Conseils :

- Prenez le temps de bien faire les choses et **rendez une copie propre**. Des points seront mis pour le soin (environ 1,5 point sur 20).
- Aucun ordre n'est imposé pour la résolution, par contre, rendez les problèmes dans l'ordre. Numérotez avec rigueur les questions que vous traitez.
- L'usage des **calculatrices est interdit**.

Représentation des nombres

On donne les puissances de deux suivantes :

2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
2048	1024	512	256	128	64	32	16	8	4	2	1	0.5	0.25	0.125

On rappelle les règles suivantes pour un nombre flottant codé sur 64 bits :

- l'exposant e est codé sur 11 bits et est représenté en base 2 par l'entier n = e + 1023 ;
- la mantisse est codée sur 52 bits.

On considère des flottants 64 bits. Si un grand nombre de 0 apparait, il sera possible de ne pas tous les écrire mais d'indiquer à l'aide d'une accolade le nombre de 0.

Q1. Que représente le nombre :
1 10000000011 0001100000000000000000000000000000000000000000000000

Q2. Quelle est la représentation du nombre 20,125 ?

Q3. Parmi les cas suivants, quels sont ceux pour lesquels l'opération d'addition se fait sans arrondis et pourquoi :

Cas n°	*1*	*2*	*3*	*4*	*5*	*6*
x	2^{25}	2^{25}	2^{25}	2^{25}	2^{-25}	2^{-25}
y	2^{8}	2^{-20}	2^{-26}	2^{-30}	2^{-65}	2^{-125}

Mesures de houle

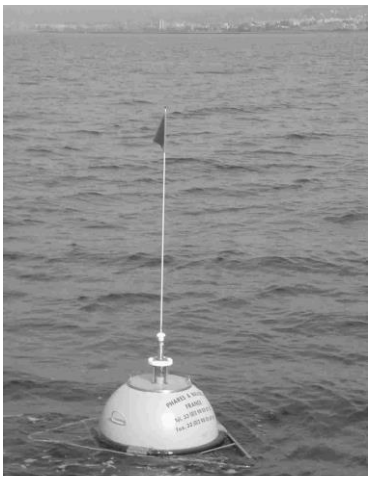


Figure 1 : bouée de mesure de houle

On s'intéresse à des mesures de niveau de la surface libre de la mer effectuées par une bouée (représentée sur la Figure 1).
Cette bouée contient un ensemble de capteurs incluant un accéléromètre vertical qui fournit, après un traitement approprié, des mesures à étudier.
Les mesures réalisées à bord de la bouée sont envoyées par liaison radio à une station à terre où elles sont enregistrées, contrôlées et diffusées pour servir à des études scientifiques. Pour plus de sûreté, les mesures sont aussi enregistrées sur une carte mémoire interne à la bouée.

Partie I. Stockage interne des données

Une campagne de mesures a été effectuée. Les caractéristiques de cette campagne sont les suivantes :

- durée de la campagne : 15 jours ;
- durée d'enregistrement : 20 min toutes les demi-heures ;
- fréquence d'échantillonnage : 2 Hz.

Les relevés de la campagne de mesure sont écrits dans un fichier texte dont le contenu est défini comme suit.
Les informations relatives à la campagne sont rassemblées sur la première ligne du fichier, séparées par des points-virgules (" ;"). On y indique différentes informations importantes comme le numéro de la campagne, le nom du site, le type du capteur, la latitude et la longitude de la bouée, la date et l'heure de la séquence.

Les lignes suivantes contiennent les mesures du déplacement vertical (m). Chaque ligne comporte 8 caractères dont le caractère de fin de ligne. Par exemple, on trouvera dans le fichier texte les 3 lignes suivantes :

```
+0.4256
+0.3174
-0.0825
...
```

Q4. On suppose que chaque caractère est codé sur 8 bits. En ne tenant pas compte de la première ligne, déterminer le nombre d'octets correspondant à 20 minutes d'enregistrement à la fréquence d'échantillonnage de 2 Hz.

Q5. En déduire le nombre approximatif (un ordre de grandeur suffira) d'octets contenus dans le fichier correspondant à la campagne de mesures définie précédemment. Une carte mémoire de 1 Go est-elle suffisante ?

Q6. Si, dans un souci de réduction de la taille du fichier, on souhaitait ôter un chiffre significatif dans les mesures, quel gain relatif d'espace mémoire obtiendrait-on ?

Deux analyses sont effectuées sur les mesures : l'une est appelée "vague par vague", l'autre est appelée "spectrale". Dans la suite du sujet, nous allons nous intéresser à l'analyse "vague par vague".

Partie II. Analyse vague par vague

On considère ici que la mesure de houle est représentée par un signal en temps discret. On appelle horodate (figure 2), un ensemble (fini) des mesures réalisées sur une période de 20 minutes à une fréquence d'échantillonnage de 2 Hz. Les informations de niveau de surface libre d'un horodate sont stockées dans une liste de flottants liste_niveaux. On suppose qu'aucun des éléments de cette liste n'est égal à la moyenne.

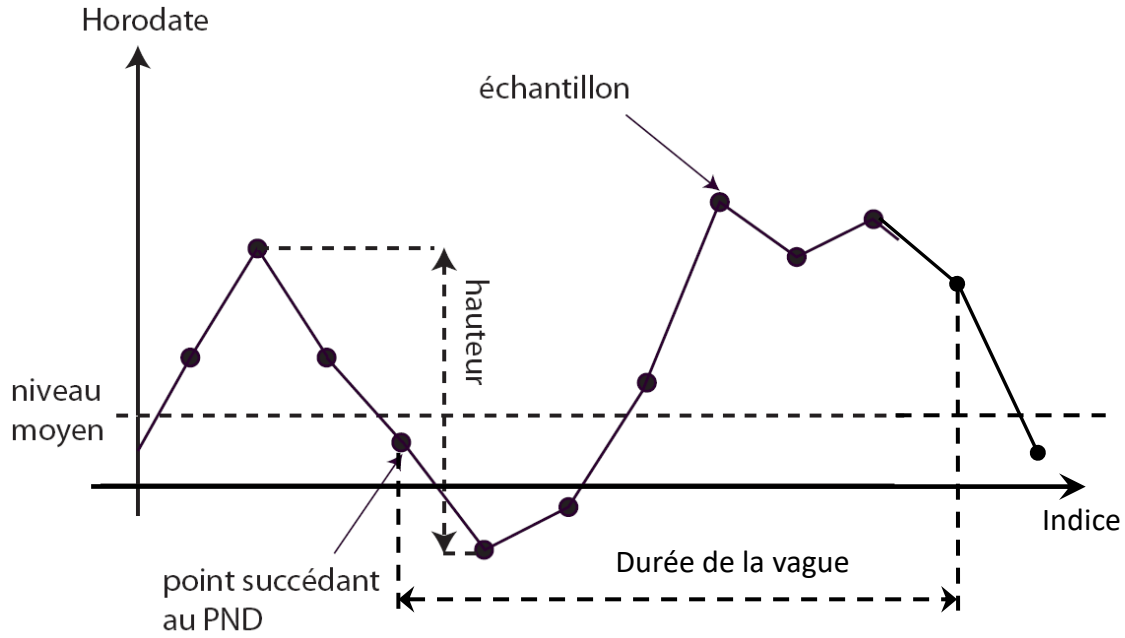


Figure 2 : propriété d'une vague ; notion de PND ⇨ Point de Niveau moyen en Descente

Q7. Proposer une fonction **moyenne** *prenant en argument une liste non vide liste_niveaux, et renvoyant sa valeur moyenne.*

Q8. Proposer une fonction **ind_premier_pzd(liste_niveaux)** *renvoyant, s'il existe, l'indice du premier élément de la liste tel que cet élément soit supérieur à la moyenne et l'élément suivant soit inférieur à la moyenne. Cette fonction devra retourner -1 si aucun élément vérifiant cette condition n'existe.*

Q9. Proposer une fonction **ind_dernier_pzd(liste_niveaux)** *renvoyant l'indice i du dernier élément de la liste tel que cet élément soit supérieur à la moyenne et l'élément suivant soit inférieur à la moyenne. Cette fonction devra retourner -2 si aucun élément vérifiant cette condition n'existe. On cherchera à proposer une fonction de complexité O(1) dans le meilleur des cas : c'est-à-dire dont le temps d'exécution ne dépend pas de la taille totale de la liste si le point recherché est proche de la fin de la liste.*

On souhaite stocker dans une liste ***successeurs***, les indices des points succédant (strictement) aux PND (voir Figure 2). Ainsi pour chaque PND, on veut seulement l'indice du premier « successeur ».
N.B : pour les questions suivantes, on fait l'hypothèse que nos fonctions peuvent toujours s'appliquer aux listes mises en arguments.

Q10. On propose la fonction **construction_successeurs** *(algorithme 1). Elle renvoie la liste successeurs. Compléter (sur la copie) les lignes 6 et 7.*

Algorithme 1

```
1 def construction_successeurs(liste_niveaux):
2     n = len(liste_niveaux)
3     successeurs = [ ]
4     m = moyenne(liste_niveau)
5     for i in range(n - 1):
6         if      # à compléter
7                 # à compléter
8     return succeseurs
```

Q11. Proposer une fonction **decompose_vagues(liste_niveaux)** *qui permet de décomposer une liste de niveaux en liste de vagues (liste constituée de niveau entre le point succédant le PND et le point précédent le PND suivant). On omettra les données précédant le premier PND et celles succédant au dernier PND. Ainsi* **decompose_vagues([1, -1, -2, 2, -2, -1, 6, 4, -2, -5])** *(Noter que cette liste est de moyenne nulle) renvoie* **[[-1, -2, 2], [-2, -1, 6, 4]]**.

On désire maintenant caractériser les vagues.
Ainsi, on cherche à concevoir une fonction **proprietes(liste_niveaux)** renvoyant une liste de listes à deux éléments [Hi,Ti] permettant de caractériser chacune des vagues i par ses attributs :

- Hi, sa hauteur en mètres (m) (voir Figure 2),
- Ti, durée de la vague en secondes (s).

Q12. Proposer une fonction **proprietes(liste_niveaux)** *réalisant cet objectif. On pourra d'abord créer une* **fonction max_liste (liste)** *qui renvoie la valeur maximale d'une liste. Vous pourrez ensuite utiliser la fonction de Python* **min(L)** *qui renvoie le minimum d'une liste L.*

On suppose stocker sous forme d'une liste les niveaux de la mer sous forme d'entier sur l'ensemble de la journée.
L'objectif est de savoir si un certain niveau de mer a été atteint dans la journée. Pour cela, il faut :

- Trier les données stockées sous forme de liste d'entier
- Rechercher dans la liste triée un entier donner

Q13. Proposer une fonction **trie_a_bulle(liste)** *qui prend en argument une liste et qui renvoie cette liste triée. Vous appliquerez un algorithme de type tri à bulles.*

Q14. Proposer une fonction **recherche_dicho(x , liste)** *qui prend en argument* ***un entier x*** *et* ***une liste L*** *et qui renvoie* ***True*** *si x est dans la liste et* ***False*** *si l'entier n'est pas dans la liste. Votre fonction doit appliquer un algorithme dichotomique.*

Un peu de récursivité

Voyons des étoiles

Soit la fonction définie sous python ci-contre :

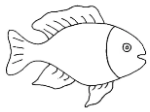
```
def affiche(n):
    if n == 0:
        return
    print(n*'*')
    affiche(n-1)
```

Q15. Qu'est-ce qui s'afficherait dans une console python si on écrit l'instruction **affiche(5)** *, justifier ?*

Q16. Ecrire une autre fonction **affiche2(n)** *qui prend en argument un entier* **n** *qui ne renvoie rien, mais qui affiche dans une console python lors de l'appel de la fonction :* **affiche2(5)** *:*

```
In [5]: affiche2(5)
*
**
***
****
*****
```

Population de poissons



Chez les poissons d'avril, les individus jeunes (moins d'un an) n'ont pas la même fécondité ni la même mortalité que les adultes. On note Jn et An les nombres de jeunes et d'adultes à l'année n, respectivement. A partir de recensements annuels, on a pu proposer un modèle décrivant l'évolution de la population :

$$\begin{cases} J_{n+1} = 0{,}4.J_n + 3{,}3.A_n \\ A_{n+1} = 0{,}5.J_n + 0{,}7.A_n \end{cases}$$

Avec :

$$\begin{cases} J_0 = 2 \\ A_0 = 4 \end{cases}$$

Q17. Ecrire deux fonctions récursives **jeune(n)** *et* **adulte(n)** *qui prennent en argument un* **entier n** *et qui renvoient respectivement la valeur de* **Jn** *et* **An**.