Université Polytechnique des Hauts-De-France
Institut des Sciences et Techniques ISTV

Licence 1 : TP N° 1 d'Algorithmique Semestre2

Gestion des dépenses

On se propose ici de créer un programme pour aider aux choix de produits sur un site d'e-commerce. Chacun des produits possède un prix et une note entre 0 et 10 donnée par les clients.

1. Dans ce TP, on utilisera **MAXPRODUIT** un entier définissant le nb max de produits à gérer. On utilisera des tableau de MAXPRODUIT réels (de type TabReels) pour gérer des prix, des moyennes, …. On utilisera aussi des tableaux à 2 dimensions (de type MatReels) composés de 2 lignes et de MAXPRODUIT colonnes pour stocker sur une colonne i, le prix (en ligne 0) et la note (en ligne 1).
   *Vous utiliserez l'éditeur de votre choix, **Pyzo**, Thonny, EduPython, Spyder, …..*
   *On utilisera la librairie **numpy** pour les tableaux avec ses fonctions **zeros** et **arrays***
   *On utilisera la libraire **typing** pour définir les types*
   Ainsi, on placera donc en entête du programme :

```
from numpy import zeros, array
from typing import Iterable, TypeVar

TabReels = TypeVar(Iterable[float])
MatReels = TypeVar(Iterable[TabReels])

MAXPRODUITS = 100
```

   Rappel du cours :

```
# pour définir un tableau 10 réels initialisés à 0
tab0 = zeros(10,float)
# pour définir un tableau en lui donnant des valeurs directement
tab = array([12,43,23])
#pour placer une valeur dans la 1ere case :
tab[0] = 100

# pour définir une matrice (2 lignes, 5 colonnes) d'entiers initialisé à 0
mat0 = zeros((2,5), int)
# pour définir une matrice en lui donnant des valeurs directement
mat = array([[12,43,23], [2,6,3]])
#pour placer une valeur dans la 1ere ligne, 1ere colonne  :
mat[0][0] = 100
```

2. Définir la procédure **saisir_produit(no:int, tabPrixNotes:MatReels)** où no est le no du produit et tabPrixNotes le tableau à 2 dimensions. Cette fonction demande à l'utilisateur le prix et la note du produit no et range ces éléments dans la colonne no. Pour rappel, tabPrixNotes[0][0] contient le prix du produit n° 0, et tabPrixNotes[1][0] contient la note du produit n°0.
3. Définir la procédure **saisir_produits(nb:int, tabPrixNotes:MatReels)** qui permet de saisir 'nb' produits en faisant appel à la procédure précédente.
4. Définir la procédure **afficher_detail_produit(no:int, tabPrixNotes:MatReels)** qui affiche le no du produit, son prix et sa note

5. Définir la procédure **afficher_produits(nb:int, tabPrixNotes:MatReels)** qui affiche les détails des 'nb' produits en faisant appel à la procédure précédente.
6. Définir la fonction **max_tab(nb:int, tab:TabReels)->float** qui *retourne* la valeur maximum parmi les 'nb' valeurs du tableau 'tab', tableau simple à 1 seule dimension.
7. Définir la fonction **min_tab(nb:int, tab:TabReels)->float** qui *retourne* la valeur minimum parmi les 'nb' valeurs du tableau 'tab', tableau simple à 1 seule dimension.
8. Définir la procédure **afficher_moins_chers(nb:int, tabPrixNotes:MatReels)** qui affiche le prix et la note des produits les moins chers parmi les nb éléments du tableau à 2 lignes tabPrixNotes. *On cherchera donc d'abord le prix minimum dans la 1ère ligne du tableau tabPrixNotes (donc le minimum de tabPrixNotes[0]), puis on parcourt tabPrixNotes pour afficher les produits correspondant à ce minimum*.
9. Définir la procédure **afficher_mieux_notes(nb:int, tabPrixNotes:MatReels)** qui affiche le détail des produits les mieux notés. *On cherchera donc d'abord la note maximum dans la 2nde ligne du tableau tabPrixNotes (donc le max de tabPrixNotes[1]), puis on parcourt tabPrixNotes pour afficher les produits correspondant à ce maximum*.
10. On souhaite maintenant trouver le ou les produits représentant les meilleur compromis entre le prix et la note. Comme ces données se trouvent sur des intervalles de valeurs différents (la note est limitée entre 0 et 10, pas les prix), ces données ne peuvent peuvent être combinées directement; il faut donc porter leurs valeurs sur l'intervalle [0,1]. Ainsi, nous obtiendrons un tableau 'tabPrixNormes' qui représente les prix ramenés sur [0,1] (par exemple, si tabPrix=[2,6,12,20], alors tabPrixNormes=[0.1, 0.3, 0.6, 1], c'est-à-dire que 'tabPrixNormes' contient les valeurs de 'tabPrix' divisées par le maximum de 'tabPrix')
    Définir la procédure **remplir_tab_norme(nb:int, tab:TabReels, tabNorm:TabReels)** qui remplit les nb cases de 'tabNorm' avec les valeurs de 'tab' divisées par le maximum de 'tab'.
11. Définir la procédure **trouver_compromis(nb:int, tabPrixNotes:TabReels)** qui :
    1. crée et rempli les tableaux 'tabNormPrix' et 'tabNormNotes' qui contiennent respectivement les valeurs normalisées des tableaux 'tabPrixNotes[0]' et 'tabPrixNotes[1]'.
    2. crée le tableau de nb valeurs 'tabMixe' qui contient une valeur qui mixe l'intérêt pour une note élevée et un prix bas. Ce mixage est défini par « noteNormée * (1-prixNormé*0.9) ».
       *Ainsi un produit de note 5 (sur un maximum de 10, donc noteNormée=5/10=0.5) et de prix 85€ (avec un prix maximum trouvé à 100€, donc prixNormé=85/100=0.85) aura comme intérêt (0.5 * (1 - 0.85)) =0.075… Un autre produit de note 4 et de prix 80€ aura un intérêt de 0.08, il sera donc perçu comme plus intéressant..*
    3. affiche le ou les produits dont l'intérêt (stocké dans tabMixe) est le plus grand.

12. Ecrire la fonction ***moyenne(nb:int, tab*:TabReels*)->float*** qui retourne la moyenne des nb premières valeurs du tableau 'tab'

13. Ajouter procédure principale **test_TP1()** à votre code :

```
def test_TP1():
    prix:TabReels = np.array([81,72,85,71,66,104,91,87])
    notes:TabReels = np.array([2,4,5,3,2,0,2,5])
    prixNotes:MatReels=np.array([prix, notes])
    nb:int = len(prix)
    afficher_moins_chers(nb, prixNotes)
    afficher_mieux_notes(nb, prixNotes)
    trouver_compromis(nb, prixNotes)
    print("Moyenne des prix : " + str(moyenne(nb, prix)))
    print("Moyenne des notes : " + str(moyenne(nb, notes)))
```

1. Exécuter le.. Quel est le produit le moins cher, quels sont les mieux notés, quel est le meilleur compromis ?

IMPORTANT : Il faut que le code soit lisible, avec des noms de variables explicites, et que chaque fonction et procédure soit précédées d'un commentaire expliquant ce que fait la fonction.

Il faut également que le dialogue avec l'utilisateur soit clair ainsi que les affichages !