Concours National Commun (CNC), Maroc

Session 2026, filière TSI

Informatique

Ce que traite le sujet

- Exercice : types de moyennes
- Partie I : base de données et SQL
- Partie II : va-t-il pleuvoir demain
- Partie III : stéganographie

Sujet reproduit sans retouche à partir de la page suivante.

Résumé des parties, chapitres de cours liés et autres années :
prepahelp.com/annales/cnc/tsi/2026/info

Prepa Help ne revendique aucun droit sur le sujet, propriété de l'organisateur du concours.

Les candidats sont informés que la précision des raisonnements algorithmiques ainsi que le soin apporté à la rédaction et à la présentation des copies seront des éléments pris en compte dans la notation. Il convient en particulier de rappeler avec précision les références des questions abordées. Si, au cours de l'épreuve, un candidat repère ce qui peut lui sembler être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Remarques générales :

- ✓ Cette épreuve est composée d'un exercice et de trois parties tous indépendants ;
- ✓ Toutes les instructions et les fonctions demandées seront écrites en Python ;
- ✓ Les questions non traitées peuvent être admises pour aborder les questions ultérieures ;
- ✓ Toute fonction peut être décomposée, si nécessaire, en plusieurs fonctions.

Important **: Le candidat doit impérativement commencer par traiter toutes les questions de l'exercice ci-dessous, et écrire les réponses dans les premières pages du cahier de réponses.**

Exercice : (4 points)

Types de moyennes

La liste X contient des nombres strictement positifs.

Q.1- Écrire la fonction ***somme (X)*** qui retourne la somme des éléments de X. (Ne pas utiliser la fonction prédéfinie sum()).

Q.2- Déterminer la complexité de la fonction ***somme (X),*** avec justification.

Q.3- Écrire la fonction ***moy_arithmetique (X)*** qui retourne la moyenne arithmétique de X, en utilisant la formule suivante :

$$\frac{1}{n} * \sum x_i \qquad (avec : \ n = taille\ de\ X)$$

Q.4- Écrire la fonction ***produit (X)*** qui retourne le produit des éléments de X.

Q.5- Écrire la fonction ***moy_geometrique (X)*** qui retourne la moyenne géométrique de X, en utilisant la formule suivante :

$$\sqrt[n]{\prod x_i} \qquad (avec : \ n = taille\ de\ X)$$

Q.6- Écrire la fonction ***carres (X)*** qui retourne la liste des carrés des éléments de X.

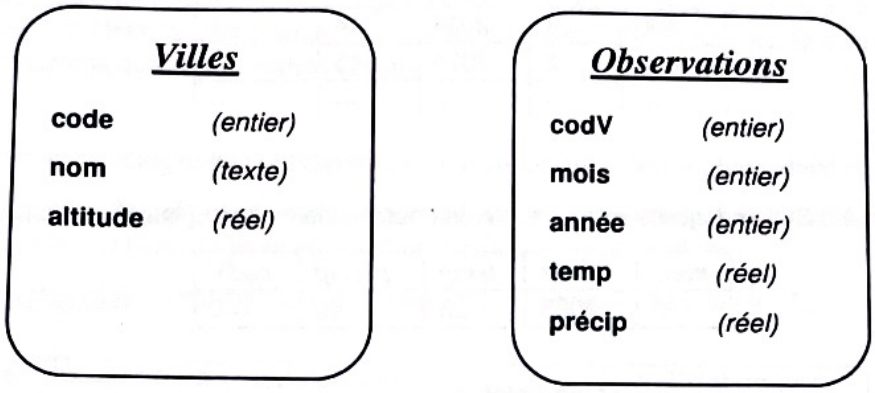
Exemple : ***carres ([9, 12, 20, 5])*** retourne la liste ***[81, 144, 400, 25]***

Q.7- Écrire la fonction ***moy_quadratique (X)*** qui retourne la moyenne quadratique de X, en utilisant la formule suivante :

$$\sqrt{\frac{1}{n} * \sum x_i^2} \qquad (avec : \ n = taille\ de\ X)$$

Partie I : *Base de données et langage SQL*

Les données provenant des observations météorologiques mensuelles de plusieurs villes, sont enregistrées dans une base de données composée de deux tables appelées : **'Villes' et 'Observations'**.



Structure de la table 'Villes' :

La table **'Villes'** est composée des champs suivants :

- Le champ ***code*** contient un code unique pour chaque ville ;
- Le champ **nom** contient le nom de chaque ville ;
- Le champ **altitude** contient l'altitude (en **mètre**) de chaque ville ;

Exemples :

code	*nom*	*altitude*
900	Tanger	56
400	Marrakech	465
200	Casa Blanca	115
100	Rabat	87
...	...	...

Structure de la table 'Observations' :

La table **'Observations'** est composée des champs suivants :

- Le champ ***codV*** contient le code de la ville ;
- Les champs ***mois***, ***année*** contiennent le mois et l'année de l'observation météorologique dans chaque ville ;
- Le champ ***temp*** contient la température moyenne (en ***°C***) dans chaque ville, durant un mois ;
- Le champ ***précip*** contient le cumul (en ***mm***) des précipitations de pluie dans chaque ville, durant un mois.

Exemples :

codV	*mois*	*année*	*temp*	*précip*
900	1	2024	17	10
900	2	2024	15	14
900	3	2024	16	79
900	1	2025	18	125
400	1	2024	14	4
400	2	2024	15	12
...	...	...	...	...

Q.8- Écrire une requête SQL qui ajoute dans la table des observations, l'enregistrement suivant :

mois	*année*	*temp*	*précip*	*codV*
3	2025	25	50	400

Q.9- Écrire, en algèbre relationnelle, une requête qui donne pour résultat *les mois, les températures et les précipitations de la ville* ***'Agadir'****, durant l'année* ***2024***.

Q.10- Écrire une requête SQL qui donne pour résultat *les noms et les altitudes des villes, les températures et les précipitations du mois* ***10*** *de l'année* ***2025***. *Le résultat doit être trié dans l'ordre décroissant des précipitations.*

Q.11- Écrire une requête SQL qui donne pour résultat *le compte des villes qui ont enregistré des températures comprises entre* ***45 °C*** *et* ***50 °C***, *durant le mois* ***7*** *ou le mois* ***8*** *de l'année* ***2025***.

Q.12- Écrire une requête SQL qui donne pour résultat *les noms des villes, la plus grande température et la plus petite température par ville durant l'année* ***2025***. *Le résultat doit être trié dans l'ordre alphabétique des noms des villes.*

Q.13- Écrire une requête SQL qui donne pour résultat *les années, les noms des villes, le cumul annuel des précipitations, et la moyenne annuelle des températures par ville par année, tels que le cumul annuel des précipitations est inférieur à* ***450 mm***.

Q.14- Écrire une requête SQL qui donne pour résultat *la moyenne des cumuls annuels des précipitations de chaque ville, durant l'année* ***2025***.

Partie II :

Va-t-il pleuvoir demain ?

Dans cette partie, on utilise un ensemble de données provenant des observations météorologiques quotidiennes de plusieurs stations météorologiques. L'objectif est de prédire s'il pleuvra demain ou non, en utilisant le modèle de classification binaire K plus proches voisins (***KNN***). Ensuite, la matrice de confusion est utilisée pour mesurer la qualité du modèle construit.

Les observations météorologiques quotidiennes sont composées de **6** variables, dans l'ordre suivant :

- *Cinq variables de prédiction* : température minimale, température maximale, température moyenne, précipitations, et l'état de pluie aujourd'hui (**1** : pluie, **0** : pas de pluie)
- *Une variable cible* : l'état de pluie le jour suivant (**1** : pluie, **0** : pas de pluie).

température minimale	température maximale	température moyenne	précipitations	l'état de pluie aujourd'hui	état de pluie le jour suivant
22.8	*28.9*	*24.25*	*1.5*	*0*	*0*
22.8	*29.9*	*26.0*	*0.0*	*0*	*0*
22.4	*30.0*	*24.2*	*1.1*	*0*	*0*
22.4	*29.7*	*25.6*	*11.3*	*1*	*0*
24.8	*30.4*	*28.1*	*0.0*	*0*	*0*
22.0	*31.8*	*25.1*	*30.2*	*1*	*1*
...	...	...	...	...	...
22.2	*32.5*	*25.6*	*0.0*	*0*	*0*

L'algorithme KNN est un algorithme de Machine Learning qui appartient à la classe des algorithmes d'apprentissage supervisé, utilisé pour résoudre les problèmes de classification ou de régression.

A- Phase d'entrainement du modèle de classification KNN

Dans la phase d'entrainement du modèle, on utilise une liste ***E*** qui contient un échantillon de ***30%*** des observations météorologiques quotidiennes.

Exemple :

```
E = [ [22.0, 27.4, 25.0, 64.8, 1, 1],
      [25.0, 31.2, 28.5, 14.2, 1, 1],
      [23.6, 32.2, 26.7,  0.0, 0, 0],
      [22.6, 34.0, 25.8,  0.0, 0, 0],
      [22.4, 34.2, 25.7, 23.1, 1, 1],
      [24.2, 28.3, 26.0, 14.3, 0, 0],
      [24.0, 28.8, 25.8, 12.9, 0, 0],
      ...
      [24.2, 31.4, 27.1, 10.3, 1, 1] ]
```

Pour mesurer la similarité entre deux observations météorologique quotidiennes, on propose d'utiliser la *distance euclidienne*.

La distance euclidienne entre deux observations météorologiques quotidiennes ***X*** et ***Y*** est définie par la formule suivante :

$$dist\,(X,Y) = \sqrt{\sum_{i=0}^{4}(X_i - Y_i)^2}$$

Q.15- Écrire la fonction ***dist (X, Y)*** qui retourne la distance euclidienne entre deux observations météorologiques quotidiennes ***X*** et ***Y***.

Exemple :

X = [22.0, 27.4, 25.0, 64.8, 1, 1]
Y = [25.0, 31.2, 28.5, 14.2, 1, 1]
dist (X, Y) retourne le nombre ***50.95144747698537***

Q.16- Écrire la fonction ***tri (E, R)*** qui reçoit en paramètres la liste ***E***, et une liste ***R*** qui représente une observation météorologique un jour dans une station météorologique. La fonction trie les éléments de ***E*** dans l'ordre croissant de la distance euclidienne entre ***R*** et chaque élément de ***E***. (Ne pas utiliser la fonction *sorted()* et la méthode *sort()*)

Exemples :

R = [23.8, 30.0, 26.1, 32.7, 1, 1]

Après l'appel de la fonction ***tri (E, R)***, on obtient la liste suivante :

```
E = [ [22.6, 32.2, 25.8, 30.6, 1, 1],
      [23.5, 32.2, 26.9, 30.0, 1, 1],
      [22.0, 31.8, 25.1, 30.2, 1, 1],
      [22.0, 31.8, 25.1, 30.2, 1, 1],
      ...
      [24.0, 34.8, 28.2,  0.0, 0, 0],
      [23.8, 35.2, 28.3,  0.0, 0, 0],
      [22.0, 34.9, 28.4,  0.0, 0, 0],
      ...
      [25.0, 30.6, 27.4, 89.0, 1, 1] ]
```

Q.17- Écrire la fonction de ***plus_proches (E, R, k)*** qui reçoit en paramètres la liste ***E***, une liste ***R*** qui représente une observation météorologique un jour dans une station météorologique, et ***k*** un entier strictement positif. La fonction retourne une liste contenant l'état de pluie le jour suivant des ***k*** observations météorologiques de ***E*** les plus proches à ***R***.

Exemple :

R = [23.8, 30.0, 26.1, 32.7, 1, 1]

- ***plus_proches (E, R, 9)*** retourne la liste ***[1, 1, 1, 1, 1, 1, 1, 1, 0]***
- ***plus_proches (E, R, 12)*** retourne la liste ***[1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1]***

Q.18- Écrire la fonction de ***knn (E, R, k)*** qui reçoit en paramètres la liste ***E***, une liste ***R*** qui représente une observation météorologique un jour dans une station météorologique, et ***k*** un entier strictement positif. La fonction retourne *l'état de pluie le jour suivant*, le plus répété dans les *états de pluie le jour suivant* des ***k*** observations météorologiques de ***E*** les plus proches à ***R***.

Exemple :

R = [23.8, 30.0, 26.1, 32.7, 1, 1]

- ***knn (E, R, 9)*** retourne ***1***
- ***knn (E, R, 11)*** retourne ***1***

B- Phase du test

Dans la phase du test du modèle, on utilise une liste ***T*** qui contient les ***70%*** restant des observations météorologiques quotidiennes.

```
T= [ [22.0, 31.8, 25.1, 30.2, 1, 1],
     [26.2, 31.8, 28.5,  0.0, 0, 0],
     [23.8, 31.3, 28.0,  8.5, 0, 0],
     [23.6, 32.4, 27.5, 16.5, 1, 1],
     [24.0, 30.2, 26.3,  3.4, 0, 0],
     [25.4, 30.6, 27.6, 67.8, 1, 1],
     [24.2, 31.5, 28.5,  1.2, 0, 0],
     [24.6, 29.0, 26.0,  0.1, 0, 0],
     ... ]
```

Q.19- Écrire la fonction ***test (E, T, k)*** qui reçoit en paramètres les listes ***E*** et ***T***, et un entier ***k*** strictement positif. En utilisant l'algorithme ***KNN***, la fonction retourne une liste ***R*** telle que :

- Chaque élément ***Ri*** est un tuple à deux éléments ;
- Le premier élément du tuple est l'état de pluie réel le jour suivant, dans l'observation météorologique ***Ti*** ;
- Le deuxième élément du tuple est le l'état de pluie le jour suivant, prédit par l'algorithme KNN, pour l'observation météorologique ***Ti*** ;

Exemple :

La fonction ***test (E, T, 13)*** retourne la liste suivante :

[(1, 1), (1, 1), (1, 1), (1, 1), (0, 0), (1, 1), (1, 1), (1, 1), (1, 1), (0, 0), (0, 0), (1, 1), (0, 0), (0, 1), (1, 1), (0, 0), (1, 1), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (1, 1), (0, 0), (0, 0), (0, 0), (1, 1), (1, 1), (0, 0), (1, 1), (1, 1), (0, 1), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (1, 1), (0, 0), (0, 0), ...]

Matrice de confusion

La matrice de confusion est un outil essentiel en apprentissage automatique pour évaluer la performance des modèles de classification en comparant les valeurs prédites aux valeurs réelles.

La matrice de confusion est une représentation tabulaire qui permet de visualiser les performances d'un modèle de classification. Elle compare les résultats réels (valeurs cibles) aux résultats prédits par le modèle. Chaque ligne de la matrice représente une classe réelle, tandis que chaque colonne représente une classe prédite. Cela permet d'identifier les prédictions correctes et incorrectes, ainsi que le type d'erreurs commises.

Structure de la matrice de confusion :

Dans sa forme la plus simple, la matrice de confusion est une matrice ***2x2*** pour les problèmes de classification binaire, qui contient les quatre catégories suivantes :

Vrais Positifs (VP)	***Faux Négatifs (FN)***
Faux Positifs (FP)	***Vrais Négatifs (VN)***

- ✓ ***VP*** : *Prédictions positives correctes*. Le vrai état de pluie le jour suivant est ***1***, et l'état de pluie le jour suivant prédit par *KNN* est ***1*** ;
- ✓ ***FP*** : *Prédictions positives incorrectes*. Le vrai état de pluie le jour suivant est ***0***, et l'état de pluie le jour suivant prédit par *KNN* est ***1*** ;
- ✓ ***FN*** : *Prédictions négatives incorrectes*. Le vrai état de pluie le jour suivant est ***1***, et l'état de pluie le jour suivant prédit par *KNN* est ***0***.
- ✓ ***VN*** : *Prédictions négatives correctes*. Le vrai état de pluie le jour suivant est ***0***, et l'état de pluie le jour suivant prédit par *KNN* est ***0*** ;

Q.20- Écrire la fonction ***matrice_confusion (E, T, k)*** qui reçoit en paramètres les listes ***E*** et ***T***, et un entier strictement positif ***k***. La fonction retourne la liste qui représente la matrice de confusion.

Exemple :

La fonction ***matrice_confusion (E, T, 13)*** retourne la matrice de confusion : ***[[154, 2], [41, 450]]***

Métriques utilisées pour évaluer le modèle :

À partir des valeurs de la matrice de confusion, plusieurs métriques peuvent être calculées pour évaluer la performance du modèle :

- **Précision** : Mesure la proportion de prédictions positives correctes par rapport à toutes les prédictions positives.

$$Précision = \frac{VP}{VP + FP}$$

- **Rappel** : Mesure la capacité du modèle à identifier toutes les instances positives.

$$Rappel = \frac{VP}{VP + FN}$$

- **Exactitude** : Mesure la proportion de prédictions correctes (positives et négatives) par rapport à l'ensemble des prédictions.

$$Exactitude = \frac{VP + VN}{VP + FP + VN + FN}$$

- **Score F1** : La moyenne harmonique de la précision et du rappel, utile lorsque les classes sont déséquilibrées.

$$Score\ F1 = 2 * \frac{Précision * Rappel}{Précision + Rappel}$$

Q.21- Écrire la fonction ***metriques (M)*** qui reçoit en paramètre la matrice de confusion ***M***, et qui retourne une liste contenant les valeurs, en pourcentage, de la ***Précision***, du ***Rappel***, de l'***Exactitude*** et du ***Score F1***.

Exemple :

metriques (M) retourne la liste ***[88.33, 98.15, 96.29, 92.98]***

Partie III :

La stéganographie

L'art de dissimuler des données

Dans cette partie, on suppose que le module **numpy** est importé :

```
import numpy as np
```

La ***stéganographie***, c'est l'art de dissimuler des données dans d'autres données. Il existe plusieurs techniques différentes qui permettent ce "tour de magie".

La stéganographie existe depuis longtemps, bien avant l'invention de l'ordinateur. Avec l'avènement de l'ordinateur et de son règne du "tout numérique", des techniques existent permettant pour cacher n'importe quel document dans un autre document. Les documents "porteurs" sont généralement des images (BMP, JPEG, ...) ou des sons (WAV, ...)

La stéganographie est utilisée dans l'industrie actuelle. Sa principale application est le watermarking. Cela consiste à cacher un copyright au sein d'une œuvre protégée. Ainsi, en cas de litige de droits d'auteurs, le watermark sera montré pour prouver l'originalité de l'œuvre.

Dans la suite de cette partie, nous allons voir comment dissimuler un texte, dans une image couleur. On suppose que le texte à dissimuler est composé de caractères dont les codes sont inférieurs strictement *256*, et qu'il ne contient pas le caractère **#**.

La fonction ***ord()*** est une fonction prédéfinie en langage Python, qui reçoit en paramètre un caractère, et elle renvoie son code.

Exemples :

- ***ord ('a')*** retourne le nombre ***97***
- ***ord ('!')*** retourne le nombre ***33***

La fonction ***chr()*** est la fonction réciproque à la fonction ord(). Elle reçoit en paramètre un code, et elle renvoie le caractère correspondant à ce code.

Exemples :

- ***chr (97)*** retourne le caractère ***'a'***
- ***chr (33)*** retourne le caractère ***'!'***

Q.22- Écrire la fonction ***codes (T)*** qui reçoit en paramètre une chaîne de caractères ***T***, et qui retourne la liste contenant successivement les codes des caractères de ***T***.

Exemple :
codes ("Brad Gilberest PHOTOGRAPHY") retourne la liste : ***[66, 114, 97, 100, 32, 71, 105, 108, 98, 101, 114, 101, 115, 116, 32, 80, 72, 79, 84, 79, 71, 82, 65, 80, 72, 89]***

Q.23- Écrire la fonction ***binaire (x)*** qui reçoit en paramètre un entier ***x*** qui représente le code d'un caractère. La fonction retourne une liste de longueur ***8***, qui représente la conversion binaire de ***x***. Le bit du poids le plus faible se trouve à la fin de la liste.

Exemples :

- ***binaire (66)*** retourne la liste ***[0, 1, 0, 0, 0, 0, 1, 0]***
- ***binaire (114)*** retourne la liste ***[0, 1, 1, 1, 0, 0, 1, 0]***

Q.24- Écrire la fonction ***liste_binaire (T)*** qui reçoit en paramètre une chaîne de caractères ***T***, et qui retourne une liste qui contient successivement les conversions binaires de tous les caractères de ***T***.

Exemple :

liste_binaire ("Brad Gilberest PHOTOGRAPHY ") retourne la liste :

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

Q.25- Écrire la fonction ***decimal (L)*** qui reçoit en paramètre une liste ***L*** contenant la conversion binaire du code d'un caractère. La fonction retourne la valeur décimale du code de cet entier, sachant que le bit du poids le plus faible se trouve à la fin de la liste.

Exemples :

- ***decimal ([0, 1, 0, 0, 0, 0, 1, 0])*** retourne le nombre ***66***
- ***decimal ([0, 1, 1, 1, 0, 0, 1, 0])*** retourne le nombre ***114***

Pour **dissimuler** un bit ***b*** (***0*** ou ***1***) dans un nombre entier ***x***, on remplace le dernier bit (bit de poids le plus faible) dans la conversion binaire de ***x***, par le bit ***b***. Et de calculer la valeur décimale de ***x***.

Exemple :

La conversion binaire de ***x=66*** est : ***[0, 1, 0, 0, 0, 0, 1, 0]***

Pour dissimuler le bit ***1*** dans ***x***, on remplace le dernier bit de ***x*** par ***1*** : ***[0, 1, 0, 0, 0, 0, 1, 1]***

La nouvelle valeur décimale de ***x*** est : ***67***

Q.26- Écrire la fonction ***dissimule_bit (x, b)*** qui reçoit en paramètres entier ***x*** tel que ***0≤x≤255***, et un bit ***b*** (***0*** ou ***1***). La fonction dissimule le bit ***b*** dans le nombre ***x***, et elle retourne la nouvelle valeur décimale de ***x***.

Exemples :

- ***dissimule_bit (66, 0)*** retourne le nombre ***66***
- ***dissimule_bit (66, 1)*** retourne le nombre ***67***
- ***dissimule_bit (97, 0)*** retourne le nombre ***96***
- ***dissimule_bit (97, 1)*** retourne le nombre ***97***

Dans une image couleur, chaque pixel est constitué de **3** octets : le premier octet pour la composante ***rouge***, le deuxième octet pour la composante ***verte*** et le troisième octet pour la composante ***bleue***. À partir de ces trois octets, on peut donc avoir 256^3 couleurs différentes, ce qui est largement plus que ne peut distinguer l'œil humain.

Exemple :

Matrice qui représente une image en couleurs :



```
[ [ [73 121 45] [ 21  73   0] [25  80   0] [84 145 42] ... ]
  [ [21  73  0] [ 34  87   0] [66 123 26] [97 155 53] ... ]
  [ [30  88  0] [ 77 135 33] [97 158 54] [67 128 24] ... ]
  [ [65 129 15] [100 164 52] [91 155 45] [34  97  0] ... ]
  [ [71 140 21] [ 79 146 31] [65 130 26] [39  98  6] ... ]

...

]
```

Principe pour dissimuler un texte dans une image en couleurs :

L'astuce est d'utiliser le dernier bit (bit de poids le plus faible) dans chaque octet qui compose chaque pixel de l'image en couleurs. En modifiant ce bit, on modifie un peu l'image, mais ce n'est pas visible à l'œil nu :

- On ajoute le caractère '#' à la fin du texte à dissimuler, pour marquer la fin du texte ;
- On converti le texte à dissimuler en une liste de bits ***B*** ;
- On dissimule successivement chaque bit de la liste ***B*** dans chaque octet qui compose chaque pixel de l'image en couleurs.

Exemple :

```
B = [ 0,   1,   0,   0,   0,   0, ... ]
      ↓    ↓    ↓    ↓    ↓    ↓
    [ 73  121  45]  [21  73   0 ] ...
          Pixel          Pixel
```

- On dissimule le bit **0** dans le nombre **73**
- On dissimule le bit **1** dans le nombre **121**
- On dissimule le bit **0** dans le nombre **45**
- ...

Q.27- Selon le principe ci-dessus, quelle est la taille maximale en ***MO*** du texte, qu'on peut dissimuler dans une image en couleurs de taille ***1GO*** ?

Q.28- Écrire la fonction ***lire_fichier (ch)*** qui reçoit en paramètre une chaîne de caractères ***ch*** qui contient le chemin d'un fichier texte. La fonction retourne le contenu du fichier texte dans une chaîne de caractères.

Q.29- Écrire la fonction ***dissimule_txt (M, ch)*** qui reçoit en paramètres une matrice ***M*** qui représente une image en couleurs, et une chaîne de caractères ***ch*** qui contient le chemin d'un fichier texte. La fonction dissimule le contenu du fichier texte dans l'image ***M***, en utilisant le principe précèdent. (*On suppose que la taille de l'image **M** est suffisante pour la dissimulation du contenu du fichier texte*)

Exemple :
On suppose que le contenu du fichier texte est : "*Brad Gilberest PHOTOGRAPHY*"



Image couleur après la dissimulation du contenu du fichier texte

Les modifications de l'image ne sont pas visibles à l'œil nu

Q.30- Écrire la fonction ***recupere_bits (M)*** qui reçoit en paramètre une matrice ***M*** qui représente une image en couleurs, dans laquelle on a dissimulé un texte, en utilisant le principe précèdent. La fonction retourne une liste contenant les bits dissimulés dans les pixels de l'image en couleurs ***M***.

Exemple :
recupere_bits (M) retourne la liste :
[0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, ...]

Q.31- Écrire la fonction ***extraire_txt (M)*** qui reçoit en paramètre une matrice ***M*** qui représente une image en couleurs, dans laquelle on a dissimulé un texte, en utilisant le principe précèdent. La fonction retourne le texte dissimulé dans l'image en couleurs ***M***.

Exemple :
extraire_txt (M) retourne le texte : "*Brad Gilberest PHOTOGRAPHY*"

********** **FIN** **********

Travailler les chapitres de ce sujet

Prepa Help suit le programme de TSI chapitre par chapitre : cours détaillé, méthodes, pièges, exercices et problèmes corrigés pas à pas, tirés des concours. L'essai gratuit donne accès aux chapitres offerts de ta filière, sans carte bancaire.

Chapitres liés à ce sujet

- Architecture de l'ordinateur et représentation des nombres
- Bases de Python : expressions, types, instructions, fonctions
- Complexité des algorithmes itératifs et récursifs
- Tri par fusion et tri rapide

Ouvrir l'essai gratuit

Scanne le code ou clique sur le bouton.

prepahelp.com/#essai

Une question sur le concours ou sur ta filière : **écrire sur WhatsApp**

Les autres années et épreuves : prepahelp.com/annales/cnc/tsi