# DS 02

PTSI 2024/2025

## I Questions de bases : à rendre sur copie

1. On considère la liste suivante : `L = [[1, 3, 0],[0, 1.7, 5], [2, 1.2, 4]]`.
   Dire ce que fait chacune des commandes suivantes et préciser le contenu des variables `a`, `L` après l'exécution de chacune d'elle :

   a. `a = len(L[1])`
   b. `a = L[1][2]`
   c. `L.append([2,7])`
   d. `L.pop(1)`
   e. `L[-1].pop(1)`
   f. `a=[e[2] for e in L]`

2. On considère le dictionnaire ci-dessous dont les clefs sont des coordonnées géographiques et les valeurs des listes contenant successivement l'altitude de ce point, la température moyenne sur l'année et le taux d'humidité de la forme :

   ```
   dico = {(110,57) : [120, 15, 70], (12,154) : [15, 8, 55] ...}
   ```

   les "..." signifiant qu'on continue ainsi autant de fois qu'il y a de points dans la liste.

   a Quelle(s) commande(s) permet(tent) permettent de calculer la température moyenne obtenue sur tous les points du dictionnaire ?

   b Écrire une fonction qui étant donnée une altitude, extrait du dictionnaire toutes les coordonnées des points ayant cette altitude. Elle prendra en entrée le paramètre `alti` et renverra une liste.

3. On considère un tableau `numpy` bi-dimensionnel qu'on note `T`.

   a. Donner deux commandes qui permettent d'extraire la valeur contenu dans ce tableau en 2-ième ligne et 3-ième colonne.

   b. Donner une commande permettant d'extraire toutes les lignes à partir de la 3-ième.

   c. Donner une commande permettant d'extraire et renvoyer le sous-tableau bi-dimensionnel constitué des élément de `T` d'indices $(i, j)$ avec $i \geqslant 2$ et $j \leqslant 5$.

4. a. Que va afficher le script suivant (justifier) :

   ```
   T = numpy.array([[0,1,1,0,1],
                    [2,1,1,0,1]])
   if not(sum(T[1])%3 != 0 or T[1,0] > 2) :
       print("oui")
   else :
       print("non")
   ```

   b. Que va afficher le script suivant (expliquer en détail votre affirmation) :

   ```
   tmp ={}
   lst = [3, 2, 6, 1]
   for element in lst :
       if element > 2 :
           tmp[element] = element**2
   print(tmp)
   ```

c. On considère la suite récurrente $(u_n)_n$ définie par $u_0 = 1, u_1 = 2$ puis pour tout entier naturel $n$ par :

$$u_{n+2} = 2u_{n+1} + nu_n^2.$$

On admet que cette suite est définie pour tout entier naturel $n$ et qu'elle a pour limite $+\infty$. Écrire une fonction qui étant donné une valeur $M > 0$ fixée renvoie le premier entier naturel $n$ pour lequel $u_n > M$.

5. a. Écrire une fonction <u>récursive</u> permettant le calcul des termes de la suite $(u_n)_n$ précédente.

b. Comparer la performance de l'usage de cette fonction pour résoudre la question 4.d avec la solution adoptée en 4.c (expliquer).

# II Un programme à analyser : à rendre sur copie

On considère le script suivant

```
1         L =[3, 1, 2, 4]
2         m1 = L[0]
3         m2 = L[1]
4         if L[0] < L[1] :
5             m1 = L[1]
6             m2 = L[0]
7         for k in range(2,len(L)+1) :
8             a = L[k]
9             if a > m1 :
10                m2,m1 = m1,a
11            elif a > m2  :
12                m2 = a
13        print(m1,m2)
```

a. Ce programme comporte une erreur qui va bloquer son exécution par Python. Corriger là.

b. Exécuter vous même ligne à ligne les instructions de ce programme

| ligne | k | m1 | m2 | a |
|---|---|---|---|---|
| 1 | ... | ... | ... | ... |
| 2 | ... | ... | ... | ... |

et indiquerez dans chaque case les valeurs des variables après l'exécution de la ligne.

c. Expliquer ce que fait ce programme en exprimant par une phrase ce que représente les valeurs `m1,m2` finales.

d. Évaluer la complexité temporelle de programme en fonction de la longueur $N$ de la liste `L`.

# III Programmation : à faire sur machine

- Vous rendrez votre fichier de programme dans le cahier de prépa dans la partie "transfert de fichier".
- L'utilisation du memento Python est autorisée.

## III.1 Une simulation : type oral Maths 1 (type Simulation)

On considère dans cet exercice un jeu de 32 cartes ayant chacune une « hauteur » prise dans :

```
hauteurs = ["7","8","9","10","V","D","R","A"]
```

et une « couleur » pris dans

```
couleurs = ["T","K","C","P"]
```

correspondant à « Trèfle, Carreau, Coeur, Pique ».

1. a. Écrire une fonction `produit` qui étant donnée deux listes `ha,co` (pouvant par exemple être les listes précédentes) renvoie la liste de tous les couples `(h,c)` avec `h` pris dans `ha` et `c` pris dans `co`.
   b. Estimer la complexité temporelle de votre fonction par rapport aux longueurs $l_h$ et $l_c$ des deux listes (on donnera un résultat sous forme de $O$ en expliquant le résultat).
   c. Utiliser cette fonction pour stocker dans une variable `cartes` les couples trouvées pour les listes `hauteurs,couleurs` et représenter ainsi un jeu de 32 cartes.
   d. Vérifier informatiquement que votre nombre de cartes et les bon.
2. a. Écrire une fonction `lcartes` qui simule le tirage avec remise de 5 cartes dans le jeu . Le résultat sera renvoyé sous forme d'une liste et on pourra utiliser la fonction `numpy.random.randint` du module `numpy`
   b. Faire de même avec un tirage d'une main de 5 cartes prises simultanément. On utilisera la méthode `sample` du module `random` dont la syntaxe est :

   ```
   random.sample(liste,nombre d'éléménts à prendre)
   ```

   On nommera cette fonction `tirerMain` et un tel tirage de 5 cartes distinctes sera appelé une « main ».
3. a. Écrire une fonction `estCouleur` d'argument une main de 5 cartes et qui renvoie `True` si cette main est une « couleur » c'est à dire si toutes les cartes qui la compose sont de la même couleur et `False` sinon.
   b. Écrire une fonction `freq(n)` qui prend comme argument un entier naturel `n` et renvoie la fréquence d'apparition d'une couleur lors de `n` tirages successifs d'une main avec la fonction `tirerMain`.
   c. En déduire une approximation de la probabilité d'avoir une couleur en calculant la fréquence des quintes obtenues sur 100000 essais.
   d. Effectuer une représentation graphique des points $(N, f_N)$ où $f_N$ est la fréquence des mains qui ont été des couleurs sur $N$ réalisation du tirage. On utilisera des valeurs de $N$ allant de 10000 à 100000 avec un pas de 10000.

## III.2 Codage minimal d'une chaines de caractères : Oral Maths 1 (type Algorithmique)

Dans une liste ou une chaîne de caractère, on appelle *composante* une sous-liste ou sous-chaîne correspondant à la répétition d'un même élément ou caractère, précédée et suivi par rien ou un autre caractère.

Les composantes de `[1,1,0,0,1,1,1,0,1,1,1]` sont `[1,1]`, `[0,0]`, `[1,1,1]`, `[0]` et `[1,1,1]`.

Une composante peut ensuite se coder sous la forme d'un couple `[n,c]` où `n` est le nombre de fois que le caractère ou élément `c` est répété.

1. Écrire une fonction `coupures` dont l'argument est une liste ou une chaîne de caractères $S$ qui renvoie la liste des indices $i$ tels que $S[i] \neq S[i-1]$, à la fin de laquelle on ajoute la longueur de $S$. Par exemple pour la liste

   ```
   L = [1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1]
   ```

   le retour doit être

   ```
   [2, 4, 7, 8, 11].
   ```

2. Écrire une fonction `CP` de même argument $S$ qui renvoie une liste résultant de la mise bout-à-bout des couples codant les composantes de $S$.

   Ainsi, `CP([1,1,0,0,1,1,1,0,1,1,1])` donne `[[2,1],[2,0],[3,1],[1,0],[3,1]]`.

3. Écrire une fonction `nbits` dont l'argument est une entier naturel $n$ et qui renvoie le plus petit entier $a$ tel que $n \leqslant 2^a$.

4. Écrire une fonction `nbelem` dont l'argument est une liste ou une chaîne de caractères $S$ et qui renvoie le nombre d'éléments ou caractères distincts contenus dans $S$.

   Par exemple : `nbelem('aaaaggaadaaa')` donne 3.

5. En déduire une fonction `ntotalbits` qui renvoie le nombre minimum de bits nécessaire pour coder en binaire une chaine par le procédé :

   `S='aagggddaag'` $\longrightarrow$ `[[2,'a'],[3,'g'],[2,'d'],[1,'g']]`

   On observera par exemple que dans le cas de la chaine `'aagggddaag'`, le fait qu'il n'y ait que 4 éléments différents permet de les coder seulement sur 2 bits en faisant l'association : `'a':00, 'd':01, 'g':10` et ensuite coder la chaîne avec les longueurs des séquences consécutives.