Henry
Letellier
T1

mercredi 3 février 2021

Numériques et sciences de l'informatique
DST d'NSI
Tiers Temps

Numériques et sciences de l'informatique

DST d'NSI

Note : _____	Appréciations :	Signature :

Problème 1 : Récursivité (Python)

1)

Le gros problème avec la fonction **pyramide_inversee(*n*)** est qu'elle est sans fin.

2)

A cette question je n'avais pas de none mais seulement le transfert du dernier élément de la première liste (*troupeau1*) sur la fin de la seconde liste (*troupeau2*). Pour résoudre ce problème là il suffit de rajouter la ligne « *for i in range(len(troupeau1)) :* » et d'indenter les lignes *mouton=troupeau1.pop()* et *troupeau2.append(mouton)*.

Le code sera donc :

code	Légende

def **saute_rep**(*troupeau1:list, troupeau2:list*):
for i in range(len(troupeau1)):
 	
mouton=troupeau1.pop()
 	
troupeau2.append(mouton)
	#On définit la fonction
	#On boucle pour la longueur de la liste troupeau1
	#On supprime le premier élément de la première liste que l'on stockera dans une variable du nom de mouton
	#On ajoute le contenu de la variable mouton à la liste troupeau 2

3)

Pour résoudre le problème de ce petit programme il suffit de remplacer *n/2* par *n//2*, ce dernier renverras un nombre entier et ne générera donc pas d'erreur.

4)

Pour résoudre le problème il suffit de faire vérifier le tour suivant pour savoir si le programme peut faire le tour actuel.

code	Légende
i=0 while i+1<len(tab1) or i+1<len(tab2): tab1[i]=tab1[i]-tab2[i] i+=1	**#On initialise i à 0** **#On boucle tant que les deux conditions sont vérifiées** **#On ajoute à tab1[i] la différence de tab1[i] et tab2[i]** **#On incrémente i de 1**

5)

La seule erreur possible pour ce programme est de devoir comparer un *int* et un *float* ou un *int* et un *str*. Pour palier à cela, il suffit de faire « *if str(e)==str(x):* » au lieu de « *if e==x:* » ce qui format les type en string, et donc évite le genre d'erreur « *must be str not int* ».

6)

Python affichera le message « *pas egal* » car pour lui 0.1+0.2 est égal à 0.300… et non *0.3*. La manière qui devrait être utilisée pour résoudre ce problème serai d'introduire la fonction **add_float**(*f1*,*f2*) dont je ne donnerais pas le code mais qui prendra chaque chiffre du nombre 1 et du nombre 2 et les ajouteras entre eux avant de retourner un *float* unique (ici *0.3*).

Problème 2 : Bases De Données

1)

La clé primaire que l'on peut utiliser dans la table est « *enseigne* » car elle est universelle est permet de rechercher pour toutes les tables. Sinon on peut se servir de la clé « *marque* » qui est propre à la table Marques.

2)

Ceci n'est pas possible car la clé « *enseigne* » est commune à toutes les tables. Pour palier à ce problème on pourrais créer la clé « *Qualifie* » qui contiendras les nom des magasins. Cette clé peut sinon contenir un booléen (oui/non) indiquant s'il qualifie.

3)

La requête pour afficher la liste organisées par le magasins Leclerc est :

code

```
"""Je vais me permettre de poser quelques petits éléments pour faciliter la compréhension du code
Le tableau Promotions (ainsi que ses voisins) sont des fichiers csv en .csv
Les données dans les tableau sont séparées par: |
Chaque ligne est déterminée par un saut de ligne (ici le symbole "\n")
La première ligne des chaque tableau défini les titres des colonnes
\t correspond à tab (soit ("	"))
La colonne 0 de chaque tableau est id
La construction du tableau Promotions est:
id	enseigne	nom	date_debut	date_fin"""

f=open("Promotions.csv","r")
content=f.read()
f.close()
Table={}
Temp=[]
colums=[]
word=""
for i in content:
	if b==0 and i=="\n":
		for i in range(len(colums)):
			Table[str(colums[i])]=[]
		b=1
	elif b==0 and i=="|":
		colums.append(word)
		word=""
	elif b==1 and i=="\n":
		for i in range(len(Temp)):
			Table[colums[i]].append(Temp[i])
		Temp=[]
	elif b==1 and i=="|":
```

```
                Temp.append(word)
                word=""
        else:
                word+=str(i)

print("id\t enseigne\t nom")
for i in range(len(Table[colums[0]])):
        if Table[colums[1]][i]=="Leclerc":
                print(i,"\t",Table[colums[1]][i],"\t",Table[colums[2]][i])
```

4)

La requête pour afficher les sites pratiquant la vente en ligne est :

code

```
"""Je vais me permettre de poser quelques petits éléments pour faciliter la compréhension du code
Le tableau Promotions (ainsi que ses voisins) sont des fichiers csv en .csv
Les données dans les tableau sont séparées par: |
Chaque ligne est déterminée par un saut de ligne (ici le symbole "\n")
La première ligne des chaque tableau défini les titres des colonnes
\t correspond à tab (soit ("	"))
La colonne 0 de chaque tableau est id
La construction du tableau Magasins est:
id	enseigne	adresse	code_postal	ville	en_ligne"""

f=open("Magasins.csv","r")
content=f.read()
f.close()
Table={}
Temp=[]
colums=[]
word=""
for i in content:
	if b==0 and i=="\n":
		for i in range(len(colums)):
			Table[str(colums[i])]=[]
		b=1
	elif b==0 and i=="|":
		colums.append(word)
		word=""
	elif b==1 and i=="\n":
		for i in range(len(Temp)):
			Table[colums[i]].append(Temp[i])
		Temp=[]
	elif b==1 and i=="|":
```

```
                Temp.append(word)
                word=""
        else:
                word+=str(i)

print("id\t adresse\t en ligne")
for i in range(len(Table[colums[0]])):
        if Table[colums[5]][i]=="1":
                print(i,"\t",Table[colums[2]][i],"\t",Table[colums[5]][i])
```

5)

Je n'ai pas réussis à répondre à cette questions. J'ai eu trop de mal au niveau de la réflexion. Cette erreur est potentiellement due à une erreur de compréhension.

6)

La requête pour afficher les sites ayant actuellement des promotions est :

code

```
"""Je vais me permettre de poser quelques petits éléments pour faciliter la compréhension du code
Le tableau Promotions (ainsi que ses voisins) sont des fichiers csv en .csv
Les données dans les tableau sont séparées par: |
Chaque ligne est déterminée par un saut de ligne (ici le symbole "\n")
La première ligne des chaque tableau défini les titres des colonnes
\t correspond à tab (soit ("	"))
La colonne 0 de chaque tableau est id
La construction du tableau Promotions est:
id	enseigne	nom	date_debut	date_fin"""

f=open("Promotions.csv","r")
content=f.read()
f.close()
Table={}
Temp=[]
colums=[]
word=""
for i in content:
	if b==0 and i=="\n":
		for i in range(len(colums)):
			Table[str(colums[i])]=[]
		b=1
	elif b==0 and i=="|":
		colums.append(word)
		word=""
	elif b==1 and i=="\n":
		for i in range(len(Temp)):
			Table[colums[i]].append(Temp[i])
		Temp=[]
	elif b==1 and i=="|":
```

```
                Temp.append(word)
                word=""
        else:
                word+=str(i)
today=NOW()
print("id\t enseigne\t nom\t date de début\t date de fin")
for i in range(len(Table[colums[0]])):
        if Table[colums[3]][i]>=today and Table[colums[4]][i]<=today:
                print(i,"\t",Table[colums[1]][i],"\t",Table[colums[2]][i],"\t",Table[colums[3]][i],"\t",Table[colums[4]][i])
```

7)

La fonction SQL ajoutera les éléments la liste count du tableau Marques dans une nouvelles colonne count (créée pour l'occasion) dans le tableau Magasins.

8)

La commande pour effacer le magasin chez Leon à Bayonne est :

code

```
"""Je vais me permettre de poser quelques petits éléments pour faciliter la compréhension du code
Le tableau Promotions (ainsi que ses voisins) sont des fichiers csv en .csv
Les données dans les tableau sont séparées par: |
Chaque ligne est déterminée par un saut de ligne (ici le symbole "\n")
La première ligne des chaque tableau défini les titres des colonnes
\t correspond à tab (soit ("	"))
La colonne 0 de chaque tableau est id
La construction du tableau Magasins est:
id	enseigne	adresse	code_postal	ville	en_ligne"""

f=open("Magasins.csv","r")
content=f.read()
f.close()
Table={}
Temp=[]
colums=[]
word=""
for i in content:
	if b==0 and i=="\n":
		for i in range(len(colums)):
			Table[str(colums[i])]=[]
		b=1
	elif b==0 and i=="|":
		colums.append(word)
		word=""
	elif b==1 and i=="\n":
		for i in range(len(Temp)):
			Table[colums[i]].append(Temp[i])
		Temp=[]
	elif b==1 and i=="|":
```

```
                Temp.append(word)
                word=""
        else:
                word+=str(i)

print("id\t adresse\t en ligne")
for i in range(len(Table[colums[0]])):
        if Table[colums[1]][i]=="Chez Léon" and Table[colums[4]][i]== "Bayonne":
                for b in range(len(colums)) :
                        Table[colums[b]].pop(i)
#actualisation de la table
f=open("Magasins.csv","w")
for i in range(len(colums)):
        if i<len(colums)-1:
                f.write("{}|".format(colums[i]))
        else:
                f.write("{}\n".format(colums[i]))
for i in range(len(Table[colums[0]])):
        for b in range(len(colums)):
                if i<len(colums)-1:
                        f.write("{}|".format(Table[colums[b]][i]))
                else:
                        f.write("{}\n".format(Table[colums[b]][i]))
f.close()
```

9)

La requête « *UPDATE Magasins SET en_ligne=1* » permet de changer la valeur *en_ligne* à *1*, la ligne suivante « *WHERE ville IS NULL* » indique que la colonne à chercher est ville, et plus précisément, l'élément à chercher est « *NULL* ». En cherchant cela, le programme connais la ligne sur laquelle il doit modifier la valeur de la colonne « *en_ligne* ».

Problème 3 : (Programmation fonctionnelle)

Problème 4 : (Diviser pour Reigner)

ma_liste=[1,4,7,8,9,27,28,30,32,99,105,110,115,200,210,220]

1)

Graph Log, voir copie

2)

La méthode naïve aurai bouclé n fois soit (dans notre cas) comme la liste a 200 éléments, la méthode naïve aurais bouclé 200 fois.

3)

Cet algorithme est très efficace car il n'a bouclé que sept fois comparé à la méthode naïve.

Méthode dichotomique version récursive

1)

La fonction **recherche_dicho1**(*T,x,gauche,droite*) est récursive car elle s'auto appelle créant ainsi le principe de boucle.

2)

La fonction intermédiaire n'est pas récursive car elle ne s'auto appelle pas.

3)

Les paramètre à indiquer à la fonction recherche_dicho1 son :

resultat=**recherche_dicho1**(*ma_liste,val,0,len(ma_liste)-1*)

4)

Code
def **dicho1**(*T,X,milieu,gauche,droite*): if x>T[milieu]:

```
            print(T[milieu])
            res=recherche_dicho1(T,X,milieu+1,droite)
            print("Droite=",droite)
            print("retour",res)
            return res
        else:
            print(T[milieu])
            res=recherche_dicho1(T,X,gauche,milieu-1)
            print("Gauche=",gauche)
            print("retour",res)
            return res
```

Resultat pour ma_liste=[1,4,7,8,9,27,28,30,32,99,105,110,115,200,210]

et appellation : recherche_dicho1(ma_liste,val=115,0,len(ma_liste)-1)