, PT1 - Durée : 2 h

Toutes les réponses aux questions doivent être rédigées en ***Python*** *et non pas en syntaxe algorithmique.*

Exercice 1. (5 points)

Soient $f : [a, b] \to R$ une fonction (continue) et un entier $n \in N^*$.

1) On se propose de représenter en Python la fonction G définie par : $G(x) = \int_a^x f(t)dt$.

On utilise la subdivision régulière $(x_0, x_1, ..., x_n)$ de $[a, b]$ de pas $h = \frac{b-a}{n}$, c'est-à-dire $x_k = a + kh$.

Définir une fonction **prim(f, a, b, n)** qui renvoie le graphe de la courbe des points $(x_k, G(x_k))_{0 \le k \le n}$.

Indication :

Le calcul de *chacun* des $G(x_k)$ se fait en utilisant la méthode des trapèzes avec un pas constant $h = \frac{b-a}{n}$.

Ainsi, $G(x_0) = 0$, et pour $k \ge 1$, $G(x_k)$ utilise les valeurs $f(x_0), f(x_1), ..., f(x_k)$.

Exemple :

$$G(x_1) = \int_a^{x1} f(t)dt = \int_a^{x0} f(t)dt + \int_{x0}^{x1} f(t)dt = G(x_0) + \int_{x0}^{x1} f(t)dt .$$

Rappel : Méthode des trapèzes

$$\int_{xk}^{xk+1} f(t)dt \approx \frac{h}{2}(f(x_k) + f(x_{k+1}))$$

Pour le calcul de G, on peut utiliser un tableau ***numpy*** de dimension 1.

2) En utilisant la fonction précédente, écrire les instructions pour représenter graphiquement la courbe de G lorsque la fonction considérée est :

$f : [0, 2] \to R$, $x \mapsto \exp(\sqrt{x})$ et en prenant $n = 100$.

Nota : Importer les modules nécessaires.

Exercice 2. Crible d'Eratosthène **(5 points)**

1) On considère une liste L de longueur n d'entiers à valeurs dans $\{0, 1\}$.

a) Ecrire une fonction **multiple(L, i)** qui, étant donnés une liste L et un entier $i \ge 1$, transforme L en attribuant la valeur 1 à tous les $L[k \times i]$ avec $k \ge 2$ c'est-à-dire en attribuant la

valeur 1 à tous les $L[j]$ où j est un multiple propre de i, c'est-à-dire un multiple de i vérifiant $i<j<n$.
Par exemple, si $L = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$, *multiple*($L$, 2) transforme L en [0, 0, 0, 0, 1, 0, 1, 0, 1, 0].
En effet, *multiple*(L, 2) affecte 1 à toutes les variables $L[j]$ d'indice pair $j > 2$, c'est-à-dire j pair et $j \geq 4$.

b) Ecrire une fonction **suivant(L, i)** qui renvoie, s'il existe, le plus petit entier $j > i$ tel que $L[j]= 0$, et qui renvoie n sinon, où n est la longueur de la liste L.
Exemple : Si $L = [0, 0, 1, 1, 1, 0, 1, 0]$, *suivant*($L$, 1) renvoie 5.

2) La méthode d'Eratosthène permet de déterminer tous les nombres premiers inférieurs à un entier donné n. On sait qu'un nombre $k \geq 2$ n'est pas premier si et seulement s'il admet un diviseur compris entre 2 et $\sqrt{k}$.
L'algorithme d'Eratosthène procède donc ainsi :
- On construit la liste nulle $L = [0, 0, \ldots, 0]$ de longueur n.
- Pour tout entier i vérifiant $i \geq 2$ et $i^2 < n$, on attribue dans L la valeur 1 à tous les multiples propres de i.
- Les entiers $j \geq 2$ vérifiant $L[j] = 0$ sont exactement les nombres premiers qui sont inférieurs strictement à n.

En utilisant cette méthode, écrire une fonction **premiers(n)** qui renvoie la liste des nombres premiers qui sont strictement inférieurs à n.
Par exemple, *premiers*(8) renvoie [2, 3, 5, 7].

Exercice 3 (10 points)
Dans cet exercice, on se propose d'étudier en Python quelques propriétés associées à des matrices carrées particulières en utilisant les tableaux **numpy**.

1) Ecrire une fonction **Puissance** qui prend en entrée une matrice carrée M et un entier strictement positif p et renvoie M^p.
Indication : Utiliser la commande **dot** du module numpy.

2) Une matrice carrée M d'ordre n est celle d'un projecteur si on a : $M^2 = M$.
Ecrire une fonction **EstProjecteur** qui prend comme paramètre une matrice carrée M et retourne *True* si M est la matrice d'un projecteur et *False* sinon.

3) On dit qu'une matrice carrée M d'ordre n est nilpotente s'il existe un entier naturel k, appelé indice de nilpotence, tel que M^k est égale à la matrice nulle d'ordre n avec $k \leq n$. L'indice de nilpotence est le plus petit $k \leq n$.

Exemple :

$$u : \begin{pmatrix} 3 & 9 & -9 \\ 2 & 0 & 0 \\ 3 & 3 & -3 \end{pmatrix} \qquad u^3 : \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

u est bien nilpotent d'indice 3.

Ecrire une fonction **IndNilpotence** qui prend comme paramètre une matrice carrée M, supposée non nulle, et retourne l'indice de nilpotence s'il existe et 0 sinon.

Nota : On peut utiliser la commande *numpy* A.all() pour tester si tous les éléments de la matrice A sont tous *True*.

Exemple :
```
>>> A = np.array([[False, True, True], [False, True, True], [True, False, True]])
>>> A.all()
False
```

4) On dit qu'une matrice carré M d'ordre n est circulante si chaque ligne d'indice i (avec $1 \leq i \leq n-1$) est la permutation circulaire vers la droite (d'une case) de la ligne d'indice $i-1$.

Exemple : $M = \begin{pmatrix} 1 & a & b & c \\ c & 1 & a & b \\ b & c & 1 & a \\ a & b & c & 1 \end{pmatrix}$ est une matrice circulante.

Ecrire une fonction **EstCirculante** qui prend comme paramètre une matrice carrée M et retourne *True* si M est circulante et *False* sinon.

5) Soit L une liste de nombres définie par : $L = [c_0, c_1, c_2, \ldots, c_{n-1}]$. On se propose de générer une matrice circulante carrée C à partir de la liste L, tel que :
- la 1ère ligne de C est égale à : $[c_0, c_1, c_2, \ldots, c_{n-1}]$.
- la 2ème ligne de C est égale à : $[c_{n-1}, c_0, c_1, \ldots, c_{n-2}]$; et ainsi de suite.

$$C = \begin{pmatrix} c_0 & c_1 & c_2 & \cdots & c_{n-1} \\ c_{n-1} & c_0 & c_1 & & c_{n-2} \\ c_{n-2} & c_{n-1} & c_0 & & c_{n-3} \\ \vdots & & & \ddots & \vdots \\ c_1 & c_2 & c_3 & \cdots & c_0 \end{pmatrix}$$

Ecrire une fonction **GenCirculante** qui prend comme paramètre une liste L et retourne une matrice circulante à partir de la liste L.

6) Sachant que le produit de deux matrices circulantes est une matrice circulante. Ecrire une fonction **PordCirculant** qui prend comme paramètre deux matrices carrées A et B de même ordre, supposées circulantes, et retourne leur produit matriciel en exploitant la propriété des matrices circulantes.