



Examen – S2 – 2023/2024

Filière : 1 ^{ère} LFI	Matière : Programmation Python		Enseignants : Dr. Abir Ben Hmida Sakly
Date : 21 / 05 / 2024	Nbr de Crédits : 2	Coefficient : 1.5	Documents autorisés : Non
Durée de l'examen : 1h30	Régime d'évaluation : Mixte EX (60%) + DS (25%) + TP (15%)		Nombre de pages : 08
Nom & Prénom :	Matricule :		
Signature :	Code confidentiel :		Classe :

NOTE : Répondre directement sur les feuilles de l'examen /

Note

/ 20

Exercice 1 : (12 points)

Dans cet exercice, nous allons manipuler des dictionnaires et les listes pour gérer un magasin de vêtements et vous souhaitez organiser vos stocks de manière efficace.

Vous disposez d'un dictionnaire **produit** qui contient les informations sur chaque produit vendu dans votre magasin.

- ✓ La clé de chaque entrée du dictionnaire est le nom du produit (str).
- ✓ La valeur associée à chaque clé est une liste contenant les informations du produit:
 - prix: le prix unitaire du produit (float).
 - quantité: la quantité en stock du produit (int).

Partie 1

1- Donner une expression python pour construire le dictionnaire « **produits** » correspondant à la table suivante :

Nom	Prix	Quantité
"Chemise"	29.99	30
"Pantalon"	39.99	40
"Jupe"	24.99	20
"Veste"	59.99	25
"Robe"	49.99	35
"Short"	19.99	15
"Pull"	19.99	45
"Chaussures"	69.99	10

Partie 2 : On vous demande d'implémenter les fonctions suivantes :

- 2- **AjouterProduit(produits, nom, prix, quantité):** Ajoute un nouveau produit au dictionnaire « **produits** ».
- 3- **RechercherProduit(produits, nom):** Recherche un produit par son nom dans le dictionnaire « **produits** » et affiche ses informations (prix et quantité).
- 4- **AfficherStock(produits):** Affiche la liste de tous les produits du dictionnaire (Nom, prix et quantité).
- 5- **VendreProduit(produits, nom, quantité):** Vend une certaine quantité d'un produit et met à jour la quantité en stock.
- 6- **ProduitLePlusCher(produits):** affiche le produit (nom et prix) avec le prix le plus élevé.

Ne rien écrire ici

- 7- **ProduitEnRupture(produits)**: chercher et afficher la liste des noms de produits dont la quantité en stock est nulle.
- 8- **RechercherParPrix(produits ,prix_min, prix_max)**: Permet de rechercher et afficher des produits(Nom et prix) dans une plage de prix donnée. Les paramètres **prix_min** et **prix_max** représentent les limites de la plage de prix.
- 9- **AppliquerRemise(produits ,nom, pourcentage)**: Permet d'appliquer une remise sur le prix d'un produit spécifique.
- Le paramètre pourcentage représente le pourcentage de remise à appliquer.
 - La fonction met à jour le prix du produit dans le dictionnaire « **produits** ».
- 10- **CalculerValeurTotaleStock(produits)**: Permet de calculer et retourner la valeur totale du stock en multipliant le prix de chaque produit par sa quantité en stock.
- 11- **IdentifierProduitsBasRotation(produits, seuil_rotation)**: Permet de chercher et afficher les noms des produits dont la rotation est inférieure à un seuil donné.
Le paramètre **seuil_rotation** représente la quantité minimale de vente souhaitée pour une période donnée.
- 12- Ecrire un **programme principal** détaillé pour la gestion des stocks d'un magasin de vêtements en utilisant les fonctions ci-dessus.

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

This image shows a full page of primary-ruled notebook paper. It features multiple sets of horizontal dashed lines for writing, separated by solid horizontal lines. Vertical solid lines on either side define left and right margins. The paper is otherwise blank, with no handwriting or other markings.

Exercice 2 : Le problème du sac à dos (8 points)

Étant donné les poids et les valeurs de n articles, nous devons mettre ces articles dans un sac à dos de capacité C pour obtenir la valeur totale maximale dans le sac à dos.

Une solution efficace consiste à utiliser l'approche gloutonne. L'idée de l'approche gloutonne est de calculer le rapport **valeur/poids** pour chaque article et de trier les articles sur la base de ce rapport. Ensuite, prenez l'élément avec le rapport le plus élevé et ajoutez-le jusqu'à ce que nous ne puissions plus ajouter l'élément suivant dans son intégralité et à la fin, ajoutez-le autant que possible.

Example:

```
articles = [(60, 10), (100, 20), (100, 60), (40, 20)]
capacite = 50
```

Le script affichera:

```
Sac_a_dos = [(60, 10), (100, 20), (40, 20)]
La valeur maximale du sac à dos = 200.0
```

Travail demandé:

Ecrire un script Python qui permet de résoudre le problème selon l'approche proposée.