Ds - Python + Formulaires html

Les 4 exercices de ce Ds sont indépendants et peuvent être traités dans n'importe quel ordre. Répondre sur feuille de copie.

1- EXERCICE 1. : *(5 points)*

Une fonction Python appelée *nb_repetitions()* prend en argument un élément et une liste . Elle retourne le nombre de fois où l'élément apparaît dans le tableau.

Exemples d'exécutions dans la console de cette fonction :

```
>>> nb_repetitions(5, [2, 5, 3, 5, 6, 9, 5])
3

>>> nb_repetitions('A', ['B', 'A', 'B', 'A', 'R'])
2

>>> nb_repetitions(12, [1, 3, 7, 21, 36, 44])
0
```

Question : Ecrire le code python de cette fonction sur feuille de copie.

```
def nb_repetitions(elt,liste) :
    nb = 0
    for e in liste :
        if e == elt :
            nb = nb + 1
    return nb
```

Corrigé

2- EXERCICE 2. : *(5 points)*

On affecte à chaque lettre de l'alphabet un code selon le tableau ci-dessous :

A	B	C	D	E	F	G	H	I	J	K	L	M
1	2	3	4	5	6	7	8	9	10	11	12	13

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

Cette table de correspondance est stockée dans un dictionnaire dico où les clés sont les lettres de l'alphabet et les valeurs les codes correspondants.

```
dico = {"A": 1, "B": 2, "C": 3, "D": 4, "E": 5, "F": 6,
        "G": 7, "H": 8, "I": 9, "J": 10, "K": 11, "L": 12,
        "M": 13, "N": 14, "O": 15, "P": 16, "Q": 17,
        "R": 18, "S": 19, "T": 20, "U": 21, "V": 22,
        "W": 23, "X": 24, "Y": 25, "Z": 26}
```

Pour un mot donné, on détermine d'une part son code alphabétique concaténé, obtenu par la juxtaposition des codes de chacun de ses caractères, et d'autre part, son code additionné, qui est la somme des codes de chacun de ses caractères.
Par ailleurs, on dit que ce mot est « parfait » si le code additionné divise le code concaténé.

Exemples :

- Pour le mot "PAUL", le code concaténé est la chaîne '1612112', soit l'entier 1 612112. Son code additionné est l'entier 50 car 16 + 1 + 21 + 12 = 50. 50 ne divise pas l'entier 1 612 112. Ainsi, le mot "PAUL" n'est pas parfait.

- Pour le mot "ALAIN", le code concaténé est la chaîne '1121914', soit l'entier 1121 914. Le code additionné est l'entier 37 car 1 + 12 + 1 + 9 + 14 = 37. 37 divise l'entier 1 121 914. Ainsi, le mot "ALAIN" est parfait.

La fonction codes_parfait() donnée ci-dessous, est incomplète. Elle prend en argument un mot et retourne son code concaténé, son code additionné un booléen indiquant si le mot est parfait ou non.

```
dico = {"A": 1, "B": 2, "C": 3, "D": 4, "E": 5, "F": 6,
        "G": 7, "H": 8, "I": 9, "J": 10, "K": 11, "L": 12,
        "M": 13, "N": 14, "O": 15, "P": 16, "Q": 17,
        "R": 18, "S": 19, "T": 20, "U": 21, "V": 22,
        "W": 23, "X": 24, "Y": 25, "Z": 26}

def codes_parfait(mot):
    code_concatene = ""
    code_additionne = 0
    for c in mot:
```

Exemple d'exécution dans la console :

```
>>> codes_parfait("PAUL")
(50, 1612112, False)

>>> codes_parfait("ALAIN")
(37, 1121914, True)
```

Question : Compléter sur feuille de copie le code python de cette fonction.

Remarque : On rappelle que pour tester si un entier b divise un entier a, on utilise l'expression modulo a%b==0. En effet, a % b renvoie le reste de la division euclidienne de a par b, s'il est nul, alors b divise a.

Corrigé

```
def codes_parfait(mot):
    code_concatene = ""
    code_additionne = 0
    for c in mot:
        code_concatene = code_concatene + str(dico[c])
        code_additionne = code_additionne + dico[c]
    code_concatene = int(code_concatene)
    mot_est_parfait = code_concatene % code_additionne == 0
    return code_additionne, code_concatene, mot_est_parfait
```

3- EXERCICE 3. : *(5 points)*

On considère des chaînes de caractères contenant uniquement des majuscules et des caractères *. Elles sont appelées *mots à trous*.

Par exemple INFO*MA*IQUE ou ***I***E** et *S* s ont des mots à trous.

La fonction *correspond()* prend en arguments deux chaînes de caractères nommées *mot* et *mot_a_trous*. Elle retourne *True* si en dehors des caractères '*', *mot* et *mot_a_trous* sont identiques. Elle retourne *False* sinon.

Exemple d'exécution dans la console :

```
>>> correspond('INFORMATIQUE', 'INFO*MA*IQUE')
True

>>> correspond('AUTOMATIQUE', 'INFO*MA*IQUE')
False

>>> correspond('AUTO', '*UT*')
True

>>> correspond('AUTO', '*U')
False

>>> correspond('AUTO', '*****')
False
```

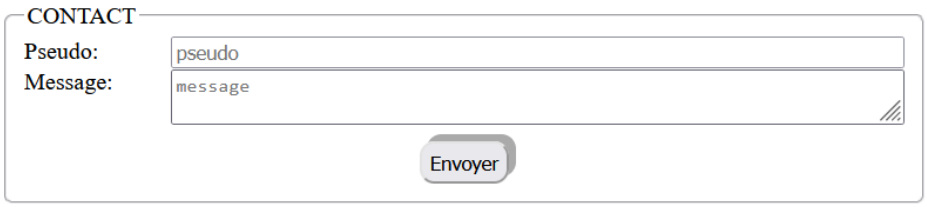
Question : Ecrire sur feuille de copie le code python de cette fonction.

Corrigé

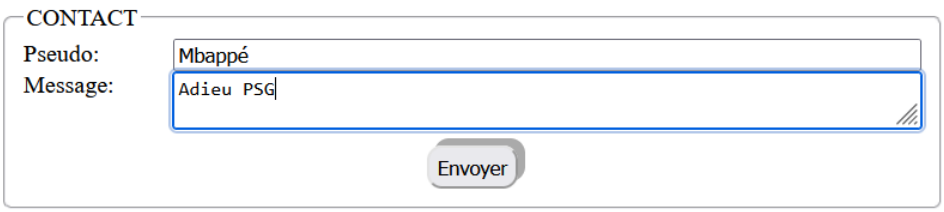
```
def correspond(mot , mot_a_trou) :
    if len(mot) != len(mot_a_trou) : return False
    for i in range(len(mot)) :
        if mot_a_trou[i] != "*" and mot_a_trou[i] != mot[i] : return False
    return True
```

4- EXERCICE 4. : *(5 points)*

Le code html ci-dessous du bloc <body> , permet l'affichage du formulaire ci-contre :



En saisissant les valeurs ci-contre dans ce formulaire et en cliquant sur le bouton, les données suivantes s'ajoutent en fin de l'Url :



l?pseudo=Mbappé&texteMessage=Adieu+PSG&boutonEnvoyer=parti

Question : Compléter sur feuille de copie, le code html du bloc <fieldset> :

Corrigé

```
<body>
    <form action="">
        <fieldset>
            <legend>CONTACT</legend>
            <div>
                <label for="n">Pseudo:</label>
                <input type="text" id="n" name="pseudo" placeholder="pseudo" autofocus>
                <label for="m">Message:</label>
                <textarea type="text" id="m" name="texteMessage" placeholder="message"></textarea>
            </div>
            <div class="bouton">
                <button type="submit" name="boutonEnvoyer" value="parti">Envoyer</button>
            </div>
        </fieldset>
    </form>
</body>
```

A titre d'information, on donne également ci-contre le code Css correspondant.

```
<style>
    fieldset {
        max-width: 600px;
        margin:auto;
        border-radius: 5px;
    }
    form fieldset :nth-child(2){
            display: grid;
            grid-template-columns: 1fr 5fr;
        }
    .bouton{
        text-align: center;
    }
    button{
        padding:5px;
        border-radius: 10px;
        box-shadow: 5px -5px #AAA;
        margin:10px 0 2px 0;
    }

</style>
```