# Corrigé du Devoir Surveillé d'Informatique n°4

Les candidats ne doivent faire usage d'aucun document, l'utilisation de toute calculatrice et de tout matériel électronique est **interdite**. Dans l'écriture de vos programmes en Python, **respectez la ponctuation et l'indentation.**

**Exercice 1 — Un petit pot pourri...**

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

1. Écrire une fonction prenant en entrée un entier naturel $n$ et retournant $\sum_{k=0}^{n} k^7$ avec une boucle.

   **Réponse :** On propose ce programme suivant :

```
def Sfor(n):
    s=0
    for k in range(n):
        s=s+(k+1)**7
    return s
```

2. Écrire une fonction récursive prenant en entrée un entier naturel $n$ et retournant $\sum_{k=0}^{n} k^7$.

   **Réponse :** On propose ce programme suivant :

```
def S(n):
    if n == 0:
        return 0
    else:
        return S(n-1)+n**7
```

3. Soit $(w_n)_{n\in\mathbb{N}}$ la suite définie par :

$$w_0 = 1 \text{ et }, \forall n \in \mathbb{N}, w_{n+1} = 12\ln(w_n) + w_n^2.$$

   Écrire une fonction qui prend en entrée un entier $n$ qui renvoie la liste $[w_0, w_1, \cdots, w_n]$.

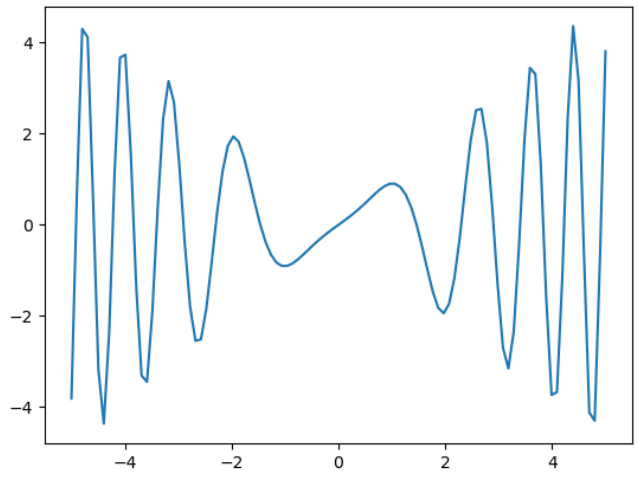
   **Réponse :** On propose ce programme suivant :

```
import math as m
def Listew(n):
    w=m.e
    L=[]
    for _ in range(n+1):
        L.append(w)
        w=2*m.log(w)+w
    return L
```

4. Écrire une fonction prenant en entrée un entier naturel non nul $n$ et traçant le graphique de la fonction $f : x \mapsto x\sin(x^2+1)$ sur l'intervalle $[-5,5]$ avec $n$ points bien répartis.
**Réponse :** On propose ce programme suivant :

```
def traceg(n):
    t= np.linspace(-5,5,n)
    c= [m.sin(a**2+1)*a for a in t]
    plt.plot(t,c)
    plt.show()
```

On a obtenu :




5. On considère l'équation différentielle (E) : $y'(t) = \cos(t) \times y^2(t)$ d'inconnue $y$ fonction dérivable sur $\mathbb{R}_+$. Soit $\varphi$ la solution de (E) telle que $\varphi(0) = 2$ Écrire une fonction `Euler` qui prend en entrée un entier naturel non nul $n$ et qui renvoie $g_n(5)$ avec $g_n$ la fonction approchant $\varphi$ en utilisant la méthode d'Euler sur $[0,5]$ avec $n$ points.
**Réponse :** On propose ce programme suivant :

```
def Euler(n):
    h=5/n
    u=2
    for k in range(n):
        u=u+h*u**2*m.cos(k*h)
    return u
```

6. On pose $\mathrm{I} = \displaystyle\int_0^1 \exp(-t^2)dt$. Écrire une fonction `rectangles` prenant en entrée un entier naturel $n$ non nul et renvoyant une approximation de I en utilisant la méthode des rectangles à $n$ pas.
**Réponse :** On propose ce programme suivant :

```python
import math as m
def f(t):
    return m.exp(-t**2)

def rectangles(n):
    s=0
    for k in range (n):
        s+=f(k/n)
    return s/n
```

**Exercice 2 — La clé du voyage**

Clovis part en voyage géologie avec sa nouvelle acquisition, une superbe clé USB Harrry Potter. Elle est très belle mais ne dispose que de 8 Go. Pour s'occuper le soir, Clovis souhaite la remplir de films. Il a 9 fichiers vidéo à sa disposition sur son ordinateur. Chaque fichier a une taille et a une durée, ce tableau présente les fichiers disponibles avec la taille en Go et les durées en minutes :

| **Nom** | Vidéo 1 | Vidéo 2 | Vidéo 3 | Vidéo 4 | Vidéo 5 | Vidéo 6 | Vidéo 7 | Vidéo 8 | Vidéo 9 |
|---|---|---|---|---|---|---|---|---|---|
| **Temps** | 113 | 102 | 83 | 12 | 39 | 138 | 143 | 92 | 104 |
| **Taille** | 1,7 | 2,3 | 1,2 | 0,5 | 1,1 | 2 | 0,7 | 0,8 | 1 |

Clovis souhaite remplir sa clé de façon à ce que la durée totale de films emportés soit la plus grande.

1. La durée des vidéos est-elle proportionnelle à la taille ? Pourquoi ?

   **Réponse :** La durée n'est pas proportionnelle à la taille puisque $\frac{104}{1} = 104$ et $\frac{12}{0.5} = 24$ et $104 \neq 24$. On peut expliquer ça en se disant que les fichiers sont de format différents et de qualité plus ou moins bonne.

2. Pour répondre à la question, le plus naturel est d'utiliser un dictionnaire dont les clés sont les noms des fichiers et, pour chaque clé, la valeur correspondante est un dictionnaire dont les clés sont les chaînes "Durée"et "Taille" et dont les valeurs correspondantes sont la durée en minute et la taille en Go. Écrire le dictionnaire de notre problème.

   **Réponse :** On va utiliser ce dictionnaire :

   ```python
   Dvideo={'Video_1':{'Duree':113, 'Taille':1.},'Video_2':{'Duree':102, 'Taille'
   ```

   On a écrit que les deux premiers... Il faudrait mettre les 9 vidéos.

3. Pour simplifier, on va utiliser une liste. Cette liste contiendra des listes de triplets, chaque triplet étant une liste de trois termes, le premier est le nom de la vidéo, le deuxième sa durée et le troisième sa taille. Dans tout le problème, on appellera `Lvideo` cette liste. Écrire un programme qui prend en entrée `Lvideo` et qui renvoie cette même liste complétée, chaque triplet devenant

un quadruplet avec les mêmes trois premiers termes d'origine (nom, durée, taille) puis, en quatrième position, le rapport durée/taille. Le premier élément de cette liste sera donc [Vidéo 1, 113, 1.7, 66.47058823529412].

**Réponse :** On propose ce programme suivant :

```
def complet(L):
    for l in L:
        l.append(l[1]/l[2])
    return L
```

On a obtenu :

```
>>> complet(Lvideo)
[['Video_1', 113, 1.7, 66.47058823529412, 66.47058823529412, 66.4705882352941
```

4. On souhaite trier cette liste. Compléter le programme suivant de façon à ce que la liste de nombre `liste` prise en entrée soit triée dans l'ordre décroissant :

```
def Tri(liste) :
    for i in range (len (liste)-1 , 0, -1):
        for j in range (??????????) :
            if liste [j] < ????????????:
                a=liste [j]
                liste [j] = liste [j+1]
                liste [j+1] = ??????????????
    return liste
```

**Réponse :** On corrige ainsi le programme :

```
def Tri(liste) :
    for i in range (len (liste)-1 , 0, -1):
        for j in range (i) :
            if liste [j] < liste [j + 1]:
                a=liste [j]
                liste [j] = liste [j+1]
                liste [j+1] = a
    return liste
```

5. Modifier le programme précédent de façon à ce qu'il puisse prendre en entrée `Lvideo` et renvoyer cette même liste trier par ordre décroissant du rapport durée/taille. Le premier élément de notre liste sera donc ['Video 7', 143, 0.7, 204.2857142857143] et le dernier sera ['Video 4', 12, 0.5, 24.0].

**Réponse :** On propose ce programme suivant :

```
def Trin(liste) :
    L=[l[3] for l in liste]
    for i in range (len (L)-1 , 0, -1):
        for j in range (i) :
            if L [j] < L [j + 1]:
                a=L [j]
                L [j] = L [j+1]
                L [j+1] = a
    l=[]
    for i in L:
        for k in liste:
            if k[3]==i:
                l.append(k)
```

```
                    break
        return l
```

On a obtenu :

```
>>> Trin(Lvideo)
[['Video_7', 143, 0.7, 204.2857142857143, 204.2857142857143, 204.285714285714
```

6. Écrire une fonction qui suit l'algorithme de remplissage de la clé suivant : on va donner la réponse dans la liste appelée `solution`, cette liste contiendra les noms des vidéos ajoutés. On ajoute les vidéos les unes après les autres en choisissant d'abord la vidéo ayant le meilleur rapport durée/taille puis on fait dans l'ordre des rapports durée/taille. On ajoute ces vidéos tant que la taille totale ne dépasse pas la taille maximale. On renvoie à la fin la liste `solution` ainsi que la durée totale de films mis sur la clé usb.

**Réponse :** On propose ce programme suivant :

```
def lesfilms(liste):
    liste=complet(liste)
    Nliste = Trin(liste)
    reponse = []
    totale_duree = 0
    totale_taille = 0
    i = 0
    while i < 9 and totale_taille <= 8:
        nom, dure, taille =  Nliste[i][0],Nliste[i][1],Nliste[i][2]
        if totale_taille + taille <= 8:
            reponse.append(nom)
            totale_taille += taille
            totale_duree += dure
        i += 1
    return reponse, totale_duree
```

On a obtenu :

```
>>> lesfilms(Lvideo)
(['Video_7', 'Video_8', 'Video_9', 'Video_3', 'Video_6', 'Video_1', 'Video_4'
```

7. On voudrait tester toutes les combinaisons possibles. Pour cela, on va utiliser un 9-upplets de 0 et de 1, l'élément $i$ de ce upplet vaut 1 si on a pris la vidéo $i$, 0 sinon. Ainsi, si ce upplet vaut $[0,0,1,1,1,0,1,0,1]$, cela signifie que Clovis a mis les vidéo 3, 4, 5, 7 et 9 sur sa clé. Il y a, d'après les cours de M.Poullaouec, $2^9$ combinaisons possibles. Donner un programme donnant la liste de ces $2^9$ combinaisons possibles.

**Réponse :** On propose ce programme suivant :

```
def listecombinaison():
    return [[i1,i2,i3,i4,i5,i6,i7,i8,i9] for i1 in range(2)
    for i2 in range(2)for i3 in range(2)for i4 in range(2)
    for i5 in range(2)for i6 in range(2)for i7 in range(2)
    for i8 in range(2) for i9 in range(2)]
```

On a obtenu :

```
>>> listecombinaison()
[[0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 1], [0, 0, 0, 0, 0, 0,
 1, 0], [0, 0, 0, 0, 0, 0, 0, 1, 1], [0, 0, 0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0
, 0, 1, 0, 1], [0, 0, 0, 0, 0, 0, 1, 1, 0], [0, 0, 0, 0, 0, 0, 1, 1, 1], [0, 0
```

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

On n'a pas donné tous les résultats, il y en a 512...

8. Donner un programme prenant en entrée un 9-upplets de 0 et de 1 révélant les vidéos prises cf. question précédente) et la liste `Lvideo` et renvoyant une liste de deux termes : la durée totale et la taille totale de ces vidéos.

   **Réponse :** On propose ce programme suivant :

```
def totaledureetaille(combi, Liste):
    d, t, i=0, 0, 0
    for a in combi:
        d += a*Liste[i][1]
        t += a*Liste[i][2]
        i=i+1
    return [d, t]
```

9. En déduire une fonction meilleuresolution qui prend en entrée `Lvideo`, qui teste toutes les combinaisons possibles et qui renvoie le meilleur choix possible ainsi que la durée totale.

   **Réponse :** On propose ce programme suivant :

```
def meilleuresolution(Liste):
    duree_max = 0
    solution = []
    ens_combi=listecombinaison()
    for partie in ens_combi:
        duree = totaledureetaille(partie,Liste)[0]
        taille = totaledureetaille(partie,Liste)[1]
        if taille <=8 and duree > duree_max:
            duree_max = duree
            solution = partie
    return solution, duree_max
```

On a obtenu :

```
>>> meilleuresolution(Lvideo)
([1, 0, 1, 1, 0, 1, 1, 1, 1], 685)
```

On se rend compte que c'est la même solution qu'avec le premier algorithme !