# CORRIGE CONTROLE 2 (55') REPRESENTATION DE L'INFORMATION PROGRAMMATION EN PYTHON

*Compte rendu :*

- *Représentation de l'Information :*

  *Cours pas assez su.*

- *Numérisation du son (Exo 2) :*

  *Beaucoup d'oublis du nombre de pistes dans les calculs ou de la conversion en octets.*

  *Justifiez vos calculs par une formule.*

  *La qualité d'un fichier réencodé ne peut être supérieur à la qualité du fichier original ! (Question 4)*

- *QCM Listes (Exo 3) :*

  *Assez bien réussi dans l'ensemble en 2018.*

- *Algorithmique-Programmation (Exo 4) :*

  1. *Analyse algorithmique :*

     *Analyse pas assez poussée des différents cas possibles.*

     *Appuyez votre analyse d'abord sur des …… puis faites un ……*

     *Un algorithme dans ses grandes lignes n'est pas encore un programme, inutile de trop le détailler.*

     *Par contre, commentez votre algorithme.*

  2. *Fonction :*

     *Lisez bien l'énoncé ! On demandait précisément* ***une fonction*** *avec une entrée et une sortie bien spécifiées ! Apparemment beaucoup ne savent pas ce que c'est !*

  3. *Propreté et lisibilité des lignes de code :*

     - *Respectez l'indentation.*
     - *Evitez les variables inutiles !*
     - *Evitez les réaffectations de la forme n+= 3 : quel est l'intérêt d'une écriture aussi absconse ! n = n + 3 est bien plus explicite !*

  4. *Commentaires :*

- *Ce qu'il ne faut pas faire :*

***Un commentaire n'est pas une redite en français d'une ligne de code****. Ce que la plupart font !*

*Exemple de mauvais commentaire inutile : if a > 0 : # si a est supérieur à 0.*

- *Ce qu'il faut faire :*

  - ***Un commentaire général de plusieurs lignes :***

    ***pour présenter l'analyse algorithmique et les schémas et exemples sur lesquels elle s'appuie.***

    ***pour présenter l'algorithme solution en indiquant ses « Grandes Lignes » et la structuration qui en découle.***

    ***pour commenter cet algorithme : en indiquant par exemple pourquoi il traite bien tous les cas possibles.***

    ***Ce commentaire est à écrire en dehors du programme. C'était l'objet de la question 2.***

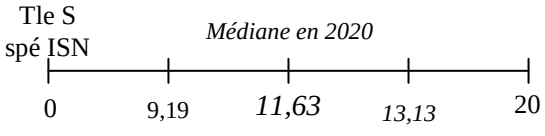
  - *Des commentaires plus ou moins facultatifs :*

    *pour expliquer des parties non évidentes du programme,*

    *pour expliciter une fonction, une variable…*

    *Ces commentaires facultatifs peuvent être insérer dans le code.*

*Médianes : 10,25 en 2019 ; 14,25 en 2018 ;*



➢ Exercice n° 1 (……..…… / 7 points) : QCM Représentations de l'Information.

Pour chaque affirmation, 3 choix vous sont proposés dont un seul est vrai. Lequel ? **L'entourer.**

Barème : réponse juste =1 pt sans réponse ou réponse fausse = 0 pt

| *Affirmations* | *Choix 1* | *Choix 2* | *Choix 3* |
|---|---|---|---|
| *① Pour convertir en base 10 un nombre écrit en base p, on peut :* | effectuer une suite de divisions euclidiennes par p. | effectuer une suite de divisions euclidiennes par 10. | additionner des puissances de p. |
| *② Un système code les entiers signés sur 8 bits. La plage d'entiers signés représentables va* | de -127 à 128 | de -128 à 127 | de -128 à 128 |
| *③ Par la technique du complémentaire à $2^n$, la représentation binaire sur 4 bits de l'opposé de 6 est* | 0110 | 1010 | 1110 |
| *④ Le charset qui contient tous les caractères utilisés sur Terre est* | UTF-8 | Unicode | UTF-32 |
| *⑤ UTF-8 code les caractères sur* | 1 octet. | 2 octets. | 1 ou 2 voire 4 octets. |
| *⑥ Les anciens modes graphiques VESA utilisent le codage couleur sur 2 octets par pixel. Le nombre de couleurs possibles est* | $2 \times 2^8$ | $16^2$ | $(2^8)^2$ |
| *⑦ Un son a été numérisé en 44 100 Hz sur 16 bits. Ce son est de qualité* | DVD | CD | Radio FM |

*① Livret Langage Binaire page 10.*

*② Livret Langage Binaire page 16 : une représentation sur N bits des entiers signés va de $-2^{N-1}$ à $2^{N-1} - 1$.*

*③ Technique du complémentaire à $2^n$ = inversion bit à bit + ajout de 1. Beaucoup d'oublis de rajout de 1.*

*0110 ça c'est bêtement 6 !*

*1110 ça c'est pour ceux qui prennent la représentation de 6 et mettent juste 1 devant : cela ne marche pas !*

*④ UTF-8 et UTF-32 ne sont pas des charsets mais des encodages.*

*⑤ Contrairement à ce que l'appellation pourrait faire croire, UTF-8 ne code pas tous les caractères sur 8 bits :*

- *La table ASCII est codée sur 8 bits pour avoir une compatibilité totale.*
- *Certains caractères (accentués par exemple) sont codés sur 2 bits.*
- *D'autres caractères comme € par exemple sont codés sur 4 octets.*

*⑥ 2 octets par pixel = 16 bits par pixel ! Donc $2^{16}$ possibilités.*

*⑦ Voir livret Numérisation p.14.*

➢ Exercice n° 2 (……..…… / 4,5 points) : Numérisation du son.



Anna Conda a téléchargé une chanson libre de droits de 3 minutes.

Cette chanson était encodée en 8 bits Mono à 16 kHz.

1. Quel est le poids en Mo du fichier téléchargé ? Justifier par un calcul. (……. / 1 pt)

$$\text{Poids du fichier téléchargé (en octets)} = \frac{\text{Fréquence} \times \text{Quantification} \times \text{Nb de pistes} \times \text{Durée (s)}}{\text{Nb de bits dans 1 octet}}$$

$$= \frac{16\,000 \times 8 \times 1 \times 3 \times 60}{8}$$

$$= 2\,880\,000 \text{ octets}$$

$$= 2{,}88 \text{ Mo}$$

*Cette chanson de 3 minutes pèse 2,88 Mo. Pas très lourd ! Compter 10 Mo/min en général, non compressé.*

2. Ce fichier est-il de bonne qualité pour une chanson ? Justifier. (…… / 0,5 pts)

*Anna aura une sensation désagréable à l'écoute : c'est un fichier de très mauvaise qualité même pas stéréo.*

*Pour rappel : qualité téléphone (8 000 hz sur 8 bits) ; qualité radio FM (22 050 hz sur 16 bits)*

*Remarque : Puisque l'oreille humaine entend jusqu'à 20 050 hz, alors, d'après le Théorème de Shanon, la fréquence d'échantillonnage devrait être au moins 2 fois cette plus haute fréquence audible, soit 41 000 hz.*

Désirant une meilleure qualité, Anna réencode le fichier téléchargé initialement en Stéréo 24 bits à une certaine fréquence. Elle obtient un fichier pesant 51,84 Mo.

3. Quelle était la fréquence d'échantillonnage du réencodage ? Justifier par un calcul. (……..… / 1 pt)

*Soit f la fréquence d'échantillonnage recherchée. On résout l'équation donnée par la condition : Poids du fichier = 51,84 Mo.*

$$\frac{\text{Fréquence} \times \text{Quantification} \times \text{Nb de pistes} \times \text{Durée (s)}}{\text{Nb de bits dans 1 octet}} = 51\,840\,000 \text{ octets}$$

$$\frac{f \times 24 \times 2 \times 3 \times 60}{8} = 51\,840\,000$$

$$\text{D'où } f = \frac{51\,840\,000 \times 8}{24 \times 2 \times 3 \times 60}$$

$$f = 48\,000 \text{ hz}$$

*Le fichier a été réencodé à une fréquence d'échantillonnage de 48 000 hz sur 24 bits en stéréo, ce qui correspond à un réglage qualité DVD.*

4. La qualité de la chanson a-t-elle été améliorée après ce réencodage ? Justifier. (……. / 0,5 pts)

*Ce réencodage n'apporte strictement aucune amélioration de la qualité. La qualité d'un fichier réencodé est inférieure ou égale à la qualité initiale. En effet, l'encodage fait perdre de l'information à cause de l'échantillonnage et de la quantification. Et le réencodage ne recrée pas les informations perdues. En particulier une piste mono ne peut pas devenir stéréo par magie après réencodage !*

*Il ne s'agissait pas ici de comparer les qualités des 2 encodages mais la qualité des 2 chansons après encodage puis réencodage.*

*Il existe cependant des techniques algorithmiques qui permettent plus ou moins d'extrapoler ou d'interpoler les données manquantes. Par exemple de simuler de la stéréo à partir d'une piste mono.*

5. Le fichier qu'Anna avait téléchargé au tout début était en fait un fichier compressé au format mp3.
   Ce fichier mp3 provenait du réencodage de la même chanson sur un CD (en Stéréo 16 bits à 44 100 Hz).
   Quel est le taux de compression (en % arrondi à l'unité) associé à ce format mp3 ? (…….. / 1,5 pts)

*• Calculons le poids de cette chanson sur le CD initial.*

$$\text{Poids de la chanson sur le CD (en octets)} = \frac{\text{Fréquence} \times \text{Quantification} \times \text{Nb de pistes} \times \text{Durée (s)}}{\text{Nb de bits dans 1 octet}}$$

$$= \frac{44\,100 \times 16 \times 2 \times 3 \times 60}{8}$$

$$= 31\,752\,000 \text{ octets}$$

$$= 31{,}752 \text{ Mo}$$

*• Le taux de compression se calcule comme une banale proportion :*

$$\text{Taux de compression} = \frac{\text{Poids du fichier mp3}}{\text{Poids de la chanson initale sur CD}} \times 100$$

$$= \frac{2{,}88}{31{,}752} \times 100$$

$$\approx 9\ \%$$

*Le taux de compression associé à ce format mp3 est de 9 % ce qui est extrêmement destructeur.*
*D'où la mauvaise qualité du son.*

➢ Exercice n° 3 (……..…… / 4 points) : QCM Tableaux, Chaînes de caractères en Python.

Pour chaque affirmation, 3 choix vous sont proposés dont un seul est vrai. Lequel ? **L'entourer.**

Barème : réponse juste =1 pt  sans réponse ou réponse fausse = 0 pt

| *Affirmations* | *Choix 1* | *Choix 2* | *Choix 3* |
|---|---|---|---|
| *① list(range (1)) renvoie* | [ ] | [ 0 ] | [ 0 , 1 ] |
| *② Soit tab = [-1, -2, -3, -4]*<br>*tab[-1] renvoie* | Error | -3 | -4 |
| *③ Pour obtenir le numéro unicode d'un caractère, on peut utiliser la fonction* | chr( ) | ord( ) | str( ) |
| *④ Soit Phrase une chaîne de caractères.*<br>*Phrase = "bonjour"*<br>*Phrase [6] renvoie* | "u" | "r" | Error |

*① • range (a, b, c) crée une liste **débutant à l'entier « a » inclus** et **finissant avant l'entier « b » exclu**, avec un pas d'avancée de « c ». Exemples : range (2, 10, 3) crée la liste [ 2, 5, 8 ].*
*range (10, 3, -2) crée la liste [ 10, 8, 6, 4 ].*
*• range (n) équivaut à range (0, n, 1) soit une liste qui commence à 0 ! et qui se termine à n – 1 !*
*② tab[-1] renvoie le dernier élément d'un tableau, tab[-2] renvoie l'avant dernier etc.*
*③ La fonction chr() renvoie, s'il existe, le caractère unicode correspondant à un entier.*
*La fonction ord() renvoie le point de code unicode d'un caractère (le numéro du caractère).*
*La fonction str() convertit en caractères tout ce qui est en argument.*
*④ N'importe quelle chaîne de caractères peut être considérée comme une liste ! Donc Phrase[6] renvoie le 7ème caractère écrit.*

➢ Exercice n° 4 (……..…. / 4,5 pts) : Codage binaire en Python.

1. Donner l'écriture binaire de 42 : *10 1010 sur 6 bits.* (…….. / 0,5 pts)
2. Analyse algorithmique : (…….. / 2 pts)

   Ecrire dans ses grandes lignes l'algorithme **par questions successives** (passage base 10 à base 2).

   Laisser schémas et exemples qui ont permis d'aboutir à cet algorithme.

   Commenter si besoin.
3. Programmation Python : (…… / 2 pts)

   Traduire l'algorithme précédent en une fonction Binary qui en argument reçoit un nombre entier positif et qui retourne en sortie la chaîne de caractères donnant l'écriture binaire de ce nombre.

*2. Analyse algorithmique :*

*Commençons par donner les « Grandes Lignes » de notre algorithme solution (voir encadré p.13 cours « Codage Binaire »).*

*Soit un nombre entier.*

*a. Trouver la plus haute puissance de 2 contenue dans ce nombre. Beaucoup d'oublis de cette recherche.*

*b. Est-ce que le reste (le nombre – la puissance de 2) contient la puissance de 2 inférieure ?*

*Si oui écrire 1 et Calculer le reste, sinon écrire 0.*

*Et ainsi de suite.*

*c. Retourner l'écriture binaire obtenue.*

*Evidemment, ces grandes lignes ne constituent pas un programme (variables non déclarées, conditions d'alternatives ou d'exécution de boucles non précisées etc.) mais débroussaillent grandement le chemin. On retrouve la structure classique :*

*a. Initialisation du programme.*

*b. Moteur principal du programme avec une alternative (la question + si) dans une boucle (« Et ainsi de suite »).*

*c. Sortie du programme.*

*Ce programme sera écrit sous forme d'une fonction, qui sera utilisée dans un petit programme principal.*

3. *Synthèse Python :*

```
#coding : utf8

def Binary (Nombre):
    """ Fonction Binary qui donne l'écriture binaire d'un entier.
        avec entrée = entier Nombre ; sortie = chaîne caractères Nbbinaire."""

    if Nombre == 0: #Traitement à part de 0 sinon exposant plus bas vaudra -1.
        Nbbinaire='0'

    else :          #Traitement général.
        n = 0
        exposant = 0
        while 2**n <= Nombre : #Recherche du plus grand exposant.
            n = n + 1
        exposant = n - 1 #Revenir à la puissance inférieure

        Nbbinaire = ''
        Reste = Nombre
        while exposant >= 0 :   #Fabrication du code binaire. Ou bien for k in range (exposant + 1)
            if 2**exposant <= Reste :
                Nbbinaire = Nbbinaire + '1'
                Reste = Reste - 2**exposant

            else :
                Nbbinaire = Nbbinaire + '0'

            exposant = exposant - 1

    return (Nbbinaire)

#Programme principal.
Nombre = int(input("Entrez un nombre entier : "))
résultat = Binary(Nombre)
print ('l\'écriture binaire de ',Nombre,' est :',résultat,end='.')
```

*Commentaires :*

1. *Ce programme est l'exact reflet de l'encadré p.1 » du cours.*

*On peut le simplifier en manipulant directement les puissances de 2 au lieu des exposants : c'est d'ailleurs la solution écrite par la majorité.*

2. *Quelques solutions originales en 2018 :*

- *Trouver la plus grande puissance de 2 contenue en utilisant la formule :*

$$2^n \leq nb \Rightarrow n = floor(\log_2(nb))$$ *(voir corrigé exo 3 complément mathématique, Contrôle1-2017)*

*Puis utiliser une boucle for sur n pour la fabrication de la chaîne binaire.*

- *Ecrire une fonction pgp (plus grande puissance) et la réutiliser en boucle pour écrire petit à petit le nombre binaire vu cette fois-ci comme un nombre décimal et non comme une chaîne de caractères :*

```
nbdeci = int(input())
nbbinaire = 0
n =0
while nbdeci != 0 :
        n = pgp (nbdeci)
        nbbinaire = nbbinaire + 10**n
        nbdeci = nbdeci – 2**n
print (nbbinaire)
```

3. ***Remarques sur Fonction, Lisibilité, Comment commenter un programme : voir compte rendu au début de ce corrigé.***
4. *On pourrait s'interroger sur la possibilité d'un parallèle entre les algorithmes d'écriture binaire (par soustractions ou par divisions successives) et les algorithmes de recherche du pgcd (par soustractions ou par divisions successives).*