Spécialité NSI Première	**DEVOIR SURVEILLE DE**	Vendredi 3 février 2023
Lycée d'Avesnières	**NSI**	Durée : 2 h
Année scolaire 2022-2023	**N° 5**	Calculatrice interdite

L'énoncé complet est à rendre avec la copie.

NOM : ..... Prénom : .....

Exercice 1 (4 points)

Pour chaque question, il y a une seule bonne réponse. Indiquer **sur la copie seulement le numéro de la question et la réponse choisie**. 0,5 point pour une bonne réponse, 0 point pour une mauvaise.

1) Quelle est la valeur de la variable s après l'exécution du code suivant ?

```
x = 3
y = 7//2
s = (x==y)
```

Réponses :

☐ 3

☐ True

☐ 3.5

☐ False

---

2) Voici les écritures binaires de quatre nombres entiers positifs. Lequel est pair ?

Réponses :

☐ 10 0001

☐ 10 0010

☐ 11 0001

☐ 11 1111

---

3) Soient P et Q deux expressions booléennes telles que P est vraie et Q est fausse.

Quelle est la valeur de l'expression *(P et Q) ou (non(P) ou Q)* ?

Réponses :

☐ Vrai

☐ Faux

☐ Ni vrai ni faux

☐ Vrai et faux en même temps

4) On s'intéresse à la valeur 14 présente dans la liste suivante :

```
L = [[1, 2, 3, 4, 5], [6, 7, 8, 9, 10], [11, 12, 13, 14, 15], [16, 17, 18, 19, 20]]
```

Quelle expression vaut 14 parmi les suivantes ?

Réponses :

☐ L[2][3]

☐ L[3][4]

☐ L[3][2]

☐ L[4][3]

---

5) Soit la liste :

```
liste = [1, [2, 3], [4, 5], 6, 7]
```

Quelle est la valeur de `len(liste)` ?

Réponses :

☐ 1

☐ 3

☐ 5

☐ 7

---

6) On considère la fonction suivante qui prend une liste de nombres en paramètre :

```
def foo(liste):
    for i in range(len(liste)-1):
        if liste[i] > liste[i+1]:
            return False
    return True
```

Quel appel de cette fonction va renvoyer True ?

Réponses :

☐ foo([5, 4, 3, 2, 1])

☐ foo([1, 0, 1, 0])

☐ foo([1, 3, 2, 4])

☐ foo([1, 2, 2, 3, 4, 5])

---

7) On a saisi le code suivant :

```
variable = [8, 12, -7, 52, -5, 32]
lst = [n for n in variable if n < 12]
```

Que contient la liste `lst` à la fin de l'exécution de ce script ?

Réponses :

☐ [8, -7, -5]

☐ [8]

☐ [True, False, True, False, True, False]

☐ [8, 12, -7, 52, -5, 32]

---

8) Soit la suite s'instructions suivantes :

```
fruit = {}

def addone(index, dic):
    if index in dic:
        dic[index] += 1
    else:
        dic[index] = 1

addone('Apple', fruit)
addone('Banana', fruit)
addone('Orange', fruit)
addone('Orange', fruit)
addone('Orange', fruit)
```

Que contient le dictionnaire `fruit` à la fin de l'exécution de ce script ?

Réponses :

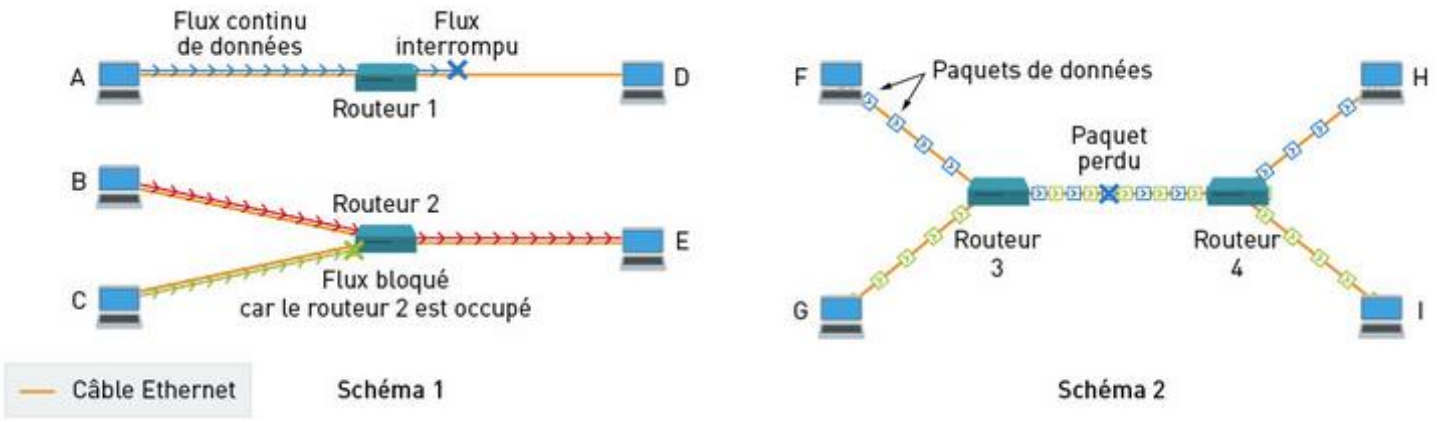
☐ {'Apple', 'Banana', 'Orange', 'Orange', 'Orange'}

☐ {0: 'Apple', 1: 'Banana', 2: 'Orange'}

☐ {'Apple': 1, 'Banana': 1, 'Orange': 3}

☐ {1: 'Apple', 1: 'Banana', 3: 'Orange'}

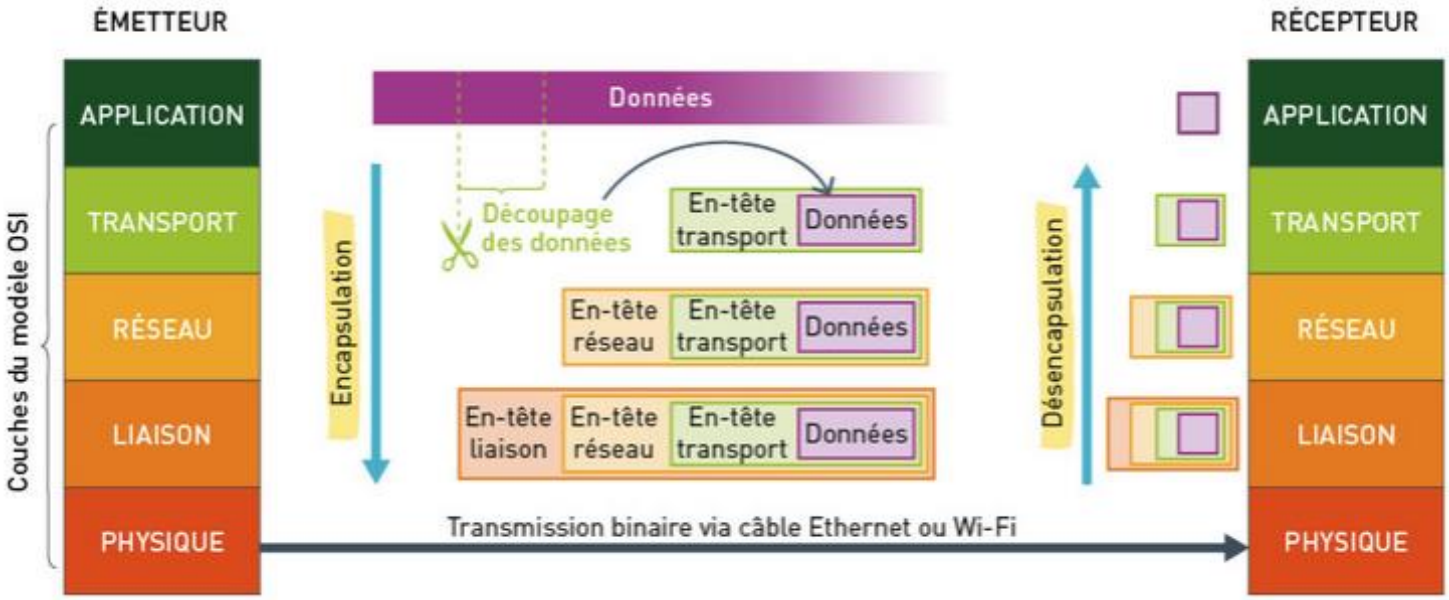
Exercice 2 (3 points)



1) En observant le schéma 1, expliquer pourquoi l'envoi de données en flux continu est source d'inconvénients.

2) En observant le schéma 2, expliquer deux avantages du découpage des données en paquets.

3) En vous aidant du schéma ci-dessous, expliquer en quoi consiste le travail de la couche de transport



Exercice 3 (7 points)

Partie 1

On dispose d'un fichier csv nommé `departements.csv` dont voici les 20 premières lignes

code_departement	nom_departement	code_region	nom_region
1	Ain	84	Auvergne-Rhône-Alpes
2	Aisne	32	Hauts-de-France
3	Allier	84	Auvergne-Rhône-Alpes
4	Alpes-de-Haute-Provence	93	Provence-Alpes-Côte d'Azur
5	Hautes-Alpes	93	Provence-Alpes-Côte d'Azur
6	Alpes-Maritimes	93	Provence-Alpes-Côte d'Azur
7	Ardèche	84	Auvergne-Rhône-Alpes
8	Ardennes	44	Grand Est
9	Ariège	76	Occitanie
10	Aube	44	Grand Est
11	Aude	76	Occitanie
12	Aveyron	76	Occitanie
13	Bouches-du-Rhône	93	Provence-Alpes-Côte d'Azur
14	Calvados	28	Normandie
15	Cantal	84	Auvergne-Rhône-Alpes
16	Charente	75	Nouvelle-Aquitaine
17	Charente-Maritime	75	Nouvelle-Aquitaine
18	Cher	24	Centre-Val de Loire
19	Corrèze	75	Nouvelle-Aquitaine

La correspondance entre le nom de la région et le code région est donné par :

```
'Auvergne-Rhône-Alpes': '84',
'Hauts-de-France': '32',
"Provence-Alpes-Côte d'Azur": '93',
'Grand Est': '44',
'Occitanie': '76',
'Normandie': '28',
'Nouvelle-Aquitaine': '75',
'Centre-Val de Loire': '24',
'Bourgogne-Franche-Comté': '27',
'Bretagne': '53',
'Corse': '94',
'Pays de la Loire': '52',
'Île-de-France': '11',
'Guadeloupe': '1',
'Martinique': '2',
'Guyane': '3',
'La Réunion': '4',
'Mayotte': '6'
```

On dispose également d'un fichier Python `exercice.py` placé dans le même répertoire que le fichier `departements.csv` et qui contient le code suivant :

```
import csv

#ouverture du fichier csv
def lecture_fichier(nom_fichier):
    with open(nom_fichier, mode='r', encoding='utf-8-sig') as fichier_ouvert:
        return [ligne for ligne in csv.reader(fichier_ouvert)]
```

1) Quels sont les descripteurs de la table ?
2) On souhaite récupérer le contenu du fichier csv dans le programme Python.

Quelle instruction Python faut-il écrire afin de lire le fichier `departements.csv` et d'affecter son contenu dans la variable appelée `table_departements` ?

3) Quel est le type de la variable `table_departements` ?
4) On ajoute la fonction `selectionner(table, critere)` au programme Python :

```
def selectionner(table, critere):
    selection=[]
    for i in range(1,len(table)):
        if critere(table[i]):
            selection.append(table[i])
    return selection
```

On exécute la ligne :

`recherche_1 = selectionner(table_departements, lambda x: x[2] == '52')`

Que va contenir la variable `recherche_1` ? Expliquez votre réponse.

5) Grâce à la fonction `selectionner(table, critere)` du programme, donner le critère de sélection à écrire en argument pour renvoyer uniquement les départements dont le code région est strictement inférieur à 10 et affecter le résultat de la recherche dans une variable nommée `recherche_2`.
6) Grâce à la fonction `selectionner(table, critere)` du programme, donner le critère de sélection à écrire en argument pour renvoyer uniquement les départements situés en Normandie ou Bretagne dans une variable nommée `recherche_3`.

Partie 2

On dispose d'un fichier csv nommé `communes.csv` dans le même répertoire que le fichier `exercice.py`. Ce fichier contient toutes les communes de France.

On a importé dans le programme Python le fichier `communes.csv` en exécutant l'instruction

`table_communes = lecture_fichier('communes.csv').`

Cela a permis d'affecter à la variable `table_communes` la valeur :

```
[['code_commune_INSEE', 'nom_commune_postal', 'code_postal',
'libelle_acheminement', 'ligne_5', 'latitude', 'longitude', 'code_commune',
'article', 'nom_commune', 'nom_commune_complet', 'code_departement',
'nom_departement', 'code_region', 'nom_region'], ['1001', 'L ABERGEMENT
CLEMENCIAT', '1400', 'L ABERGEMENT CLEMENCIAT', '', '46.1534255214',
'4.92611354223', '1', "L'", 'Abergement-Clémenciat', "L'Abergement-
Clémenciat", '1', 'Ain', '84', 'Auvergne-Rhône-Alpes'],                ,

            , ['99138', 'MONACO', '98000', 'MONACO', '', '', '', '138', '',
'Monaco', 'Monaco', '99', '', '', '']]
```

(on ne voit ici que le début et la fin de cette valeur)

1) Écrire une instruction Python qui permet d'affecter à une variable nommée `les_descripteurs` la liste des descripteurs de la table `table_communes`.

2) Il est possible de trier la table grâce à la fonction `sorted()` intégrée à Python et à l'utilisation d'une fonction lambda. Que va contenir la liste de listes `s1` après l'instruction suivante ?

```
s1 = sorted(table_communes[1:], key=lambda ligne: ligne[9])
```

On ne demande pas d'écrire la liste de listes `s1` complètement, mais de décrire ce qu'elle contient.

3) Même question avec

```
s2 = sorted(table_communes[1:], key=lambda ligne: float(ligne[5]),
reverse=True)
```

4) Écrire une instruction Python qui permet d'affecter à une variable nommée `ma_selection` les cinq communes ayant la latitude la plus grande.

5) Soit la variable `s3` la liste de listes `table_communes` rangée par ordre alphabétique des noms de commune postal.

   a. Écrire l'instruction Python permettant d'obtenir `s3`.

   b. Écrire l'instruction Python permettant d'obtenir à partir de la table `s3` uniquement les communes de la Mayenne.

Exercice 4 (6 points)

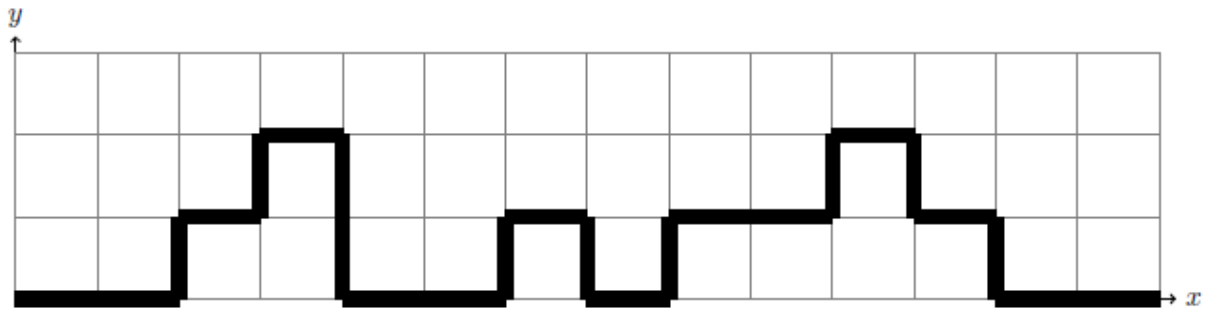
Un jeu de plateforme à une seule dimension est modélisé par un paysage parsemé d'obstacles sur lesquels le personnage doit monter ou descendre. Le personnage se déplace de la gauche vers la droite à chaque étape. Pour que le niveau soit faisable, il doit respecter 2 règles :

- Le personnage ne peut pas monter plus d'1 étage à chaque étape.
- Le personnage ne peut pas sauter vers le bas de plus de 2 étages à chaque étape.

Si les deux règles sont respectées, alors on dit que le niveau est **conforme**.

De plus, on mesure la difficulté par la somme de tous les étages **montés ou descendus**.

Exemple (Niveau 1 conforme de 14 étapes)

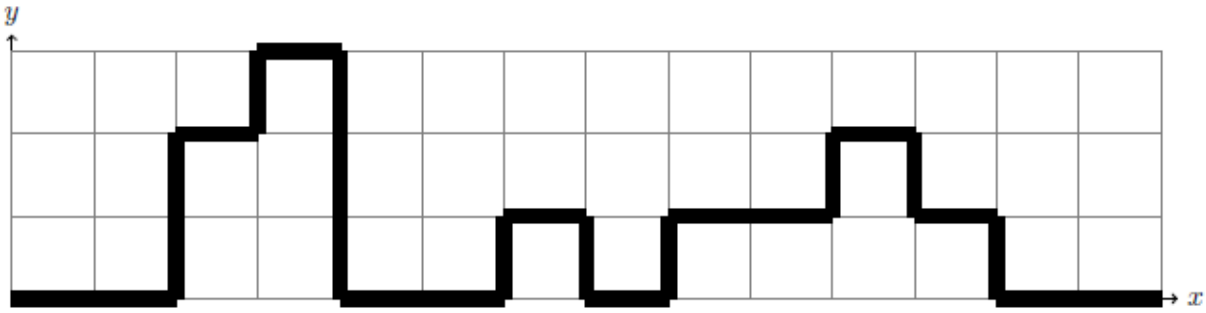


Ce niveau est modélisé par la liste :

```
niveau1 = [0, 0, 1, 2, 0, 0, 1, 0, 1, 1, 2, 1, 0, 0]
```

- Ce niveau est **conforme**.
- $d = 0 + 0 + 1 + 2 + 0 + 0 + 1 + 0 + 1 + 1 + 2 + 1 + 0 + 0 = 9$ donc sa difficulté est 9.

Exemple (Niveau 2 non conforme de 14 étapes)



Ce niveau est modélisé par la liste :

```
niveau2 = [0, 0, 2, 3, 0, 0, 1, 0, 1, 1, 2, 1, 0, 0]
```

- Ce niveau est **non conforme**.

1) On souhaite modifier le niveau 2 pour qu'il devienne conforme.

Pour cela on exécute les deux instructions ci-dessous.

```
niveau2[...] = ...
niveau2[...] = ...
```

Recopier ces instructions sur la copie en les complétant.

2) Avant d'intégrer un niveau au jeu, on souhaite vérifier sa conformité.

Pour cela on code une fonction `conforme(niveau)` qui prend en argument la variable `niveau` de type liste d'entiers et qui renvoie la valeur booléenne True si le niveau est conforme et False sinon.

Exemples :

```
>>> niveau1 = [0, 0, 1, 2, 0, 0, 1, 0, 1, 1, 2, 1, 0, 0]
>>> conforme(niveau1)
True
>>> niveau2 = [0, 0, 2, 3, 0, 0, 1, 0, 1, 1, 2, 1, 0, 0]
>>> conforme(niveau2)
False
```

Recopier toutes les lignes du code suivant sur la copie en les complétant.

```
def conforme(niveau):
    for i in range(.............):
        if niveau[i+1] - niveau[i] > 1:
            return ...
        elif niveau[i+1] - niveau[i] < ...:
            return ...
    return ...
```

3) Ecrire une fonction `difficulte(niveau)` qui prend en argument la variable `niveau` représentant un niveau conforme et qui renvoie la variable d de type entier telle que définie plus haut.

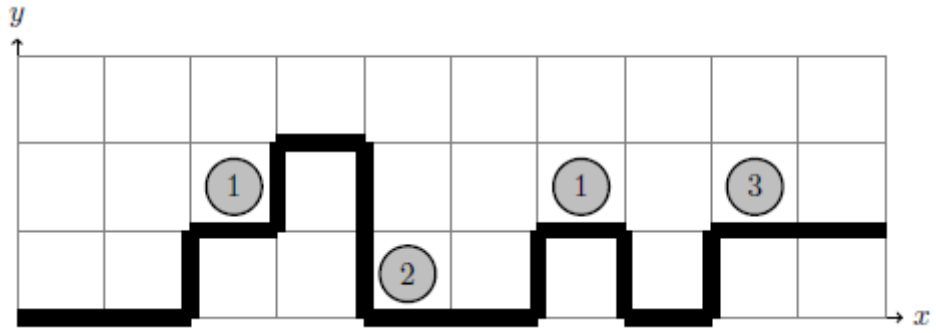
Exemple :

```
>>> niveau1 = [0, 0, 1, 2, 0, 0, 1, 0, 1, 1, 2, 1, 0, 0]
>>> difficulte(niveau1)
9
```

4) On dispose maintenant des pièces sur le parcours. Chaque pièce possède une valeur. Il y a 0 ou 1 pièce à chaque étape. Le joueur marque les points correspondant aux valeurs des pièces lorsqu'il les ramasse.

Chaque étape du parcours est maintenant représentée par une liste de deux entiers de la forme `[hauteur, valeur_piece]`. S'il n'y a pas de pièce à une étape alors la valeur associée à la pièce sera nulle.

Exemple (Niveau 3 conforme de 10 étapes avec des pièces)



Ce niveau est modélisé par la liste de listes :

```
niveau3 = [[0, 0], [0, 0], [1, 1], [2, 0], [0, 2], [0, 0], [1, 1], [0, 0], [1, 3], [1, 0]]
```

On veut modifier la valeur d'une pièce dans le niveau donné en exemple ci-dessus. Donner l'instruction permettant de changer la pièce de valeur 3 en une pièce de valeur 1.

5) On suppose que le joueur atteint l'étape x du niveau sans être éliminé. x désigne le numéro de l'étape en commençant avec x valant 0 pour la première étape.

Écrire une fonction `score(niveau, x)` qui prend les paramètres `niveau` de type liste de listes et x de type entier. La fonction doit renvoyer la variable `score` de type entier. C'est le score réalisé par le joueur à la fin du niveau en supposant qu'il ait ramassé toutes les pièces jusqu'à l'étape x incluse.

Exemple :

```
>>> niveau3 = [[0, 0], [0, 0], [1, 1], [2, 0], [0, 2], [0, 0], [1, 1], [0, 0], [1, 3], [1, 0]]
>>> score(niveau3, 4)
3
```

6) Les meilleurs scores sont mémorisés dans un dictionnaire de la forme :

```
meilleurs_scores = {'Alice': 25, 'Bob': 18, 'John': 63, 'Alan': 21}
```

Alice vient de battre son record avec un score de 32. Donner l'instruction permettant de mettre à jour le dictionnaire `meilleurs_scores`

7) On considère la fonction définie ci-dessous qui prend en argument le dictionnaire `meilleurs_scores`.

Expliquez ce que fait cette fonction.

```
def mystere(dico):
    x = 0
    y = ""
    for i in dico:
        if dico[i] > x:
            x = dico[i]
            y = i
    return y
```