Feuille Info_6 : Parcours des listes

Ex 1 : On considère la fonction suivante :

```
def exercice(L):
    """ L est une liste de listes de nombres """
    c = 0
    for S in L:
        for t in S:
            if t == 0:
                c += 1
    return c
```

1) Réécrire cette fonction en utilisant des boucles basées sur les indices plutôt que sur les valeurs.
2) Ajouter des commentaires explicatifs sur les lignes importantes.
3) Expliquer en une phrase ce que fait cette fonction.
4) Ecrire en trois lignes un programme testant cette fonction.

Ex 2 : On considère la fonction suivante :

```
def exercice(L):
    """ L est une liste de listes de nombres """
    for k in range(len(L)):
        L[k].append(0)
```

1) Réécrire cette fonction en utilisant des boucles basées sur les valeurs plutôt que sur les indices.
2) Ajouter des commentaires explicatifs sur les lignes importantes.
3) Expliquer en une phrase ce que fait cette fonction.
4) Ecrire en trois lignes un programme testant cette fonction.

Ex 3 : On considère la fonction suivante :

```
def exercice(Texte, c):
    """ Renvoie le nombre de caractère c dans la chaine Texte  """
```

1) Ecrire cette fonction avec un boucles basée sur les indices.
2) Ecrire cette fonction avec un boucles basée sur les valeurs.
3) Ajouter des commentaires explicatifs sur les lignes importantes.
4) Ecrire en trois lignes un programme testant cette fonction.

Ex 4 : On considère la fonction suivante :

```
def exercice(L):
    """ L est une liste de chaines de caractères """
    c = 0
    for mot in L:
        for k in range(len(mot)):
            c += (mot[k] == ' ')
    return c
```

1) Réécrire cette fonction en utilisant des boucles basées uniquement sur les indices.
2) Réécrire cette fonction en utilisant des boucles basées uniquement sur les valeurs.
3) Ajouter des commentaires explicatifs sur les lignes importantes.
4) Expliquer en une phrase ce que fait cette fonction.
5) Ecrire en trois lignes un programme testant chacune de ces fonctions.

Ex 5 : On considère la fonction suivante :

```
def exercice(M):
    """ M est une chaine de caratères """
    L = ['']
    for _ in range(len(M)):
        S = []
        for A in L:
            for c in set(M) :
                if A.count(c) < M.count(c):
                    S.append(A + c)
        L = S
    return L
```

1) Ajouter des commentaires explicatifs sur les lignes importantes.
2) Réécrire ce programme en changeant les noms des variables pour améliorer la compréhension :
   Vous pouvez vous inspirer des mots suivants :
   `chemins_actuels` `nouveaux_chemins` `caractere` `mot` `chemin`
3) Que fait cette fonction si on passe en argument `M = "ANNA"`
4) Changez le nom de la fonction et dites ce qu'elle fait.
5) Réécrire la fonction de l'énoncé en remplaçant les 6 lignes dans la boucle `for` par une ligne avec une liste en compréhension.

Ex 6 : On considère la fonction suivante :

```
def mystere(n, p):
    """  ... """
    L = [ [] ]
    for i in range(p-1):
        L = [ x + [k] for x in L
                      for k in range(n+1)  if sum(x+[k]) <= n ]
    return [ x + [n-sum(x)] for x in L ]
```

1) Quel est le type des variables `L`, `x`, et `k` ?
2) A l'exécution de cette fonction combien de tours fait la boucle `for i` ?
3) Dans cette question, on passe pour arguments `n = 3` et `p = 3`. (*On exécute* `mystere(3, 3)`)
   Que vaut `L` à la fin de chaque tour de boucle ?
4) Pour `x` une liste d'entiers et `n` un entier quelconques,
   quelle est la somme des valeurs de la liste `x+[n-sum(x)]` ?
5) Que fait cette fonction ?

Ex 7 : En vous inspirant de **Ex 5** et **Ex 6**.

1) Ecrire une fonction qui renvoie la liste de toutes les permutations de $[\![0; n-1]\!]$.
2) Ecrire une fonction qui renvoie la liste de tous les dérangements de $[\![0; n-1]\!]$.
   (les permutations $(x_0, ...x_{n-1})$ *vérifiant* $\forall i \in [\![0; n-1]\!], x_i \neq i$*)*