Concours Blanc :corrigé

PTSI Lycée Eiffel

3 juin 2019

Première partie : base de données des animaux du zoo.

1. CREATE TABLE animaux (n°id INTEGER PRIMARY KEY, espece VARCHAR(30), nom VARCHAR(30), age INTEGER, Date DATE)
2. Dans la mesure où il est tout à fait possible qu'un même employé effectue plusieurs types de soins différents sur un même animal, on ne peut pas se contenter de prendre par exemple le coupe constitué des numéros d'identifiants de l'employé et de l'animal comme clé primaire. En fait, le plus simple est de créer de toute pièce un numéro qu'on appellera typiquement n°soin. Les clés secondaires seront le n°employe du soigneur et le n°id de l'animal, qui doivent apparaitre comme attribut dans cette table.
3. SELECT nom, age FROM animaux WHERE espece='éléphant'
4. SELECT COUNT(nticket) FROM ticket WHERE tarif='plein tarif' AND date='2018/05/12'
5. SELECT Employés.nom, prenom FROM (Employés INNER JOIN Soins ON Employés.n°employe=Soins.n°employe) INNER JOIN animaux ON animaux.n°id=Soins.n°id WHERE animaux.nom='Baloo'

Deuxième partie : exploitation d'un fichier de données et programmation Python.

1. La syntaxe est f=open('fichier.txt','m'), pour créer une variable f qui pointe vers le fichier texte nommé fichier.txt. Le caractère m doit être remplacé par une des trois possibilités suivantes : 'r' pour lire le fichier sans le modifier, 'a' pour ajouter du contenu à la fin du fichier (sans modifier ce qui était déjà dans le fichier) ou 'w' pour écrire dans un fichier vierge.
2. L'énoncé racontait n'importe quoi, puisque split est une méthode et non une fonction. On va respecter la syntaxe Python dans le programme qui suit :

```
def listealpha() :
    l=[]
    f=open('animaux.txt','r')
    for i in f.readlines() :
        a=i.split(';')
        l.append(a[1])
    l.sort()
    return l
```

3. On ajoute la ligne suivante entre le split et le append du programme précédent :

```
if a[2]<=5 :
```

et on indente bien sûr la ligne avec le append.

Troisième partie : étude de la température du logement des girafes (programmation Python).

1. Les nombres flottant sont codés en complément à 2 : sur 16 bits par exemple, on peut coder les entiers de -2^{15} jusqu'à 2^{15-1}, chaque entier binaire b compris entre 2^{15} et $2^{16}-1$ codant l'entier négatif $b-2^{16}$. Ainsi, le premier bit du codage correspond à un bit de signe (tous le entiers positifs commencent par un 0 et les entiers négatifs par un 1). Voir le cours pour plus de détails.
2. Justement, 2 octets correspond à 16 bits, soit 2^{16} valeurs possibles. On sait que $2^{10} \simeq 1000$, donc $2^{16} \simeq 64\ 000$, ce qui laisse espérer quatre chiffres significatifs corrects en écriture décimale.
3. Oui, puisque quatre chiffres significatifs suffisent à priori.
4.
```
import matplotlib.pyplot as plt
plt.plot(horaires, temperatures)
plt.show()
```
5.
```
def moyenne(L) :
    s=0
    for i in L :
        s=s+i
    return s/len(L)
```
6.
```
def variance(L) :
    m=moyenne(L)
    a=0
    for i in L :
        a=a+(i-m)**2
    return a/len(L)
```

   Le programme effectue une fois le calcule de la moyenne, ce qui nécessaite $n-1$ additions et une division. Ensuite on effectue à nouveau dans la boucle $n-1$ additions, plus n soustractions et n élévations au carré (donc des multiplications) et enfin une division. Globalement l'algorithme est linéaire (nombre d'opérations élémentaires proportionnel à n).

7.
```
def checktemp(L) :
    m=moyenne(L)
    if m<25 or m>28 :
        return False
    if variance(L)>3 :
        return False
    return True
```
8.
```
def detectionanomalies(h,t) :
    n=0
    l=[]
    for i in range(len(t)) :
        if t[i]>=22 :
            n=0
        else :
            n=n+1
            if n==3 :
                l.append(h[i])
    return l
```

Quatrième partie : étude de la courbe de croissance d'un bébé panda (analyse numérique).

1. Si $\beta = 0$, on a $P'(t) = \alpha P(t)$, équation différentielle dont les solutions sont des fonctions exponentielles. Si on veut que le poids du bébé soit croissant, on va donc imposer $\alpha > 0$, mais il faudra prendre $\beta < 0$ pour éviter que le poids ne grandisse trop violemment.
2. Je vous laisse relire le cours, mais en gros, on découpe l'intervalle de résolution en n intervalles de même largeur, et on approche sur chacun de ces intervalles la courbe par une tangente approchée dont la pente est obtenue à partir de la valeur approchée calculée au point précédent, en exploitant bien sûr l'équation différentielle qui relie la fonction et sa dérivée.
3.
```
import matplotlib.pyplot as plt
def simulpanda(a,b) :
    temps,poids=[0],[0.1]
    for i in range(1000) :
        poids.append(poids[-1]+0.01*(a*poids[-1]+b*poids[-1]**2))
        temps.append(temps[-1]+0.01)
    plt.plot(temps,poids)
    return poids
```
4. On a utilisé dans le programme précédent un découpage qui renvoie 1 000 valeurs du poids pour les 10 premiers jours, il faut donc extraire les 10 valeurs qu'on veut comparer avec le vrai poids mesuré :
```
from math import abs
def valide(a,b,poids) :
    theorie=simulpanda(a,b)
    for i in range(1,11) :
        if abs(poids[i]-theorie[100*i-1])/poids[i]>0.05 :
            return False
    return True
```
5. (a)
```
def approxder(poids,delta)
    l=[]
    for i in range(1,len(poids)-1) :
        l.append((poids[i-1]+poids[i+1])/(2*delta))
    return l
```
   (b) Il faut enlever les valeurs extrêmes de la liste poids pour avoir le même nombre d'éléments que dans l'autre liste, puis faire une approximation polynomiale de degré 2. Cette approximation va nous ressortir trois coefficients, on se contentera de prendre les deux derniers pour nos valeurs de α et β (si le modèle n'est pas pourri, le coefficient constant devrait de toute façon être proche de 0).

```
P=polyfit(approxder(poids,delta),poids[1 : :-1],2)
(a,b)=(P[1],P[2])
```