

Examen SEMESTRE 1

Matière : INFORMATIQUE

Classe : 1^{ère} Année BG

Durée : 2h

Toutes les réponses doivent être rédigées en Python.
L'indentation dans les instructions Python sera prise en considération dans le barème

**Exercice 1 : (3 pts)**

Ecrire un script python qui permet de :

1. Saisir deux listes L1 et L2 de taille respectivement **n1** et **n2** avec $n1 > 0$ et $n2 > 0$.
2. Tester si L1 et L2 sont équivalentes (chaque élément de L1 existe dans L2 et chaque élément de L2 existe dans L1).
3. Afficher le résultat du test.

Exemple : les listes $L1=[1,2,3]$ et $L2=[3,2,1,3,3,1,2]$ sont équivalentes

Exercice 2 : (7 pts)

La base de prix des produits du magasin est représentée en Python par un dictionnaire **Base** dont :

- Les clés sont les noms de produits, de type **str**.
- Les valeurs sont des **tuples** (Prix, Quantité), où Prix est de type **float** et Quantité est de type **int**.

*Exemple : $Base=\{ "Ordinateur": (1000.0,20), "Imprimante": (650.75,15),$
 $"Souris": (15.0,14), "Scanner": (550.5,6) \}$*

Travail demandé

Ecrire un script Python qui permet de :

1. Afficher le prix moyen des produits disponibles dans le magasin.
2. Afficher le nom du produit le plus cher disponible dans le magasin.
3. Saisir une chaîne de caractères **Produit** représentant le nom du produit et formée des caractères alphabétiques (des lettres) dont la première lettre est une lettre majuscule.
4. Tester la disponibilité du **Produit** saisi dans le dictionnaire **Base** en affichant un message "le produit recherché existe dans le magasin" si le produit existe et un message "le produit recherché n'existe pas dans le magasin", dans le cas contraire.

5. Saisir un entier **q strictement positif** représentant la quantité, du Produit précédemment saisi, qu'on veut acheter.
6. En cas de sa disponibilité, mettre à jour la quantité du produit restante après achat tout en vérifiant si elle est, encore suffisante dans le stock. Dans le cas contraire, afficher le message "quantité insuffisante".

Exercice 3 : (10 pts)

Dans ce problème on cherche à évaluer la force des mots de passe des utilisateurs d'un site web.

Un mot de passe est une chaîne de caractères qui ne comporte pas d'espaces.

La force d'un mot de passe varie, selon la valeur d'un **Score** calculé, de 'Très faible' jusqu'à 'Très fort' :

- Si le **score** <20, la force du mot de passe est '**Très faible**'
- Sinon si le **score** <40, la force d'un mot de passe est '**Faible**'
- Sinon si le **score** <80, la force du mot de passe est '**Fort**'
- Sinon la force du mot de passe est '**Très Fort**'

Le score se calcule en additionnant des bonus et en retranchant des pénalités.

Les bonus attribués sont :

- Nombre total de caractères * 4
- (Nombre total de caractères – nombre de lettres majuscules) * 2
- (Nombre total de caractères – nombre de lettres minuscules) * 3
- Nombre de caractères non alphabétiques * 5

Les pénalités imposées sont :

- La longueur de la plus longue séquence de lettres minuscules * 2
- La longueur de la plus longue séquence de lettres majuscules * 3

Exemple :

Pour le mot de passe 'P@cSI_promo2017', le score se calcule comme suit :

La somme de bonus = $15*4 + (15-3)*2 + (15-6)*3 + 6*5 = 141$

- Le nombre total de caractères = 15
- Le nombre de lettres majuscules = 3
- Le nombre de lettres minuscules = 6
- Le nombre de caractères non alphabétiques = 6

La somme des pénalités = $5*2 + 2*3 = 16$

- La longueur de la plus longue séquence de lettres minuscules ('promo') = 5

- La longueur de la plus longue séquence de lettres majuscules ('SI') =2

Le score final = $141 - 16 = 125$; puisque $125 > 80$ alors le mot de passe est considéré comme 'Très fort'.

Travail demandé :

Ecrire un programme python qui permet de :

1. Saisir une chaîne correspondante au mot de passe ,
2. Vérifier si la chaîne ne contient pas des espaces, si la chaîne contient des espaces on doit les supprimer.
3. Calculer le nombre de caractères minuscules **Nb_min**.
4. Calculer le nombre de caractères majuscules **Nb_max**.
5. Calculer le nombre de caractères non alphabétiques **Nb_NonAlph**.
6. Calculer longueur de la plus longue séquence de lettres majuscules **Long_Maj**.
7. Calculer la longueur de la plus longue séquence de lettres minuscules **Long_Min**.
8. Calculer la somme des bonus, la somme des pénalités et le score de mot de passe sachant que : **le score= la somme des bonus – la somme des pénalités**.
9. Afficher la force du mot de passe saisi selon la valeur du score.

ANNEXE – Sans aucune obligation, les fonctions suivantes pourraient vous être utiles.

- **L.append(e):** permet d'ajouter l'élément **e** à la fin de la liste **L**.
- **ch.split(sep):** permet de retourner la liste formée par les sous-chaînes de la chaîne de caractères **ch** séparées par le caractère **sep** donné en argument.
- **ch.isupper():** retourne **True** si la chaîne **ch** est en majuscule
- **ch.islower():** retourne **True** si la chaîne **ch** est en minuscule
- **ch.isalpha():** retourne **True** si la chaîne **ch** contient uniquement des alphabétiques.
- **ch.istitle():** retourne **True** si seule la première lettre de chaque mot de la chaîne **ch** est en majuscule, **False**, sinon.
- **ch.isdigit():** retourne **True** si la chaîne de caractères **ch** n'est formée que par des caractères numériques, **False**, sinon.
- **sep.join(iterable):** permet de retourner une chaîne en joignant tous les éléments d'un itérable suivant le caractère **sep**.
- **ch.count(c):** retourne le nombre d'occurrence du caractère **c** dans la chaîne **ch**.
- **s1.symmetric_difference(s2):** retourne un ensemble **s** contenant les éléments qui appartiennent à **s1** et non à **s2** et les éléments qui appartiennent à **s2** et non à **s1**
- **s1.union(s2) :** retourne un ensemble **s** contenant tous les éléments des deux ensembles **s1** et **s2**.
- **x in d :** vérifie si **x** appartient à **d**.
- **d.values() :** retourne une liste contenant les valeurs du dictionnaire **d**.
- **d.keys() :** retourne une liste contenant les clés du dictionnaire **d**.
- **max(d) :** retourne la plus grande valeur des clés.
- **d.pop(cle) :** supprime le couple (clé/valeur) et retourne la valeur correspondante