2TSI. TD PYTHON 20. INDICATIONS

MESURE DE HOULE

Partie I. Stockage interne des données

Q01 Un Hertz est la mesure de la fréquence de répetition d'un événement qui se répète une fois par seconde. L'unité est s^{-1}. On rappelle aussi qu'un octet vaut 8 bits. On pourra mettre le résultat en Ko.

Q02 On rappelle que l'on fait deux enregistrements par heure. On pourra mettre le résultat en Mo.

Q03. Attention, chaque ligne a 8 caractères (dont celui invisible de fin de ligne). Ainsi si l'on enlève un chiffre significatif, $+0.4256$ devient $+0.425$

Q04. **Question hors programme à ne pas faire**. Vous aurez la correction pour les curieux.

Partie II. Analyse vague par vague

Q05 Il suffit de lire sur le graphique de la figure 2.

Q06 Il s'agit de faire une procédure PYTHON appelée *moyenne* et donc on tape :
$def\ moyenne(liste\_niveaux)$:
Puis, on somme les éléments $liste\_niveaux[i]$ et on $return$ leur moyenne arithmétique.

Q07 Rappelons la mèthode des trapèzes. Si l'intervalle $[a,b]$ est découpé en segments $[a_0,a_1]$, $[a_1,a_2]$, .., $[a_{n-1},a_n]$ de même longueur h avec $a_0=a$ et $a_n=b$ alors

$$\int_a^b f(t)\,dt \simeq \sum_{i=0}^{n-1}\left(\frac{f(a_i)+f(a_{i+1})}{2}\right)h.$$

Si L est la liste stockant les valeurs successives de f alors, pour tout indice i, $L[i]=f(a_i)$. Comme l'échantillonnage est de 2 Hz, la longueur de chaque intervalle est de $\frac{1}{2}$ seconde d'où le code. C'est à vous.
On tapera au départ :
$def\ integrale\_precise(liste\_niveaux)$:
Rappelons ensuite que la valeur moyenne d'une fonction f sur un intervalle $[a,b]$ est le nombre

$$\frac{1}{b-a}\int_a^b f.$$

Comme ici la longueur de l'intervalle est de 20 mn, c'est a vous pour le code.

Q08 On part de
$def\ ind\_premier\_pzd(liste\_niveaux)$:
Nous allons parcourir tous les éléments de la liste $liste\_niveaux$ et trouver le premier indice i tel que $liste\_niveaux[i]$ est supérieur à la moyenne $m = moyenne\_precise(liste\_niveaux)$ de la liste et l'indice suivant ne l'est pas. Si aucun tel indice n'existe, la boucle s'achève et on revoit -1.
Vous pouvez user de la boucle while ou d'une boucle for et les deux versions seront dans le corrigé.

Q09 Il suffit d'utiliser un code voisin du code précédent en utilisant une boucle descendante du type :
$for\ i\ in\ range(n-1,0,-1)$:
puis on teste si $liste\_niveaux[i] > m$ et $liste\_niveaux[i-1] > m$. Si c'est le cas, on $return\ i$ sinon on $return\ -2$.

Q10 Cette fonction est juste une modification de la fonction *ind_premier_pzd* (de la question 8). Au lieu de retourner le premier indice, on ajoute chaque indice convenable à la liste (ou plus précisèment, son successeur strict comme le demande l'énoncé). Voici la fonction correspondante.

T.S.V.P →

Q11 Pour cela, on construit les successeurs (stricts) de chaque PND.
On tape :
$succ = construction\_successeurs(liste\_niveaux)$
Puis on initialise $vagues = [\,]$ au départ.
On travaille dans une boucle :
$for\ i\ in\ range(len(succ) - 1)$:
Si Z_i et Z_{i+1} sont deux successeurs consécutifs, il crée la vague allant de l'indice Z_i (inclus) à l'indice Z_{i+1} (non inclus). Rappelons que si L est une liste et i, j deux entiers avec $i < j$ alors $L\,[i : j]$ est la liste formée des éléments de L allant de l'indice i à l'indice j (non inclus).
Puis on nourrit *vagues* et quand *vagues* est bien remplie, on la *return*.

Q12 On tape d'abord :
$def\ proprietes(listes\_niveaux)$:
On utilise la fonction décomposant en vague, en tapant :
$vagues = decompose\_vagues(liste\_niveaux)$
On affecte $Z1$ avec $ind\_premier\_pzd(liste\_niveaux)$.
Il faut distinguer le premier intervalle $[0, Z_1]$ des autres. Nous utilisons la commande $L\,[0 : Z_1 + 1]$ pour obtenir tous les éléments ments de L situés entre l'indice 0 et l'indice Z_1 (inclus). On affecte alors $H1$ puis $T1$. Enfin, dans *prop*, on rentre$[[H1, T1]]$.
On déroule ensuite une boucle *for* pour calculer les différentes hauteurs et temps (qu'il faut multiplier par 0.5 car le pas de temps est d'une demi-seconde, fréquence de 2 Hz). On affecte dans la boucle les H_i et les T_i. Puis, on nourrit *prop* et on le *return*.

Partie III. Contrôle des données

Q13 On crée la liste des propriétés *prop* associées par la commande $proprietes(liste\_niveaux)$.
Puis, en nommant $Hmax$ la fonction cherchée, on tape :
$def\ Hmax(liste\_niveaux)$:
Par les listes en compréhension, on crée la liste des hauteurs associé en extrayant de chaque élément p de la liste *prop* sa première composante *p[0]* (qui donne la hauteur de la vague).
Ainsi, la syntaxe $L[i]\ for\ L\ in\ prop$ extrait de chaque élément L de la liste *prop* sa composante $L[i]$
Pour finir, on applique la fonction *max* sur cette liste (puisque le sujet l'autorise à la question précédente).

Q14 Dans le tri rapide, on peut choisir tout élément de la liste *liste* comme pivot. Par exemple, celui d'indice g ou bien d'indice d ou tout autre de votre choix (même au hasard entre g et d). Choisir en n'oubliant pas que *liste[i]* est une liste à deux éléments et c'est le premier qui contient la hauteur de la vague.

Q15 Si l'on appelle *triRapide_bis* la modification et on conserve comme nom *triRapide* à la fonction proposée par l'énoncé, on tape au départ :
$def\ triRapide\_bis(liste, g, d)$:
On fait alors appel à $tri\_insertion(liste, g, d)$ si $d - g < 15$ et sinon, on fait appel à $triRapide(liste, g,)$
À finir en code Python.

Q16 Voir votre cours sur le Tri par insertion.
Si vous ne voyez toujours pas, dans le tri insertion d'une liste L, on se place en un indice $j = i$. On teste si $L\,[j] < L\,[j-1]$. Si c'est le cas, on échange $L\,[j]$ et $L\,[j-1]$ puis on recommence avec l'indice $j = i - 1$. On poursuit ainsi jusqu'à l'indice minimal (ici g). On commence par $i = g$ puis on arrête lorsque $i = d$ d'où le code suivant (en tenant compte que les éléments de la liste sont ici des listes que l'on tri selon la première composante, correspondant à la hauteur de vague).