

DEVOIR DE SYNTHESE SEMESTRE 2

Matière : INFORMATIQUE

Classes : MP 1, PC 1, PT 1 – Durée : 2 h

Les fonctions demandées doivent être écrites en langage **Python**. On sera très attentif à la rédaction et notamment à l'indentation du code.

Exercice 1 (4.5 pts)

1) Soit la fonction récursive **Guess(n, m)** suivante, donner la trace d'exécution manuelle des appels récursifs pour déterminer la valeur renvoyée suite à l'appel **Guess(5, 3)**.

```
def Guess(n, m) :  
    if m > n : return 0  
    if m == n : return 1  
    if n == 1 or m == 0 : return 1  
    if m < n : return (n-1)* Guess(n-1, m-1)
```

2) On se propose de vérifier l'équation mathématique suivante, où n est le nombre de répétitions des chiffres :

$$\underbrace{111 \dots 1}_{2n \text{ fois}} = \underbrace{222 \dots 2}_{n \text{ fois}} + \underbrace{(333 \dots 3)^2}_{n \text{ fois}}$$

Exemple :

Pour $n = 1$, l'équation est vraie : $11 = 2 + 3^2$.

Pour $n = 2$, l'équation est aussi vraie : $1111 = 22 + 33^2$.

Ecrire une fonction **Verification(p)** qui reçoit comme paramètre un entier p et renvoie **True** si l'équation est vraie pour tout $n \leq p$ et **False** sinon.

3) Le théorème des quatre carrés de *Lagrange* montre que tout entier positif n peut être représenté comme la somme de quatre carrés d'entiers. Donc, il existe des entiers a, b, c et d tels que : $n = a^2 + b^2 + c^2 + d^2$ avec $0 \leq a \leq b \leq c \leq d \leq \lfloor \sqrt{n} \rfloor$, où $\lfloor \cdot \rfloor$ désigne la partie entière.

Exemple :

Pour $n = 25$, on a trois manières de représentation de Lagrange :

$$25 = 0^2 + 0^2 + 0^2 + 5^2$$

$$25 = 0^2 + 0^2 + 3^2 + 4^2$$

$$25 = 1^2 + 2^2 + 2^2 + 4^2$$

Ecrire une fonction **Lagrange(n)** qui reçoit un entier positif n et retourne une liste des différentes manières de représentation de Lagrange. Elle est composée de tuples dont chacun est formé des quatre entiers a, b, c et d .

L'appel de la fonction **Lagrange(25)** renvoie le résultat suivant :
 $[(0, 0, 0, 5), (0, 0, 3, 4), (1, 2, 2, 4)]$

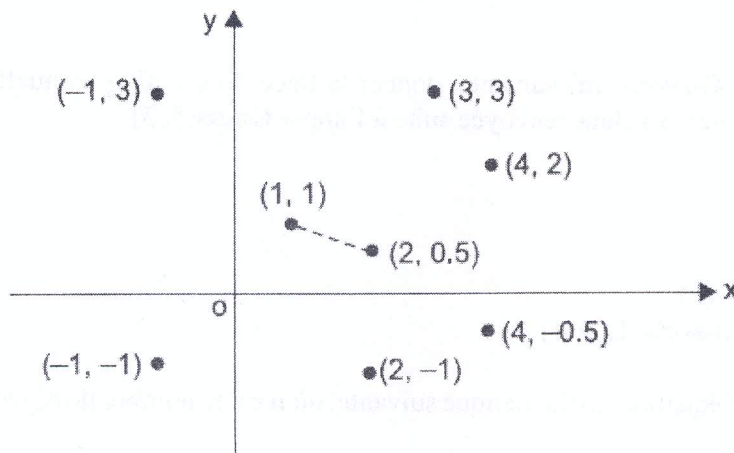
Indication : On peut utiliser des boucles imbriquées.

Exercice 2 (5 pts)

On dispose d'un ensemble de n points dans un plan de coordonnées (x_i, y_i) enregistrées dans un fichier texte **points.txt** composé de n lignes dont chacune représente l'abscisse et l'ordonnée d'un point séparées par une tabulation ($\backslash t$).

Exemple :

Soit un ensemble de 8 points avec le fichier texte **points.txt** correspondant :



Représentation des 8 points dans le plan

-1	3
-1	-1
1	1
3	3
4	2
2	0.5
4	-0.5
2	-1

points.txt

On cherche les deux points les plus proches c'est-à-dire qui représentent la distance euclidienne minimale. Dans l'exemple ci-dessus, les points de coordonnées $(1, 1)$ et $(2, 0.5)$ sont les plus proches.

La distance euclidienne entre deux points de coordonnées (x_1, y_1) et (x_2, y_2) est calculée par la formule suivante :

$$distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

1) Ecrire une fonction **Distance(x1, y1, x2, y2)** qui reçoit les coordonnées de deux points et retourne la distance euclidienne qui les sépare.

2) Ecrire une fonction **Valeurs(nom)** qui reçoit **nom** comme fichier texte et renvoie une liste composée de sous listes dont chacune représente les coordonnées d'un point.

Pour l'exemple ci-dessus, l'appel de **Valeurs('points.txt')** retourne la liste suivante :
 $[[-1.0, 3.0], [-1.0, -1.0], [1.0, 1.0], [3.0, 3.0], [4.0, 2.0], [2.0, 0.5], [4.0, -0.5], [2.0, -1.0]]$

3) Ecrire une fonction **Proches(P)** qui reçoit la liste **P** des coordonnées des points et qui renvoie un tuple composé de deux sous tuples qui représentent les coordonnées des points les plus proches. Pour l'exemple, l'appel de **Proches(P)** donne : $((1.0, 1.0), (2.0, 0.5))$

Exercice 3 (6 pts)

Soit f une fonction à variable réelle x continue et dérivable sur l'intervalle $[a, b]$. On se propose de calculer numériquement les extrémums de la fonction f (maximum local ou minimum local). La condition suffisante pour un extrémum local est :

- Si f est dérivable sur $[a, b]$ et si, en un point $\alpha \in [a, b]$, la dérivée de f s'annule en changeant de signe, alors f atteint un extrémum local en α .

L'approche consiste à découper l'intervalle $[a, b]$ avec un petit pas $p = 0.001$ et calculer la dérivée de f en tous points x , tel que $x = a + k.p$; si $f'(x).f'(x+p) < 0$ alors le point d'abscisse x est un extrémum.

En choisissant un petit nombre $h = 10^{-6}$, qui représente une petite variation autour de x , la dérivée approximative $f'(x)$ de la fonction $f(x)$ est obtenue par la relation suivante :

$$f'(x) \cong \frac{f(x+h) - f(x)}{h} \quad (1)$$

1) Ecrire une fonction **Fprime(f, x)** qui reçoit en paramètres une fonction f et un réel x et renvoie $f'(x)$ en utilisant la relation (1).

2) Ecrire une fonction **Extremums(f, a, b, p = 0.001)** qui reçoit en paramètres une fonction f , les réels a, b de l'intervalle d'étude et le pas p (défini par défaut à 10^{-3}) et qui retourne une liste de tuples dont chacun représente l'abscisse et l'ordonnée d'un extrémum.

Soit la fonction f à variable réelle x définie par :

$$f(x) = \sin(x - \sin(x))e^{-x^2} \quad (2)$$

3) Ecrire une fonction **f(x)** qui reçoit x comme paramètre et retourne la valeur de la fonction $f(x)$ de l'équation (2).

4) Ecrire une fonction **Graphes(f, a, b)** qui reçoit comme paramètres une fonction f et deux réels a, b et permet de tracer les deux courbes de la fonction $f(x)$ ainsi que sa dérivée $f'(x)$ (en utilisant l'approximation de la relation (1)) sur l'intervalle $[a, b]$ en 200 points régulièrement espacés.

Exercice 4 (4.5 pts)

Pour $n, p \in \mathbb{N}^*$ et $x = (x_1, x_2, \dots, x_p)$ un vecteur de taille p , la matrice de *Vandermonde* correspondante est définie par :

$$V(x, n) = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_{p-1} & x_{p-1}^2 & \cdots & x_{p-1}^{n-1} & x_{p-1}^n \\ 1 & x_p & x_p^2 & \cdots & x_p^{n-1} & x_p^n \end{pmatrix}$$

Nota : le vecteur x ainsi que la matrice de Vandermonde $V(x, n)$ sont des tableaux de type *numpy*.

1) Ecrire une fonction **Vander_1(x, n)** qui reçoit un vecteur x et un entier n et retourne la matrice $V(x, n)$ en utilisant deux boucles.

2) Après avoir établi une relation pour exprimer la $k^{\text{ème}}$ colonne de la matrice $V(x, n)$ en fonction de x et k , écrire une deuxième fonction **Vander_2(x, n)** qui construit la matrice colonne par colonne en utilisant cette relation, et ce avec une seule boucle.

3) Après avoir établi une relation entre la $k^{\text{ème}}$ colonne, la $(k - 1)^{\text{ème}}$ colonne de la matrice $V(x, n)$ et le vecteur x , écrire une troisième fonction **Vander_3(x, n)** qui construit la matrice colonne par colonne en utilisant cette relation, et ce avec une seule boucle.