Pour le contrôle continu no 1

1. Sur 1 heure

Exercice 1.1.

1) Écrire une fonction `echange` prenant en arguments deux nombres a et b (entiers ou réels) et qui retourne trois nombres a', b' et k, avec $k \in \{0, 1\}$, tels que

- $a' = a$, $b' = b$, $k = 0$ si $a \leq b$
- $a' = b$, $b' = a$, $k = 1$ si $a > b$.

2) Écrire une fonction `unPassage` prenant en argument une liste de nombres réels et qui retourne une liste L' et un entier s, où L' est obtenue à partir de L en appliquant successivement, pour chaque paire $L'[i]$, $L'[i+1]$ la fonction `echange`, $i = 0, 1, \ldots$; la liste L' est actualisée au fur et à mesure. L'entier s représente le nombre d'échanges effectifs réalisés pendant l'exécution de la boucle.

3) En utilisant la fonction `unPassage`, écrire une fonction `ordreCroissant` prenant en argument une liste de nombres réels et qui retourne la liste ayant les même éléments que L dans l'ordre croissant.

Exercice 1.2.

1) Écrire une fonction `matricePuissances` prenant en argument une liste de nombres $L = [a_0, \ldots, a_{n-1}]$ et qui retourne la matrice (array) de taille $\text{len}(L) \times \text{len}(L)$,

$$V_L = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ a_0 & a_1 & a_2 & \cdots & a_{n-1} \\ a_0^2 & a_1^2 & a_2^2 & \cdots & a_{n-1}^2 \\ \vdots & \vdots & \vdots & & \vdots \\ a_0^{n-1} & a_1^{n-1} & a_2^{n-1} & \cdots & a_{n-1}^{n-1} \end{pmatrix}.$$

2) Calculer le déterminant de la matrice 2×2, $V_{[a,b]}$.

3) Étudier la fonction `numpy.linalg.det`. En effectuant des tests avec différents listes $[a, b, c]$, et en regardant les différences $a - b$, $a - c$ et $b - c$, conjecturer une formule pour $\det(V_{[a,b,c]})$.

4) Vérifier votre conjecture pour la matrice 4×4, V_L.

Exercice 1.3. On modélise un réseau social R_n de n individus par une matrice $n \times n$ dans laquelle la valeur 1 en ligne i et colonne j signifie que i et j sont amis alors que la valeur 0 signifie qu'ils ne sont pas amis. À titre d'exemple, la matrice

$$R_6 = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

modélise un réseau social à $n = 6$ individus dans lequel 1 est ami avec 2 et 3, 5 est ami avec 2 et 3 tandis que 4 et 6 ne sont amis avec personne. Par convention, une personne n'est pas amie avec elle-même (ce qui se traduit par des 0 dans la diagonale).

1) Créer la matrice R_6 ci-dessus à l'aide d'un tableau `numpy`. Elle vous servira à tester les fonctions suivantes.

2) Écrire une fonction `estCorrecte(R)` qui prend une matrice `R` en entrée et qui renvoie `True` si la matrice modélise bien un réseau social (symétrique avec seulement des 0 et des 1), `False` sinon.

3) Écrire une fonction `listeAmis(R, i)` qui prend une matrice `R` et une personne `i` en entrée et qui affiche la liste de ses amis.
Exemple : l'appel `listeAmis(R6, 2)` devra afficher `Amis de 2 :  [1, 5]`.

4) Écrire une fonction `listeAmis2(R, i)` qui prend une matrice `R` et une personne `i` en entrée et qui affiche la liste de ses relations d'ordre 2 (les amis de ses amis). Il faudra prendre ici quelques précautions...
Exemple : l'appel `listeAmis2(R6, 1)` devra afficher `Amis des amis de 1 :  [5]`.

5) Écrire une fonction `sansAmis(R)` qui prend une matrice `R` en entrée et qui affiche la liste des individus qui n'ont pas d'amis.
Exemple : l'appel `sansAmis(R6)` devra afficher `N'ont pas d'amis :  [4, 6]`.

Exercice 1.4. On rappelle que le *théorème fondamental de l'arithmétique* stipule que tout entier strictement positif se décompose de manière unique en un produit de nombres premiers (à l'ordre près des facteurs).

1) Écrire une fonction `estPremier(k)` prenant un entier `k` strictement positif en entrée et renvoyant `True` ou `False`, selon que `k` est premier ou non.

2) Écrire une fonction `decomposer(n)` prenant un entier `n` strictement positif en entrée et effectuant la décomposition en produits de facteurs premiers de `n`. Pour cela, on pourra suivre la logique suivante :

a) Initialiser une liste vide `ListeFacteurs`.

b) Pour tous les entiers `k` compris entre `2` et `n/2`, si `k` est premier, effectuer une boucle "tant que" `k` divise `n` dans laquelle on remplace `n` par `n/k` on ajoutera `k` à `ListeFacteurs`.

c) Renvoyer `ListeFacteurs`.

Exemple : l'appel `decomposer(382800)` devra renvoyer `[2, 2, 2, 2, 3, 5, 5, 11, 29]`.

3) Quelle est la décomposition de 6687848001285 ?

Exercice 1.5. On rappelle qu'un polynôme à coefficients entiers $P = \sum_{k=0}^{n} a_k X^k \in \mathbb{Z}[X]$ est la donnée de la liste $[a_0, \dots, a_n]$ de ses coefficients, où $a_n \neq 0$.

1) En utilisant la fonction `random.randint(a, b)`, écrire une fonction associant à un entier naturel N et à deux entiers $a < b$, un polynôme aléatoire de degré $\leq N$ à coefficients entiers compris entre a et b.

2) Écrire une fonction calculant la somme de deux tels polynômes *Indication :* La sortie doit être une liste de la bonne taille. Par exemple, pour $[1, 0, 2]$ et $[-2, 1, -2]$, la somme serait

$$(1 + 2X^2) + (-2 + X - 2X^2) = -1 + X \rightsquigarrow [-1, 0].$$

3) Écrire une fonction calculant le produit de deux polynômes à coefficients entiers. On rappelle que le produit de $P = \sum_{p=0}^{m} a_p X^p$ et $Q = \sum_{q=0}^{n} b_q X^q$ est

$$PQ = \sum_{j=0}^{m+n} \Big(\sum_{p+q=j} a_p b_q \Big) X^j.$$

4) Écrire une fonction permettant d'évaluer un polynôme P en un réel t_0 de deux façons différentes :

a) par la méthode naïve en calculant chaque puissance de t_0

b) en écrivant le polynôme $P = \sum_{j=0}^{n} a_j X^j$ comme

$$P = a_0 + X(a_1 + X(a_2 + X(a_3 + \ldots + X(a_{n-1} + a_n X)\cdots))).$$

Comparer le temps mis par ces deux méthodes pour évaluer un polynôme aléatoire de degré 10^4 en 0.5.

2. Sur 30 minutes

Exercice 2.1. Soit N un nombre entier positif.

1) Expliquer les résultats des opérations $N//10$ et $N\%10$.

2) Écrire une fonction `chiffresEnBase10` prenant en argument un entier positif N et retournant la liste $[c_0, c_1, \ldots]$ des chiffres de l'écriture de N en base 10, c'est-à-dire $N = c_k \ldots c_1 c_0$.

Exercice 2.2.

1) Écrire une fonction `plus_grande_puissance(n)` qui prend en argument un nombre entier strictement positif n et qui renvoie k, où k est le plus grand entier tel que $2^k \leq n$.

2) En déduire une fonction `binaire(n)` qui prend en argument un entier strictement positif n, et qui renvoie une liste décroissante $[k_N, \ldots, k_1]$ telle que

$$n = 2^{k_1} + 2^{k_2} + \cdots + 2^{k_N}.$$

Indication : On pourra remarquer que si k est le plus grand entier tel que $2^k \leq n$, on a $n = 2^k + r$, en notant $r = n - 2^k$. On pourra alors appliquer un raisonement itératif à r jusqu'à ce que celui-ci soit nul.

Exercice 2.3. Écrire une fonction `nombreDApparitions(s, t)` qui prend en arguments deux chaînes de caractères s et t, et qui retourne le nombre d'apparitions de s dans t. Faire tourner la procédure pour "n" ou "nn" pour le premier argument, et "Le spectacle du monde ressemble a celui des jeux Olympiques: les uns y tiennent boutique; d'autres paient de leur personnes; d'autres se contentent de regarder. (Pythagore)" pour le deuxième argument.

Exercice 2.4. Écrire une fonction `crypter(s, N)` qui prend en argument une chaînes de caractères s et un entier $1 \leq N \leq 25$. On considère que la chaîne s est formée exclusivement de lettres minuscules. La fonction retourne s crypté de la manière suivante :

- On considère la liste de longeur 26,

$$A = [\text{"}a\text{"}, \text{"}b\text{"}, \text{"}c\text{"}, \text{"}d\text{"}, \text{"}e\text{"}, \text{"}f\text{"}, \ldots, \text{"}v\text{"}, \text{"}w\text{"}, \text{"}x\text{"}, \text{"}y\text{"}, \text{"}z\text{"}]$$

- Pour chaque symbole α de s, on prend son indice i dans la liste A, c'est-à-dire $\alpha = A[i]$, et on le remplace par le symbole $A[(i + N)\%26]$.

1) Que retourne `crypter("obscurcissement", 8)` ?

2) Écrire une fonction `decrypter(s, N)` qui réalise le décryptage de la chaîne produite par la fonction `crypter(s, N)`. Testez-la sur le résultat du point précédent.

3. Sur 30 minutes

Exercice 3.1. Soit a est une constante réelle, $a \in \mathbb{R}$. Soit la courbe $\mathcal{C}_a$ paramétrée par les équations

$$\mathcal{C}_a : \begin{cases} x(t) = a\Big(\cos t + \sqrt{8}\,\cos\dfrac{t}{2}\Big) \\ y(t) = a\sin t, \end{cases}$$

où $0 \leq t \leq 4\pi$.

1) Écrire une fonction `nemo(a)` proposant une représentation graphique de la courbe $\mathcal{C}_a$, de couleur orange, une épaisseur de ligne de 3, des noms aux axes et un titre de la forme `Courbe C pour a=...` à compléter avec l'argument d'entrée.

2) Tester votre programme avec, par exemple, `nemo(2)` et `nemo(-1)`.

3) Pouvez-vous, pour `nemo(2)`, colorier la "queue" du poisson en bleu ?

Exercice 3.2.

1) Écrire une fonction `npi(n)` associant à un entier $n \in \mathbb{N}$ le nombre de nombres premiers inférieurs ou égaux à n.

2) Écrire une fonction `distribution(a)` qui prend en argument un réel ≥ 0 et qui dessine le graphe de la fonction $p : [0, a] \to \mathbb{N}$ définie par

$$x \longmapsto \text{card}(\{\text{nombres premiers} \leq x\}) \in \mathbb{N}.$$

On veut que le graphe soit de couleur bleu, que les axes de coordonnées soient nommés, et que le titre soit *Distribution des nombres premiers inférieurs à a.*

Exercice 3.3. On considère la courbe paramétrée définie par

$$(x(t), y(t)) = \left(1 + \cos\frac{pt}{q}\right)(\cos t, \sin t) \tag{3.1}$$

avec p et q des entiers et $0 \leq t < 2q\pi$.

1) Écrire une fonction `rosace(t, p, q)` qui prend en arguments une valeur du paramètre et les entiers p et q, et qui retourne le vecteur $(x(t), y(t))$ défini dans (3.1).

2) Écrire une fonction `dessineRosace(p, q, N)` qui prend en arguments les entiers p et q ainsi que le nombre de points utilisés pour dessiner la courbe, et qui a comme effet collatéral la représentation graphique de la courbe paramétrée (3.1). On voudrais que la couleur utilisée soit le vert, que l'épaisseur du trait soit de 1 et que la fenêtre graphique porte le titre

Rosace pour $p = \ldots$ et $q = \ldots$

en remplaçant les $[\ldots]$ par les valeurs de p et q.

3) Après avoir fait plusieurs tests, pour comprendre la signification du paramétré dans l'image de la courbe, écrire une fonction `dessineMorceauDeRosace(tini, tfinal, p, q, N)` qui prend en arguments les valeurs initiale et finale du paramètre, les entiers p et q ainsi que N, le nombre de points utilisés pour dessiner le morceau de courbe, et qui construit en rouge avec épaisseur 3, ce morceau de la courbe paramétrée (3.1) (c'est-à-dire pour $t_{ini} \leq t \leq t_{final}$).

4) Dessiner la rosace (3.1) quand $p = 4$ et $q = 3$. De loin, elle ressembla à une réunion de quatre "cercles". Colorier en rouge le "cercle" de droite.

Exercice 3.4.

1) Écrire une fonction `dessineCercle(nbPoints)` qui trace en rouge le cercle de centre $(0, 0)$ et de rayon 1 en utilisant `nbPoints` points.

2) Soient les N points

$$P_k = \left(\cos \frac{2k\pi}{N}, \sin \frac{2k\pi}{N} \right), \qquad k = 0, 1, \ldots, N-1.$$

a) Écrire une fonction `dessinePolygoneRegulier(N)` qui trace en bleu le polygone régulier $[P_0 P_1 \cdots P_{N-1}]$.

b) Écrire une fonction `dessineDiagonales(j, N)` qui prend en argument un entier j et un entier N, avec $0 \leq j \leq N-1$, et qui trace les $N-3$ diagonales du polygone régulier $[P_0 P_1 \cdots P_{N-1}]$ issues de P_j.