Corrigé du Devoir Surveillé d'Informatique n°2

Exercice 1 — Un petit pot pourri...

Aucune cohérence dans cet exercice, ce sont juste des extraits de vos TP !

1. Créer un programme prenant en entrée un réel x et renvoyant $g(x)$ avec : $g : \begin{cases} \mathbb{R} & \to \mathbb{R} \\ x & \mapsto \begin{cases} \sqrt{x} & \text{si } x > \pi \\ 3x^2 & \text{sinon} \end{cases} \end{cases}$

   Réponse : On propose ce programme suivant :

```
import math as m
def g(x):
    if  x>=m.pi:
        return m.sqrt(x)
    else:
        return 3*x**2
```

2. Écrire un programme capable de déterminer si l'année entrée par l'utilisateur est une année bissextile ou pas. On rappelle qu'une année est bissextile si elle est divisible par 4 mais pas par 100, ou si elle est divisible par 400. Par exemple, 2000 est bissextile mais pas 1900.

   Réponse : On propose ce programme suivant :

```
def bissextile(N):
    if N%4==0 and N%100!=0:
        return("Ann e bissextile")
    elif N%400==0:
        return("Ann e bissextile")
    return("Ann e pas bissextile")
```

3. Écrire un programme qui demande à l'utilisateur de donner un entier compris entre 0 et 100 puis choisit au hasard un entier n entre 0 et 100 et qui répond "Trop grand" si l'entier proposé par le joueur est strictement supérieur à n, "Trop petit" si l'entier proposé par le joueur est strictement inférieur à n et enfin "Bien joué" si l'entier proposé par le joueur est n. On rappelle que la fonction `randint` que l'on trouve dans la sous bibliothèque random de la bibliothèque numpy prend en entrée un entier naturel n et renvoie un entier choisi au hasard entre 0 et n.

Réponse : On propose ce programme suivant :

```
import numpy as np

def cherchenombre():
    nb=eval(input("Que vaut le nombre myst re?"))
    a=np.random.randint(100)
    if nb==a:
        print("Bien jou ")
    elif nb<a:
        print("trop petit")
    else:
        print("trop grand")
```

4. (a) Écrire une fonction prenant en paramètre n un entier naturel et affichant les valeurs de 5^0 puis 5^1 puis 5^2 puis ... jusqu'à 5^n.

Réponse : On propose ce programme suivant :

```
def puiss5(n):
    for k in range(n+1):
        print(5**k)
```

(b) Écrire une fonction prenant en paramètre n un entier naturel et affichant les valeurs de 5^0 puis 5^1 puis 5^2 puis ... jusqu'à obtenir une valeur de 5^k, k entier naturel, tel que $5^k > n$.

Réponse : On propose ce programme suivant :

```
def puiss5(n):
    k=0
    while 5**k<=n:
        print(5**k)
        k=k+1
```

(c) Écrire une fonction prenant en paramètre n un entier naturel et retournant dans une liste les valeurs de 5^0 puis 5^1 puis 5^2 puis ... jusqu'à 5^n.

Réponse : On propose ce programme suivant :

```
def Lipuiss5(n):
    L=[]
    for k in range(n+1):
        L.append(5**k)
    return L
```

5. Soient $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ les suite définies par :

$$u_0 = 5, v_0 = 5 \text{ et } \forall n \in \mathbb{N},\ u_{n+1} = 2u_n + 1,\ v_{n+1} = 2v_n + n.$$

(a) Écrire une fonction sans entrée et sans sortie, qui demande explicitement à l'utilisateur un entier naturel n puis qui affiche les n premiers termes de la suite $(u_n)_{n\in\mathbb{N}}$.

Réponse : On propose ce programme suivant :

```
def suiteu():
    u=5
    n=eval(input('Quel rang veux-tu?'))
    for _ in range(n):
        print(u)
        u=2*u+1
```

(b) Écrire une fonction qui prend en entrée un entier naturel n et qui renvoie v_n.
Réponse : On propose ce programme suivant :

```
def suitev(n):
    v=5
    for k in range(n):
        v=2*v+k
    return v
```

(c) Écrire une fonction qui prend en entrée un entier naturel non nul n et qui renvoie la liste $[u_1, \cdots, u_n]$.
Réponse : On propose ce programme suivant :

```
def Lsuiteu(n):
    u=5
    L=[]
    for _ in range(n):
        u=2*u+1
        L.append(u)
    return L
```

Exercice 2 — A la recherche des petits !

Les petits ont toujours été populaires. C'est pourquoi nous allons tenter, dans cet exercice, de les chercher. **On vous demande dans cet exercice de ne pas utiliser de méthodes sur les listes (insert, reverse, remove, sort...) à part la méthode append, ni les fonctions `max` et `min`.**

1. Écrire une fonction `minimum` qui prend en entrée une liste de nombres (entiers ou flottants) et qui renvoie le minimum de cette liste.
   Réponse : Voici une fonction `minimum` qui prend en entrée une liste de nombres (entiers ou flottants) et qui renvoie la première occurrence du minimum de cette liste :

```
def minimum(li):
    m = li[0]
    for x in li:
        if x < m:
            m = x
    return m
```

2. (a) Écrire une fonction `indiceminimum` qui prend en entrée une liste de nombres (entiers ou flottants) et qui renvoie l'indice de la première occurrence du minimum de cette liste. Par exemple, `indiceminimum([1, 14, 2, 1, 5, 1, 98])` renverra 0 et pas 5.

   Réponse : Voici une fonction `indiceminimum` qui prend en entrée une liste de nombres (entiers ou flottants) et qui renvoie l'indice de la première occurrence du minimum de cette liste :

```
def indiceminimum(L):
    i, m = 0, L[0]
    for k in range (1, len(L)):
        if L[k] < m:
            i, m = k, L[k]
    return i
```

   (b) Écrire une fonction `occurrencesmin` qui prend en entrée une liste de nombres et qui renvoie la liste des indices de toutes les occurrences du minimum dans la liste. Par exemple, `occurrencesmin([1, 14, 2, 1, 5, 1, 98])` renverra la liste `[0, 3, 5]`.

   Réponse : Voici une fonction `occurrencesmin` qui prend en entrée une liste de nombres et qui renvoie la liste des indices de toutes les occurrences du minimum dans la liste :

```
def occurrencesmin(L):
    m = minimum(li)
    lmin=[]
    for k in range (len(L)):
        if L[k] == m:
            lmin.append(k)
    return lmin
```

3. Écrire une fonction `tiensunminimum` qui prend en entrée une liste de nombres et qui remplace dans la liste toutes les occurrences du minimum par la chaîne de caractères `"oh, un petit, qu'il est chou!"`. Notons que la fonction modifie directement la liste prise en entrée. Par exemple, si `L = [1, 14, 2, 1, 5, 1, 98]`, alors après exécution de l'instruction `tiensunminimum(L)`, la liste L devient :

```
["oh, un petit, qu'il est chou!", 14, 2, "oh, un petit, qu'il est chou!",
5, "oh, un petit, qu'il est chou!", 98]
```

   Réponse : Voici une fonction `tiensunminimum` qui prend en entrée une liste de nombres et qui remplace dans la liste toutes les occurrences du minimum par la chaîne de caractères `"oh, un petit, qu'il est chou!"` :

```
def tiensunminimum(L):
    l=occurrencesmin(L)
    for x in l:
        L[x]="oh, un petit, qu'il est chou!"
```

4. Écrire une fonction `insere` prenant en entrée trois arguments : une liste L, un entier naturel k cohérent par rapport à la taille de la liste et un élément x (de type quelconque), qui crée une

liste l qui est la même que L mais avec l'élément x inséré juste après l'élément d'indice k. Par exemple, si `L = [1, 14, 2, 1, 5, 1, 98]`, alors l'instruction : `insere(L, 3, "Nouveau !")` renvoie :

```
[1, 14, 2, 1, "Nouveau !", 5, 1, 98]
```

Réponse : Voici une fonction `insere` prenant en entrée trois arguments : une liste L, un entier k et un élément x (de type quelconque), qui insère l'élément x dans la liste L juste après l'élément d'indice k :

```
def insere(L,k,x):
    l=[]
    for i in range (k+1):
        l.append(L[i])
    l.append(x)
    for i in range (k+1, len(L)):
        l.append(L[i])
    return l
```

5. Écrire une fonction `titi` qui prend en entrée une liste L de nombres et qui insère dans la liste L la chaîne de caractères `"J'ai cru voir un gros min."` après chaque occurrence du minimum de la liste L. Par exemple, si `L = [1, 14, 2, 1, 5, 1, 98]`, alors, après exécution de l'instruction `titi(L)`, la liste L devient :

```
[1, "J'ai cru voir un gros min.", 14, 2,
1, "J'ai cru voir un gros min.", 5,
1, "J'ai cru voir un gros min.", 98]
```

Réponse : Voici une fonction `titi` qui prend en entrée une liste L de nombres et qui insère dans la liste L la chaîne de caractères `"J'ai cru voir un gros min."` après chaque occurrence du minimum de la liste L :

```
def titi(L):
    l=occurrencesmin(L)
    k=1
    for i in l:
        L=insere(L,i+k,"J'ai cru voir un gros min.")
        k+=1
    return(L)
```