FRANCE IOI
COMMENTAIRES SUR LES EXERCICES PYTHON

I. NIVEAU 2 : STRUCTURES AVANCEES.

A. Chapitre 2 : Listes, Tableaux.

1. Exo 6 : Stockage puis affichage inversé de valeurs.

De manière plus générale : Principe de transformation des entrées :

Tableau 1 liste les entrées (les antécédents x).

Tableau 2 liste la table T des transformations (la fonction T).

Tableau 3 liste les sorties y : y[i] = T(x[i])

2. Exo 7 : Recherche de médiane après tri.

- Il s'agit de trouver la médiane d'une liste de valeurs.
- Attention aux indices de tableaux : si un tableau a n valeurs, son système d'indices va de 0 à n-1 !

Donc penser à enlever 1 pour tout calcul d'indice faisant intervenir le nombre n de valeurs.

- Par exemple si on veut la valeur à la moitié d'un tableau T (la médiane donc !) :

  cas n pair : Médiane = Moyenne (T[n//2-1] et T[n//2]) et non Moyenne (T[n//2] et T[n//2+1])

  cas n impair : Médiane = T[n//2] et non T[n//2+1]

3. Exo 8 : Couples (minimums – maximums) après tri.

Nb pair de participants pour pouvoir faire des couples.

Nbs tous différents pour éviter les problèmes d'association.

Encore une fois, attention aux indices de tableaux.

4. Exo 9 : Permutations sur un tableau de valeurs.

Rappel de la permutation de 2 valeurs a et b en utilisant une troisième valeur auxiliaire :

```
aux = a
a = b
b = aux
```

5. Exo 10 : Inversion Valeurs et indices d'une liste.

Utilisation de la méthode index sur une liste exemple :

```
Liste = [ 1, 5, 3, 4, 4 ]
```

Liste.index(4) renvoie le premier indice où apparait la valeur 4 c-à-d indice 3 ici dans notre exemple.

B. Chapitre 3 : Chaînes de caractères.

1. Exo A] 5) : lister des chaînes de caractères que si elles sont strictement plus grandes que les précédentes.

Toujours la même difficulté pour le début de boucle : traitement à part.

On aurait pu s'affranchir de ce traitement à part en prenant fixant longueur de départ = 0 comme dans la correction, ce qui est meilleur.

2. Exo B] 1) : Inverser 2 mots sur une même ligne :

Les 3 écritures suivantes donnent le même formatage pour la sortie :

1. print (personne[1], personne[0])
2. print ("{} {}".format(mots[1], mots[0]))
3. prenom, nom = input().split(" ") équivaut à prénom = mots[0] et nom = mots[1]
   print ("{} {}".format(nom, prenom))

Le 3ème cas est très intéressant : il permet de catégoriser un splittage en évitant de créer le tableau mots mais en affectant directement le résultat du splittage aux variables prenom et nom.

3. Exo B] 2) : Tableau des longueurs des mots tirés d'un texte et nombre d'occurrences associées : histogramme.

Exercice difficile :

```
1  nblignes, nbmots = [int(x)for x in input().split(' ')]
2  listelongueurmot=[]
3  cataloguelongueurs=[]
4  for i in range (nblignes) :   #Parcours du nb de lignes
5      listelongueurmot.extend([len(mot) for mot in input().split(' ')]) #Fabrication par extension
6                                                                        #de la longue liste des
7                                                                        #longueurs de tous les mots
8  listelongueurmot.sort()      #Tri de la liste des longueurs de mots
9  for k in listelongueurmot:   #Parcours de la liste des longueurs de tous les mots
10     if k not in cataloguelongueurs :  #Fabrication de la liste cataloguelongueur par ajout
11         cataloguelongueurs.append(k)  #d'élément non présent dans le catalogue
12 for j in cataloguelongueurs :        #Parcours du catalogue
13     print ('{} : {}'.format(j,listelongueurmot.count(j)))     #sortie des valeurs du catalogue et
14                                                               #du nb d'occurences correspondant
```

Beaucoup de méthodes sur les listes ou de fonctions sur les listes ont été utilisées dans ma solution :

- entrée en liste splittée convertie en entier :

  nblignes, nbmots = [int(x) for x in input().split(' ')] méthode par compréhension de liste

ou bien nblignes, nbmots = map (int(), input().split(' ')) méthode par map (fonction, [liste])

- extension d'une liste par une compréhension de liste :

  listelongueurmot.extend([len(mot) for mot in input().split(' ')])

- méthode sort() pour trier une liste : listelongueurmot.sort().
- boucle sur liste : for k in listelongueurmot :
- occurrence ou non d'un élément dans une liste : k not in cataloguelongueurs.
- extension d'une liste par ajout d'un élément : cataloguelongueurs.append(k).
- bouclage sur une liste et formatage en sortie :

  for j in cataloguelongueurs :

print ('{} : {}'.format (j, listelongueurmot.count(j)))

Dans la solution présentée sur France ioi : on peut se poser la question pourquoi apriori des mots de 100 lettres ?

4. Exo C] 1) : Boucler sur une chaîne de caractère.

Ma solution est meilleure que celle de France ioi : pas besoin de calculer de longueur ou je ne sais quoi.

5. Exo C] 2) : Inverser un texte.

On boucle sur la longueur du texte en partant de la fin :

for k in range (longueur) :

print (texte[longueur-1-k], end='')

6. Exo C] 3) : Triage selon initiale.

Dans la solution proposée par France ioi, la condition du elif : `elif nom[0] <= "P":` sous-entend `elif 'G'<=nom[0] <='P' :`

7. *Exo C] 4) : suppression de caractères dans une chaîne de caractères (string).

- Ma solution : en utilisant la méthode join sur une compréhension de liste :

```
resultat=''.join([k for k in chaine if k not in [" ", 'A', 'E', 'I', 'O', 'U', 'Y'] ])
```

- Méthode France ioi : parcours de la chaine de caractères et affichage ou non des caractères un par un.

C. Chapitre 4 : Fonctions.

1. Exo 2) : Bi-code.

- Ma solution : une fonction qui factorise au max les instructions, un tableau de codes secrets et un tableau de phrases de sorties correspondantes, avec une boucle for pour appeler la fonction.
  Attention aux pré-traitement et post-traitement du while.
  Ma fonction passe en argument le numéro d'étape de cette vérification en 2 temps.

```
1  def verifcode(i):
2      print('Entrez le code :')     # Pré-traitement de la boucle : un premier print d'entrée hors
3  boucle.
4      while int(input()) !=codesecret[i] :
5          print('Entrez le code :')
6      print(phrase[i])              # Post-traitement de la boucle : print de sortie
7
8  codesecret=[4242,2121]
9  phrase=['Premier code bon.','Bravo.']
10 for k in range (2) :
11     verifcode(k)
```

- Solution de France IOI : pré-traitement du while par la variable tentative qui est en fait le int(input()), initialisée à code +1 donc forcément différent de code.
  La fonction passe le code en argument.