TP Info 1 : révisions

Exercice 1 : Manipulations de chaînes de caractères

1. Écrire une fonction *recherche* qui prend en argument une lettre a et une chaîne de caractères A et qui cherche si a apparait dans A.
2. Écrire une fonction *occurence* qui prend en argument une lettre a et une chaîne de caractères A et qui compte le nombre d'occurrence de a dans A.
3. Écrire une fonction *position* qui prend en argument une lettre a et une chaîne de caractères A et qui renvoie la liste des indices pour lesquels a apparait (la liste est vide si a n'apparaît pas).
4. Écrire une fonction *cherche_mot* qui prend en argument une chaîne de caractères *mot* et une chaîne de caractères A et qui cherche si *mot* apparait dans A.

Exercice 2 : Manipulations de listes

1. Écrire une fonction *maxi* qui prend en argument une liste de nombres et renvoie une liste contenant la valeur maximale et sa première occurrence dans la liste.
2. Écrire une fonction *moyenne* qui prend en argument une liste de nombres et renvoie la moyenne de cette liste.
3. Écrire une fonction *tri_selection* qui prend en argument une liste de nombres et qui renvoie la liste triée par ordre croissant avec la méthode du tri par sélection (on crée une nouvelle liste en ajoutant les nombres du plus petits au plus grand).
4. Écrire une fonction *tri_insertion* qui prend en argument une liste de nombres et qui renvoie la liste triée par ordre croissant avec la méthode du tri par insertion (on crée une nouvelle liste en insérant les nombres les uns après les autres à leur place).
5. Écrire une fonction *tri_bulle* qui prend en argument une liste de nombres et qui renvoie la liste triée par ordre croissant avec la méthode du tri par bulle (on parcourt la liste de gauche à droite en échangeant un nombre tant qu'il est plus grands que le suivant).
6. Écrire une fonction *tri_rapide* qui prend en argument une liste de nombres et qui renvoie la liste triée par ordre croissant avec la méthode du tri rapide (cette méthode est récursive, c'est à dire qu'elle fait appel à elle même : une liste vide ou contenant un nombre est triée. Si elle contient plus d'un nombre, on choisit un pivot. On crée deux listes qui contiennent les éléments plus petits que le pivot d'une part et plus grands que le pivots d'autre part et on trie ces listes avant de les concaténer).
7. Écrire une fonction *mdiane* qui détermine la médiane d'une liste de nombres quelconque.

Exercice 2 : Approximation d'un zéro

1. (méthode de la dichotomie)
   (a) rappeler le principe de la méthode de dichotomie.
   (b) Écrire un programme qui prend en argument une fonction f, les bornes a et b d'un intervalle, une erreur e et qui renvoie la valeur approchée d'un zero de f sur $[a;b]$ à e près.

(c) Utiliser cette fonction pour déterminer une valeur approchée de la solution de l'équation $x + \ln(1+x) - 1 = 0$ sur $[0;1]$.

2. (Méthode de Newton)Pour déterminer le zero d'une fonction f, on remplace la fonction f par la fonction tangente à la courbe de f en une des bornes de l'intervalle. l'intersection avec l'axe des abscisses est une valeur approchée du zéro de f et on réitère le procédé sur le nouvel intervalle ainsi défini.
   (a) Réaliser sur papier une représentation graphique de cette méthode pour une fonction f continue croissante et concave sur $[a;b]$, en partant du point d'abscisse a.
   (b) Dans le cas général, pour la première étape, Justifier que le point c approchant le zéro de f vérifie $c = a - \frac{f(a)}{f'(a)}$.
   (c) Programmer une fonction *Newton* qui prend en argument une fonction f croissante et convexe, les bornes a et b de l'intervalle et une erreur e et qui renvoie une valeur approchée du zero de f à e près.
   (d) Utiliser la fonction *Newton* pour trouver le zéro de la fonction du (1.c).
3. Utiliser le module time et la fonction time.clock() pour comparer le temps d'exécution des deux programmes précédents.

Exercice 3 : Approximation d'une intégrale

1. Rappeler la formule de Riemann permettant de calculer la valeur approchée d'une intégrale.
2. Écrire une fonction *approx_rectangle* qui prend en argument deux réels a et b, une fonction f continue sur $[a;b]$ et qui renvoie une valeur approchée de l'intégrale de f sur $[a;b]$.
3. Modifier le programme précédent pour créer une fonction *approx_trapeze* qui donne une valeur approchée de l'intégrale de f sur $[a;b]$ en utilisant un découpage de l'aire en trapèzes plutôt qu'en rectangles.
4. Comparer les deux méthodes.

Exercice 4 : Résolution d'équations différentielles

On se donne une équation différentielle linéaire d'ordre 1 du type $f' + gf = h$ d'inconnue f dérivable sur $I = [a,b]$, avec g et h deux fonctions continues sur l'intervalle I. On se donne également une condition initiale $f(a) = f_0$.

La méthode de résolution numérique consiste à diviser l'intervalle I en n intervalles de longueur $pas = \frac{b-a}{n}$. On note $a_k = a + k * pas, \forall k \in [0;n]$ et on note f_k la valeur approchée de $f(a_k), \forall k \in [0;n]$.

On calcule de proche en proche les valeurs f_k en approchant la courbe de la solution par la tangente à cette courbe au point calculée précédemment.

1. Rappeler l'équation de la tangente à la courbe en un point a_k. L'exprimer uniquement en fonction de f, g, h, a_k.
2. En déduire la relation de récurrence entre f_k et f_{k+1} pour tout entier naturel k.

3. Écrire une fonction *Euler* qui prend en argument les fonctions g, h, les réels a, b, la donnée initiale f_0 et le nombres n d'intervalles et qui renvoie la liste des valeurs de la solution approchées.
4. Écrire un programme qui trace sur un même graphique la solution exacte et la solution approchée de l'équation $y' + 3y = 4$ sur l'intervalle $[0; 5]$ avec pour condition initiale $y(0) = 2$.

Exercice 5 : Modifier une image en nuance de gris

1. Négatif : Faire un programme qui charge une image en noir et blanc, puis en fait un négatif.
2. Éclaircir une image : Faire une fonction qui demande le nombre d'unités à ajouter à chaque pixel pour éclaircir l'image(par exemple 150), et ajoute ce nombre à chaque pixel. Il faudra faire attention à ne pas dépasser 255, qui est la valeur maximale (blanc) ; pour cela, on pourra utiliser la fonction min(a,b) qui renvoie le minimum des deux nombres a et b.
3. Accentuation de contraste : Écrire une fonction qui modifie chaque pixel de façon à multiplier l'écart entre la valeur du pixel et la valeur moyenne 127 par une valeur précisée en argument.
4. Seuillage : Écrire une fonction qui prend en argument un seuil (nombre entre 0 et 255), puis qui fait une nouvelle image de la même taille, avec pour chaque pixel la valeur 255 si le pixel de l'image initiale est supérieur ou égal au seuil, et 0 dans le cas contraire.
5. Détection de bordure : Écrire une fonction qui met dans la nouvelle image un pixel à 255 si l'écart-type des 4points voisins du pixel de l'image initiale (haut,bas, gauche, droite) est supérieur à un seuil, par exemple 10. Pour cela, on pourra utiliser numpy.std qui renvoie l'écart-type d'une liste passée en paramètre.
6. Floutage : Ecrire une fonctionqui remplace chaque pixel par la moyenne des 5 points utilisés dans la question précédente.