

# **BACCALAUREAT**

**SESSION 2022**

---

**Épreuve de l'enseignement de spécialité**

**NUMERIQUE et SCIENCES  
INFORMATIQUES**

**Partie pratique**

**Classe Terminale de la voie générale**

---

**Sujet n°31**

---

**DUREE DE L'ÉPREUVE : 1 heure**

**Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3  
Dès que le sujet vous est remis, assurez-vous qu'il est complet.**

*Le candidat doit traiter les 2 exercices.*

## EXERCICE 1 (4 points)

Écrire en langage Python une fonction `recherche` prenant comme paramètres une variable `a` de type numérique (`float` ou `int`) et un tableau `t` (type `list`) et qui renvoie le nombre d'occurrences de `a` dans `t`.

Exemples d'utilisations de la fonction `recherche` :

```
>>> recherche(5, [])
0
>>> recherche(5, [-2, 3, 4, 8])
0
>>> recherche(5, [-2, 3, 1, 5, 3, 7, 4])
1
>>> recherche(5, [-2, 5, 3, 5, 4, 5])
3
```

## EXERCICE 2 (4 points)

La fonction `rendu_monnaie_centimes` prend en paramètres deux nombres entiers positifs `s_due` et `s_versee` et elle permet de procéder au rendu de monnaie de la différence `s_versee - s_due` pour des achats effectués avec le système de pièces de la zone Euro. On utilise pour cela un algorithme qui commence par rendre le maximum de pièces de plus grandes valeurs et ainsi de suite. La fonction renvoie la liste des pièces qui composent le rendu.

Toutes les sommes sont exprimées en centimes d'euros. Les valeurs possibles pour les pièces sont donc `[1, 2, 5, 10, 20, 50, 100, 200]`.

Ainsi, l'instruction `rendu_monnaie_centimes(452, 500)` renverra la liste `[20, 20, 5, 2, 1]`.

En effet, la somme à rendre est de 48 centimes soit  $20 + 20 + 5 + 2 + 1$ .

Le code de la fonction est donné ci-dessous :

```
def rendu_monnaie_centimes(s_due, s_versee):
    pieces = [1, 2, 5, 10, 20, 50, 100, 200]
    rendu = ...
    a_rendre = ...
    i = len(pieces) - 1
    while a_rendre > ... :
        if pieces[i] <= a_rendre :
            rendu.append(...)
            a_rendre = ...
        else :
            i = ...
    return rendu
```

Compléter ce code pour qu'il donne :

```
>>> rendu_monnaie_centimes(700,700)
[]
>>> rendu_monnaie_centimes(112,500)
[200, 100, 50, 20, 10, 5, 2, 1]
```

# **BACCALAUREAT**

**SESSION 2022**

---

**Épreuve de l'enseignement de spécialité**

**NUMERIQUE et SCIENCES  
INFORMATIQUES**

**Partie pratique**

**Classe Terminale de la voie générale**

---

**Sujet n°32**

---

**DUREE DE L'ÉPREUVE : 1 heure**

**Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3  
Dès que le sujet vous est remis, assurez-vous qu'il est complet.**

*Le candidat doit traiter les 2 exercices.*

## EXERCICE 1 (4 points)

Écrire une fonction `recherche` qui prend en paramètres `elt` un nombre et `tab` un tableau de nombres, et qui renvoie l'indice de la dernière occurrence de `elt` dans `tab` si `elt` est dans `tab` et le `-1` sinon.

Exemples :

```
>>> recherche(1, [2, 3, 4])
-1
>>> recherche(1, [10, 12, 1, 56])
2
>>> recherche(1, [1, 50, 1])
2
>>> recherche(1, [8, 1, 10, 1, 7, 1, 8])
5
```

## EXERCICE 2 (4 points)

On définit une classe gérant une adresse IPv4.

On rappelle qu'une adresse IPv4 est une adresse de longueur 4 octets, notée en décimale à point, en séparant chacun des octets par un point. On considère un réseau privé avec une plage d'adresses IP de 192.168.0.0 à 192.168.0.255.

On considère que les adresses IP saisies sont valides.

Les adresses IP 192.168.0.0 et 192.168.0.255 sont des adresses réservées.

Le code ci-dessous implémente la classe `AdresseIP`.

```
class AdresseIP:

    def __init__(self, adresse):
        self.adresse = ...

    def liste_octet(self):
        """renvoie une liste de nombres entiers,
        la liste des octets de l'adresse IP"""
        return [int(i) for i in self.adresse.split(".")]

    def est_reservee(self):
        """renvoie True si l'adresse IP est une adresse
        réservée, False sinon"""
        return ... or ...

    def adresse_suivante(self):
        """renvoie un objet de AdresseIP avec l'adresse
        IP qui suit l'adresse self
        si elle existe et False sinon"""
        if ... < 254:
            octet_nouveau = ... + ...
            return AdresseIP('192.168.0.' + ...)
        else:
            return False
```

Compléter le code ci-dessus et instancier trois objets : `adresse1`, `adresse2`, `adresse3` avec respectivement les arguments suivants :

'192.168.0.1', '192.168.0.2', '192.168.0.0'

Vérifier que :

```
>>> adresse1.est_reservee()
False
>>> adresse3.est_reservee()
True
>>> adresse2.adresse_suivante().adresse
'192.168.0.3'
```

# **BACCALAUREAT**

**SESSION 2022**

---

**Épreuve de l'enseignement de spécialité**

## **NUMERIQUE et SCIENCES INFORMATIQUES**

**Partie pratique**

**Classe Terminale de la voie générale**

---

**Sujet n°33**

---

**DUREE DE L'ÉPREUVE : 1 heure**

**Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3  
Dès que le sujet vous est remis, assurez-vous qu'il est complet.**

*Le candidat doit traiter les 2 exercices.*

### EXERCICE 1 (4 points)

On modélise la représentation binaire d'un entier non signé par un tableau d'entiers dont les éléments sont 0 ou 1. Par exemple, le tableau `[1, 0, 1, 0, 0, 1, 1]` représente l'écriture binaire de l'entier dont l'écriture décimale est

$$2^{**6} + 2^{**4} + 2^{**1} + 2^{**0} = 83.$$

À l'aide d'un parcours séquentiel, écrire la fonction `convertir` répondant aux spécifications suivantes :

```
def convertir(T):  
    """  
    T est un tableau d'entiers, dont les éléments sont 0 ou 1 et  
    représentant un entier écrit en binaire. Renvoie l'écriture  
    décimale de l'entier positif dont la représentation binaire  
    est donnée par le tableau T  
    """
```

Exemple :

```
>>> convertir([1, 0, 1, 0, 0, 1, 1])  
83  
>>> convertir([1, 0, 0, 0, 0, 0, 1, 0])  
130
```

### EXERCICE 2 (4 points)

La fonction `tri_insertion` suivante prend en argument une liste `L` et trie cette liste en utilisant la méthode du tri par insertion. Compléter cette fonction pour qu'elle réponde à la spécification demandée.

```
def tri_insertion(L):  
    n = len(L)  
  
    # cas du tableau vide  
    if ...:  
        return L  
  
    for j in range(1,n):  
        e = L[j]  
        i = j  
  
    # A l'étape j, le sous-tableau L[0,j-1] est trié  
    # et on insère L[j] dans ce sous-tableau en déterminant
```



```

# le plus petit i tel que 0 <= i <= j et L[i-1] > L[j].
while i > 0 and L[i-1] > ...:
    i = ...

# si i != j, on décale le sous tableau L[i,j-1] d'un cran
# vers la droite et on place L[j] en position i
if i != j:
    for k in range(j,i,...):
        L[k] = L[...]
```

L[i] = ...

```

return L
```

### Exemples :

```

>>> tri_insertion([2,5,-1,7,0,28])
[-1, 0, 2, 5, 7, 28]
>>> tri_insertion([10,9,8,7,6,5,4,3,2,1,0])
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

# **BACCALAUREAT**

**SESSION 2022**

---

**Épreuve de l'enseignement de spécialité**

**NUMERIQUE et SCIENCES  
INFORMATIQUES**

**Partie pratique**

**Classe Terminale de la voie générale**

---

**Sujet n°34**

---

**DUREE DE L'ÉPREUVE : 1 heure**

**Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3  
Dès que le sujet vous est remis, assurez-vous qu'il est complet.**

*Le candidat doit traiter les 2 exercices.*

## EXERCICE 1 (4 points)

Écrire une fonction `occurrence_max` prenant en paramètres une chaîne de caractères `chaine` et qui renvoie le caractère le plus fréquent de la chaîne. La chaîne ne contient que des lettres en minuscules sans accent.

On pourra s'aider du tableau

```
alphabet=['a','b','c','d','e','f','g','h','i','j','k','l','m','n',
          'o','p','q','r','s','t','u','v','w','x','y','z']
```

et du tableau `occurrence` de 26 éléments où l'on mettra dans `occurrence[i]` le nombre d'apparitions de `alphabet[i]` dans la chaîne. Puis on calculera l'indice `k` d'un maximum du tableau `occurrence` et on affichera `alphabet[k]`.

Exemple :

```
>>> ch='je suis en terminale et je passe le bac et je souhaite
poursuivre des etudes pour devenir expert en informatique'
>>> occurrence_max(ch)
'e'
```

## EXERCICE 2 (4 points)

On considère une image en 256 niveaux de gris que l'on représente par une grille de nombres, c'est-à-dire une liste composée de sous-listes toutes de longueurs identiques.

La largeur de l'image est donc la longueur d'une sous-liste et la hauteur de l'image est le nombre de sous-listes.

Chaque sous-liste représente une ligne de l'image et chaque élément des sous-listes est un entier compris entre 0 et 255, représentant l'intensité lumineuse du pixel.

Le négatif d'une image est l'image constituée des pixels  $x_n$  tels que  $x_n + x_i = 255$  où  $x_i$  est le pixel correspondant de l'image initiale.

Compléter le programme ci-dessous :

```
def nbLig(image):
    '''renvoie le nombre de lignes de l'image'''
    return ...

def nbCol(image):
    '''renvoie la largeur de l'image'''
    return ...

def negatif(image):
    '''renvoie le négatif de l'image sous la forme
    d'une liste de listes'''
```

```

    L = [[0 for k in range(nbCol(image))] for i in
range(nbLig(image))] # on crée une image de 0 aux mêmes
dimensions que le paramètre image
    for i in range(len(image)):
        for j in range(...):
            L[i][j] = ...
    return L

def binaire(image, seuil):
    '''renvoie une image binarisée de l'image sous la forme
    d'une liste de listes contenant des 0 si la valeur
    du pixel est strictement inférieure au seuil
    et 1 sinon'''
    L = [[0 for k in range(nbCol(image))] for i in
range(nbLig(image))] # on crée une image de 0 aux mêmes
dimensions que le paramètre image
    for i in range(len(image)):
        for j in range(...):
            if image[i][j] < ... :
                L[i][j] = ...
            else:
                L[i][j] = ...
    return L

```

#### Exemple :

```

>>> img=[[20, 34, 254, 145, 6], [23, 124, 287, 225, 69], [197,
174, 207, 25, 87], [255, 0, 24, 197, 189]]
>>> nbLig(img)
4
>>> nbCol(img)
5
>>> negatif(img)
[[235, 221, 1, 110, 249], [232, 131, -32, 30, 186], [58, 81, 48,
230, 168], [0, 255, 231, 58, 66]]
>>> binaire(img,120)
[[1, 1, 0, 0, 1], [1, 1, 0, 0, 1], [0, 0, 0, 1, 1], [0, 1, 1, 0,
0]]

```

# **BACCALAUREAT**

**SESSION 2022**

---

**Épreuve de l'enseignement de spécialité**

## **NUMERIQUE et SCIENCES INFORMATIQUES**

**Partie pratique**

**Classe Terminale de la voie générale**

---

**Sujet n°35**

---

**DUREE DE L'ÉPREUVE : 1 heure**

**Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3  
Dès que le sujet vous est remis, assurez-vous qu'il est complet.**

*Le candidat doit traiter les 2 exercices.*

## EXERCICE 1 (4 points)

Ecrire une fonction qui prend en paramètre un tableau d'entiers non vide et qui renvoie la moyenne de ces entiers. La fonction est spécifiée ci-après et doit passer les assertions fournies.

```
def moyenne (tab):
    """
        moyenne(list) -> float
        Entrée : un tableau non vide d'entiers
        Sortie : nombre de type float
        Correspondant à la moyenne des valeurs présentes dans le
        tableau
    """

assert moyenne([1]) == 1
assert moyenne([1,2,3,4,5,6,7]) == 4
assert moyenne([1,2]) == 1.5
```

## EXERCICE 2 (4 points)

Le but de l'exercice est de compléter une fonction qui détermine si une valeur est présente dans un tableau de valeurs triées dans l'ordre croissant.

L'algorithme traite le cas du tableau vide.

L'algorithme est écrit pour que la recherche dichotomique ne se fasse que dans le cas où la valeur est comprise entre les valeurs extrêmes du tableau.

On distingue les trois cas qui renvoient `False` en renvoyant `False, 1`, `False, 2` et `False, 3`.

Compléter l'algorithme de dichotomie donné ci-après.

```
def dichotomie(tab, x):
    """
        tab : tableau trié dans l'ordre croissant
        x : nombre entier
        La fonction renvoie True si tab contient x et False sinon
    """
    # cas du tableau vide
    if ...:
        return False, 1

    # cas où x n'est pas compris entre les valeurs extrêmes
    if (x < tab[0]) or ...:
        return False, 2
```

```

debut = 0
fin = len(tab) - 1
while debut <= fin:
    m = ...
    if x == tab[m]:
        return ...
    if x > tab[m]:
        debut = m + 1
    else:
        fin = ...
return ...

```

### Exemples :

```

>>> dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],28)
True
>>> dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],27)
(False, 3)
>>> dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],1)
(False, 2)
>>> dichotomie([],28)
(False, 1)

```

# **BACCALAUREAT**

**SESSION 2022**

---

**Épreuve de l'enseignement de spécialité**

## **NUMERIQUE et SCIENCES INFORMATIQUES**

**Partie pratique**

**Classe Terminale de la voie générale**

---

**Sujet n°36**

---

**DUREE DE L'ÉPREUVE : 1 heure**

**Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3  
Dès que le sujet vous est remis, assurez-vous qu'il est complet.**

*Le candidat doit traiter les 2 exercices.*



## EXERCICE 1 (4 points)

Programmer la fonction `recherche`, prenant en paramètre un tableau non vide `tab` (type `list`) d'entiers et un entier `n`, et qui renvoie l'indice de la **dernière** occurrence de l'élément cherché. Si l'élément n'est pas présent, la fonction renvoie la longueur du tableau.

Exemples :

```
>>> recherche([5, 3], 1)
2
>>> recherche([2, 4], 2)
0
>>> recherche([2, 3, 5, 2, 4], 2)
3
```

## EXERCICE 2 (4 points)

On souhaite programmer une fonction donnant la distance la plus courte entre un point de départ et une liste de points. Les points sont tous à coordonnées entières.

Les points sont donnés sous la forme d'un tuple de deux entiers.

La liste des points à traiter est donc un tableau de tuples.

On rappelle que la distance entre deux points du plan de coordonnées  $(x ; y)$  et  $(x' ; y')$  est donnée par la formule :

$$d = \sqrt{(x - x')^2 + (y - y')^2}.$$

On importe pour cela la fonction racine carrée (`sqrt`) du module `math` de Python.

On dispose d'une fonction `distance` et d'une fonction `plus_courte_distance_`:

```
from math import sqrt    # import de la fonction racine carrée

def distance(point1, point2):
    """ Calcule et renvoie la distance entre deux points. """
    return sqrt((...) ** 2 + (...) ** 2)

assert distance((1, 0), (5, 3)) == 5.0, "erreur de calcul"

def plus_courte_distance(tab, depart):
```

```

""" Renvoie le point du tableau tab se trouvant à la plus
courte distance du point depart."""
point = tab[0]
min_dist = ...
for i in range (1, ...):
    if distance(tab[i], depart)...:
        point = ...
        min_dist = ...
return point

```

```

assert plus_courte_distance([(7, 9), (2, 5), (5, 2)], (0, 0)) ==
(2, 5), "erreur"

```

Recopier sous Python (sans les commentaires) ces deux fonctions puis compléter leur code et ajouter une ou des déclarations (`assert`) à la fonction `distance` permettant de vérifier la ou les préconditions.

# **BACCALAUREAT**

**SESSION 2022**

---

**Épreuve de l'enseignement de spécialité**

## **NUMERIQUE et SCIENCES INFORMATIQUES**

**Partie pratique**

**Classe Terminale de la voie générale**

---

**Sujet n°37**

---

**DUREE DE L'ÉPREUVE : 1 heure**

**Le sujet comporte 2 pages numérotées de 1 / 3 à 3 / 3  
Dès que le sujet vous est remis, assurez-vous qu'il est complet.**

*Le candidat doit traiter les 2 exercices.*

### EXERCICE 1 (4 points)

Programmer la fonction `verifie` qui prend en paramètre un tableau de valeurs numériques non vide et qui renvoie `True` si ce tableau est trié dans l'ordre croissant, `False` sinon.

Exemples :

```
>>> verifie([0, 5, 8, 8, 9])
True
>>> verifie([8, 12, 4])
False
>>> verifie([-1, 4])
True
>>> verifie([5])
True
```

### EXERCICE 2 (4 points)

Chaque soir, les auditeurs d'une radio votent en ligne pour leur artiste favori. Ces votes sont stockés dans un tableau.

Exemple :

```
Urne = ['Oreilles sales', 'Oreilles sales', 'Oreilles sales', 'Extra Vomit',
        'Lady Baba', 'Extra Vomit', 'Lady Baba', 'Extra Vomit', 'Lady Baba', 'Extra Vomit']
```

La fonction `depouille` doit permettre de compter le nombre de votes exprimés pour chaque artiste. Elle prend en paramètre un tableau et renvoie le résultat dans un dictionnaire dont les clés sont les noms des artistes et les valeurs le nombre de votes en leur faveur.

La fonction `vainqueur` doit désigner le nom du ou des gagnants. Elle prend en paramètre un dictionnaire dont la structure est celle du dictionnaire renvoyé par la fonction `depouille` et renvoie un tableau. Ce tableau peut donc contenir plusieurs éléments s'il y a des artistes ex-aequo.

Compléter les fonctions `depouille` et `vainqueur` ci-après pour qu'elles renvoient les résultats attendus.

```
def depouille(urne):
    resultat = ...
    for bulletin in urne:
        if ...:
            resultat[bulletin] = resultat[bulletin] + 1
        else:
            ...
    return resultat
```

```
def vainqueur(election):
    vainqueur = ''
    nmax = 0
    for candidat in election:
        if ... > ... :
            nmax = ...
            vainqueur = candidat
    liste_finale = [nom for nom in election if election[nom] == ...]
    return ...
```

Exemples d'utilisation :

```
>>> election = depouille(urne)
>>> election
{'Extra Vomit': 4, 'Oreilles sales': 3, 'Lady Baba': 3}

>>> vainqueur(election)
['Extra Vomit']
```

# **BACCALAUREAT**

**SESSION 2022**

---

**Épreuve de l'enseignement de spécialité**

**NUMERIQUE et SCIENCES  
INFORMATIQUES**

**Partie pratique**

**Classe Terminale de la voie générale**

---

**Sujet n°38**

---

**DUREE DE L'ÉPREUVE : 1 heure**

**Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3  
Dès que le sujet vous est remis, assurez-vous qu'il est complet.**

*Le candidat doit traiter les 2 exercices.*

## EXERCICE 1 (4 points)

Écrire une fonction `tri_selection` qui prend en paramètre une liste `tab` de nombres entiers et qui renvoie le tableau trié par ordre croissant.

On utilisera l'algorithme suivant :

- on recherche le plus petit élément du tableau, et on l'échange avec l'élément d'indice 0 ;
- on recherche le second plus petit élément du tableau, et on l'échange avec l'élément d'indice 1 ;
- on continue de cette façon jusqu'à ce que le tableau soit entièrement trié.

Exemple :

```
>>> tri_selection([1,52,6,-9,12])  
[-9, 1, 6, 12, 52]
```

## EXERCICE 2 (4 points)

Le jeu du « plus ou moins » consiste à deviner un nombre entier choisi entre 1 et 99.

Un élève de NSI décide de le coder en langage Python de la manière suivante :

- le programme génère un nombre entier aléatoire compris entre 1 et 99 ;
- si la proposition de l'utilisateur est plus petite que le nombre cherché, l'utilisateur en est averti. Il peut alors en tester un autre ;
- si la proposition de l'utilisateur est plus grande que le nombre cherché, l'utilisateur en est averti. Il peut alors en tester un autre ;
- si l'utilisateur trouve le bon nombre en 10 essais ou moins, il gagne ;
- si l'utilisateur a fait plus de 10 essais sans trouver le bon nombre, il perd.

La fonction `randint` est utilisée. Si `a` et `b` sont des entiers, `randint(a,b)` renvoie un nombre entier compris entre `a` et `b`.

Compléter le code ci-dessous et le tester :

```
from random import randint

def plus_ou_moins():
    nb_mystere = randint(1,...)
    nb_test = int(input("Proposez un nombre entre 1 et 99 : "))
    compteur = ...

    while nb_mystere != ... and compteur < ... :
        compteur = compteur + ...
        if nb_mystere ... nb_test:
            nb_test = int(input("Trop petit ! Testez encore : "))
        else:
            nb_test = int(input("Trop grand ! Testez encore : "))

    if nb_mystere == nb_test:
        print ("Bravo ! Le nombre était ",...)
        print("Nombre d'essais: ",...)
    else:
        print ("Perdu ! Le nombre était ",...)
```



# **BACCALAUREAT**

**SESSION 2022**

---

**Épreuve de l'enseignement de spécialité**

**NUMERIQUE et SCIENCES  
INFORMATIQUES**

**Partie pratique**

**Classe Terminale de la voie générale**

---

**Sujet n°39**

---

**DUREE DE L'ÉPREUVE : 1 heure**

**Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3  
Dès que le sujet vous est remis, assurez-vous qu'il est complet.**

*Le candidat doit traiter les 2 exercices.*

## EXERCICE 1 (4 points)

Écrire une fonction `moyenne` prenant en paramètres une liste d'entiers et qui renvoie la moyenne des valeurs de cette liste.

Exemple :

```
>>> moyenne([10,20,30,40,60,110])
45.0
```

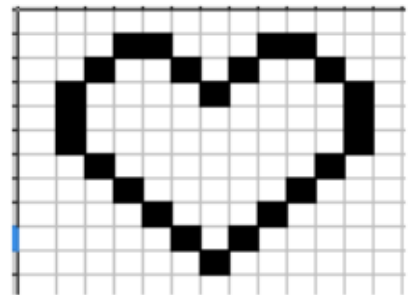
## EXERCICE 2 (4 points)

On travaille sur des dessins en noir et blanc obtenu à partir de pixels noirs et blancs :

La figure « cœur » ci-contre va servir d'exemple.

On la représente par une grille de nombres, c'est-à-dire par une liste composée de sous-listes de mêmes longueurs.

Chaque sous-liste représentera donc une ligne du dessin.



Dans le code ci-dessous, la fonction `affiche` permet d'afficher le dessin. Les pixels noirs (1 dans la grille) seront représentés par le caractère "\*" et les blancs (0 dans la grille) par deux espaces.

La fonction `zoomListe` prend en argument une liste `liste_depart` et un entier `k`. Elle renvoie une liste où chaque élément de `liste_depart` est dupliqué `k` fois.

La fonction `zoomDessin` prend en argument la grille `dessin` et renvoie une grille où toutes les lignes de `dessin` sont zoomées `k` fois et répétées `k` fois.

Soit le code ci-dessous :

```
coeur = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], \
         [0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0], \
         [0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0], \
         [0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1], \
         [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0], \
         [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0], \
         [0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0], \
         [0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0], \
         [0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0], \
         [0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0], \
         [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0], \
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

```
def affiche(dessin):
    ''' affichage d'une grille : les 1 sont représentés par
```

```

        des "*" , les 0 par deux espaces "  " '''
for ligne in dessin:
    for col in ligne:
        if col == 1:
            print(" *",end="")
        else:
            print("  ",end="")
    print()

def zoomListe(liste_depart,k):
    '''renvoie une liste contenant k fois chaque
    élément de liste_depart'''
    liste_zoom = ...
    for elt in ... :
        for i in range(k):
            ...
    return liste_zoom

def zoomDessin(grille,k):
    '''renvoie une grille où les lignes sont zoomées k fois
    ET répétées k fois'''
    grille_zoom=[]
    for elt in grille:
        liste_zoom = ...
        for i in range(k):
            ... .append(...)
    return grille_zoom

```

Résultats à obtenir :

```
>>> affiche(coeur)
```

```

  * *   * *
 * * * *
*   * *
*   *
 *   *
 *   *
 *   *
 *   *
 *   *
 *   *
 *

```

```
>>> affiche(zoomDessin(coeur,3))
```

```

  ++++++ ++++++
 ++++++ ++++++
++++++ ++++++
++++++ ++++++
++++++ ++++++
++++++ ++++++
++++++ ++++++
++++++ ++++++
++++++ ++++++
++++++

```

# **BACCALAUREAT**

**SESSION 2022**

---

**Épreuve de l'enseignement de spécialité**

**NUMERIQUE et SCIENCES  
INFORMATIQUES**

**Partie pratique**

**Classe Terminale de la voie générale**

---

**Sujet n°40**

---

**DUREE DE L'ÉPREUVE : 1 heure**

**Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3  
Dès que le sujet vous est remis, assurez-vous qu'il est complet.**

*Le candidat doit traiter les 2 exercices.*

## EXERCICE 1 (4 points)

Écrire une fonction `recherche` qui prend en paramètres `elt` un nombre et `tab` un tableau de nombres, et qui renvoie le tableau des indices de `elt` dans `tab` si `elt` est dans `tab` et le tableau vide `[]` sinon.

Exemples :

```
>>> recherche(3, [3, 2, 1, 3, 2, 1])
[0, 3]
>>> recherche(4, [1, 2, 3])
[]
```

## EXERCICE 2 (4 points)

Un professeur de NSI décide de gérer les résultats de sa classe sous la forme d'un dictionnaire :

- les clefs sont les noms des élèves ;
- les valeurs sont des dictionnaires dont les clefs sont les types d'épreuves et les valeurs sont les notes obtenues associées à leurs coefficients.

Avec :

```
resultats = {'Dupont':{'DS1' : [15.5, 4],
                      'DM1' : [14.5, 1],
                      'DS2' : [13, 4],
                      'PROJET1' : [16, 3],
                      'DS3' : [14, 4]}},
             'Durand':{'DS1' : [6, 4],
                      'DM1' : [14.5, 1],
                      'DS2' : [8, 4],
                      'PROJET1' : [9, 3],
                      'IE1' : [7, 2],
                      'DS3' : [8, 4],
                      'DS4' : [15, 4]}}
```

L'élève dont le nom est Durand a ainsi obtenu au DS2 la note de 8 avec un coefficient 4.

Le professeur crée une fonction `moyenne` qui prend en paramètre le nom d'un de ces élèves et lui renvoie sa moyenne arrondie au dixième.

Compléter le code du professeur ci-dessous :

```
def moyenne(nom):
    if nom in ...:
        notes = resultats[nom]
        total_points = ...
        total_coefficients = ...
        for ... in notes.values():
            note, coefficient = valeurs
            total_points = total_points + ... * coefficient
            total_coefficients = ... + coefficient
        return round( ... / total_coefficients, 1 )
    else:
        return -1
```