Concours blanc d'informatique
Calculatrice interdite
Durée : 3heures

Partie 1

On s'intéresse à l'évolution d'une population de poissons dans un océan.
On note $p(t)$ un réel dépendant du temps qui représente le nombre de poissons et on note $\frac{dp}{dt}$ la dérivée de la fonction p par rapport au temps.

I Les poissons

On admet dans un premier temps que l'équation $\frac{dp}{dt} = p - p^3$ permet de modéliser l'évolution des poissons au cours du temps, le terme p représentant la croissance exponentielle d'une population lorsque ses ressources sont illimitées et le terme $-p^3$ introduisant une saturation pour tenir compte des limites de place et de nourritures disponibles.

Q1 Expliquer en quelques phrases et à l'aide d'un schéma le principe de la méthode d'Euler pour résoudre une équation différentielle du type $y'(t) = f(t, y(t))$.

On suppose que des variables t_ini, t_fin, et N sont préalablement définies dans le programme pour représenter respectivement le temps initial, le temps final et le nombre de points de discrétisation pour le temps.

Q2 Écrire quelques lignes de codes permettant de définir la liste tab_t contenant les différents instants et le réel dt qui est le pas de discrétisation.

On note tab_p la liste contenant le nombre de poissons à chaque instant, ainsi tab_p[i] est le nombre de poissons à l'instant tab_t[i] .

Q3 Donner la relation de récurrence permettant de calculer tab_p[i+1] en fonction de tab_p[i] selon la méthode d'Euler.

Q4 À l'aide des questions précédentes, écrire une fonction ***evolution :***

- Prenant en argument 4 variables : t_ini, t_fin, N précédemment définies et p_ini le nombre de poissons à l'instant t_ini
- Qui renvoie les deux tableaux (ou listes) tab_t et tab_p.

II. Paul veut manger du chocolat

Pour bien profiter de Pâques, Paul souhaite manger beaucoup de poissons au chocolat. Son problème est qu'il ne dispose que d'un budget limité. On se propose dans ce problème d'aider Paul à optimiser ses achats pour manger la plus grande quantité de chocolat possible.
La chocolaterie dispose de n chocolats, numérotés de 0 à $n-1$. Chaque chocolat a un prix et un poids et n'est disponible qu'en un seul exemplaire. Par exemple :

n	Nom	prix	poids
0	Gros poisson	50 €	500 g
1	Petits œufs de Pâques	10 €	50g
2	Petit poisson	30 €	250 g
3	Baleine bleue	190 €	1500 g

En termes de représentation des données, les chocolats disponibles seront représentés par deux listes : L_prix et L_poids de n éléments. Dans le cas ci-dessus, on aurait :

L_prix = [50, 10, 30,190] et L_poids = [500, 50, 250, 1500]

Une liste d'achat de Paul sera représentée par une liste de n éléments contenant des 0 ou des 1 suivant qu'il achète l'objet ou non.

Par exemple L_achat =[1, 0, 0, 1] signifierait que Paul achète un gros poisson, pas de petits œufs de Pâques, pas de petit poisson, et qu'il achète une baleine bleue.

1 Quelques premières fonctions

Q5 Écrire une fonction ***poidsEtPrix(L_prix, L_poids, L_achat)*** prenant en argument les trois listes précédemment définies et renvoyant deux réels représentant le prix à payer par Paul pour sa liste d'achat ainsi que le poids total de chocolat. Dans l'exemple ci-dessus, votre fonction devrait renvoyer 240, 2000.

Q6 En fonction de n, le nombre d'objets dans la chocolaterie, quelle est la complexité de votre fonction?

Q7 Écrire une fonction ***meilleureListe*** qui prend en arguments : les deux listes de prix et de poids précédentes, une liste de listes d'achat à tester, et un réel *epargne* représentant l'argent que possède Paul. La fonction devra renvoyer la meilleure liste d'achat pour Paul,

c'est-à-dire la liste qui permet à Paul d'acheter la plus grande quantité de chocolat avec son épargne. (On ne prendra pas en compte le cas où aucune liste n'est possible et en cas d'égalité entre plusieurs listes d'achat, il suffit de renvoyer une des listes de poids maximum).
Par exemple avec les trois listes précédemment utilisées dans les exemples :

meilleureListe(L_prix, L_poids,[[0,0,0,0], [1,1,1,1], [1,0,0,0]], 200)

devra renvoyer : [1,0,0,0].

En effet :
1. la liste d'indice 0 est [0,0,0,0], elle correspond à un poids nul et à un prix nul
2. la liste d'indice 1 est [1,1,1,1], elle correspond à un poids de 2300 g et à un prix de 280€, elle doit donc être éliminée car hors budget (280>200)
3. la liste d'indice 2 est [1,0,0,0], elle correspond à un poids de 500 g et à un prix de 50 €, c'est donc la meilleur des trois listes pour Paul.

Q8 Estimer la complexité de votre fonction ***meilleureListe*** en fonction de N ,le nombre de listes d'achat proposée, et de n , le nombre d'objets dans la chocolaterie.

2. Paul est une brute

Pour trouver la configuration optimale, Paul décide d'employer une méthode « force brute », c'est-à-dire de tester toutes les possibilités. Une première étape est de générer toutes les possibilités de listes d'achat que l'on stockera dans une liste, il s'agira donc d'une liste de listes.
Ainsi, dans le cas où l'on a qu'un objet, une liste de toutes les possibilités de liste d'achat est [[0], [1]].
Si l'on a deux objets, une liste possible est :

[[0, 0], [1, 0], [0, 1], [1, 1]]

On propose un algorithme itératif pour créer la liste de toutes les possibilités en remarquant que la liste des possibilités pour $n+1$ objets peut s'obtenir facilement à partir de la liste des possibilités pour n objets. En effet, il suffit de reprendre la liste pour n objets et d'ajouter des 0 de chacune des listes possibles, de reprendre à nouveau la liste des possibilités, d'ajouter un 1 à la fin de chacune des listes et de concaténer les deux listes ainsi obtenues.

Par exemple pour obtenir la liste des possibilités pour deux objets à partir de la liste des possibilité pour un seul : on garde en mémoire la liste pour un objet : [[0], [1]] à pour chaque possibilité d'achat, on ajoute à la fin 0, ce qui donne la liste [[0,0], [1,0]] on reprend la liste [[0], [1]], à chaque possibilité on ajoute à la fin 1, ce qui donne la liste [[0,1], [1,1]] on

concatène les deux listes ce qui donne le résultat : [[0, 0], [1, 0],[0, 1], [1, 1]]
On propose pour cela un squelette de code :

```
1  from copy import deepcopy #permet de copier des listes de liste
2  def suivant(L):
3      n = len(L)
4      L1 = deepcopy(L)

5      L2 = deepcopy(L)
6      for j in range(n):
7          #modifie la j-ième liste de L1
8          #modifie la j-ième liste de L2
9      L3 = L1 + L2
10     return L3
```

Q9. Compléter les lignes 7 et 8 ?

Q10. Écrire une fonction ***toutesLesListes(n)*** créant la liste de toutes les possibilités pour n objets.
!!! Attention à l'initialisation de la liste au début de l'algorithme.

Q11. En fonction de n, combien y-a-t -il de listes d'achat possibles? (chaque objet peut-être acheté ou non)

Q12. À l'aide des fonctions précédentes créer une fonction ***bonAppetit(L_prix, L_poids, epargne)*** qui renvoie la meilleure liste d'achat pour Paul compte tenu de son épargne (en cas d'égalité entre plusieurs listes d'achat, il suffit de renvoyer une des listes de poids maximum).

3. Paul est un Glouton

Paul n'étant pas patient, il décide d'opter pour une stratégie non optimale, mais donnant un résultat de façon beaucoup plus rapide : une stratégie gloutonne. Il s'agit pour cela d'utiliser en premier les objets les plus efficaces, c'est-à-dire les chocolats offrant le meilleur rapport qualité/prix (ou plutôt poids/prix selon les critères de Paul).
Pour cela, Paul va d'abord créer une liste contenant le prix au gramme de chaque chocolat proposé.

Q13 Écrire une fonction ***qualitePrix(L_prix, L_poids)*** qui renvoie la liste où l'élément d'indice n est le prix au gramme du chocolat d'indice n. Par exemple pour la chocolaterie du début du problème, la fonction devrait renvoyer [0.1, 0.2, 0.12, 0.1267]

On suppose que l'on dispose maintenant de listes triées par prix au gramme croissant. Les listes sont donc maintenant dans le cas de notre exemple L_prix = [50, 30, 190, 10] et L_poids = [500 , 250, 1500, 50].

L'algorithme consiste désormais à commencer par acheter les objets présentant le meilleur rapport poids/prix, et de prendre ensuite les objets moins « efficaces ».
On n'achète bien entendu l'objet que si le budget restant est suffisant.

Q14. Proposer une fonction ***glouton(L_prix, L_poids, epargne)*** prenant en argument des listes que l'on suppose triées par rapport poids/prix et renvoyant une liste d'achat obtenue avec un tel algorithme.

Partie 2 : Bases de données

Dans un lycée proposant des formations CPGE, les enseignants souhaitent pouvoir suivre le parcours des élèves avec une base de données.
- Le lycée dispose de 2 classes de MPSI, 1 MP, 1 MP*, 1 PC, 2 PCSI, 1 PSI, 1 TSI1, 1 TSI2 et une ATS,
- Les élèves sont dans une classe en 1ère année (ex PCSI), puis passent dans une classe de 2ème année (par ex PSI),
- Les élèves peuvent être boursiers certaines années,
- Certains élèves arrivent directement dans une classe de 2ème année après avoir fait une première année ailleurs,
- Des élèves 3/2 souhaitant retenter le concours peuvent refaire une deuxième année,
- En vue de statistiques, les nouveaux élèves renseignent leur moyenne au bac ainsi que leur lycée d'origine,
- Les élèves passent des concours blanc et des vrais (CCP, CS, Mines-Pont,...), on souhaite conserver la moyenne qu'ils ont obtenu,
- En fonction de leurs résultats aux concours ou sur l'évaluation de leur dossier, ils poursuivent leurs études dans une école d'ingénieur ou se réorientent dans un cycle universitaire.

Un extrait de cette base est donné en Annexe.

1°) Ecrire la requête SQL qui permet de donner la liste des nom et prénom de tous les élèves de la table ELEVE.

2°) Ecrire la requête SQL qui permet de donner le nombre de garçons de la table ELEVE.

3°) Ecrire la requête SQL qui permet de donner la liste des lycées d'origine triés par ordre alphabétique de ville, puis de nom.

4°) Ecrire la requête SQL qui permet de donner la liste des nomes des élèves de la table ELEVE venant du lycée Eiffel .

LYCEE

id_lycee	nom	ville
1	inconnu	inconnue
2	Richelieu	Rueil
3	Eiffel	Dijon
4	Robert Doisneau	Corbeil-Essonnes
5	Lombards	Troyes
6	Thiers	Marseille
7	Jean Bart	Dunkerque
8	Descartes	Tours
9	Louis Le Grand	Paris
10	Saint Joseph	Fort-de-France
11	Dorian	Paris
12	Pierre de Fermat	Toulouse
13	Eiffel	Cachan
14	Chrétien de Troyes	Troyes

CLASSE

id_classe	nom	niveau
1	TSI1	1
2	TSI2	2
3	PCSI	1
4	MP	2
5	PTSI	1
6	PSI	2
7	ATS	2

ECOLE

id_ecole	acronyme	nom	ville
1	ENS	Ecole Normale Supérieure	Cachan
2	Mines	Ecole des Mines	Douai
3	ESIGETEL	École Supérieure d'Ingénieurs en Informatique et	Villejuif
4	ENS	Ecole Normale Supérieure	Lyon
5	Centrale	Ecole Centrale des Arts et Manufactures	Paris
6	Centrale	Ecole Centrale des Arts et Manufactures	Marseille
7	Supelec	Ecole Supérieure d'Electricité	Gyf-Sur-Yvette
8	ENI	Ecole Nationale D'ingénieurs	Brest
9	ENSI	Ecole Nationale Supérieure d'Ingénieur	Poitiers
10	Mines	Ecole des Mines	Nancy

CONCOURS

id_concours	nom
1	Concours Blanc
2	Concours Centrale-Supelec
3	Concours CCP
4	Dossier

ELEVE_CLASSE

id_eleve	id_classe	annee	boursier
11	1	2013	1
1	1	2011	0
1	2	2012	1
1	2	2013	1
2	2	2013	0
3	3	2012	1
3	6	2013	0
4	3	2011	0
4	6	2012	0
4	6	2013	0
5	4	2011	1
6	5	2012	0
7	7	2013	1
8	4	2012	0
9	1	2011	1
9	2	2012	0
12	5	2010	1
13	4	2012	0
14	6	2013	1

ELEVE

id_eleve	nom	prenom	email	sexe	adresse	telephone	id_lycee	note_moyenne_bac
1	Lindien	Joe		0	USA	0000000000	1	4,4
2	Curie	Marie		1	Rue du radon qui tue	0868686868	3	19,5
3	Landers	Mark		0	Stade de la Toho	0125	2	5,8
4	Zidane	Zinedine		0	Rue de la bouilabesse à Marseille		4	9,1
5	Deniro	Robert		0	Coca Cola Rd, Los Angeles		7	12,8
6	Chirac	Jacques		0	Rue de l'honnetété à Paris	028765427	4	15,4
7	Jackson	Mickael		0	Bld de l'immortalité à Sacramento		3	17
8	Donna	Ma		1	1 Rue de la retraite à Miami		2	6,3
9	Ethutch	Starsky		0	Nananana Terrrace, New York	0102030405	6	13,3
10	Carey	Mariah	mariah@carey.com	1	Rue des octaves à Poissy		2	9,9
11	Einstein	Albert		0			6	17
12	Ross	Diana		1			7	13,7
13	Enement	Eve		1			8	13,7
14	Parks	Rosa		1			3	15,97
15	Simpson	Marge		1			9	14,6