Lycée Les 3 Sources — **Bourg-Lès-Valence**
NSI 1re — Année **2025-26**

Devoir surveillé n°1 – correction

Pour les deux exerices suivants, on pourra utiliser les éléments ci-dessous.

$4^3 = 64$ $4^2 = 16$ $4^1 = 4$ $4^0 = 1$
$6^2 = 36$ $6^1 = 6$ $6^0 = 1$
$16^1 = 16$ $16^0 = 1$

Décimal	128	64	32	16	8	4	2	1
182	1	0	1	1	0	1	1	0
205	1	1	0	0	1	1	0	1

Exercice 1 : (2pt) Convertir en base 10 les nombres ci-dessous :
1) 10110110_2 = 182_{10} 2) $2231_4 = 173_{10}$ 3) $513_6 = 189_{10}$ 4) $5D_{16} = 93_{10}$

Solution :
1) Voir le tableau.
2) $2 \times 64 + 2 \times 16 + 3 \times 4 + 1 \times 1 = 173$
3) $5 \times 36 + 1 \times 6 + 3 \times 1 = 189$
4) $5 \times 16 + 13 = 93$

Exercice 2 : (2pt) Convertir dans la base demandée les nombres suivants :
1) 205_{10} en binaire 2) 109_{10} en base 4 3) 134_{10} en base 6 4) 184_{10} en base 16

Solution :
1) $205_{10} = 11001101_2$ d'après le tableau.

2)
$109 : 4 = 27$ reste 1
$27 : 4 = 6$ reste 3
$6 : 4 = 1$ reste 2
$1 : 4 = 0$ reste 1
$1\,2\,3\,1$

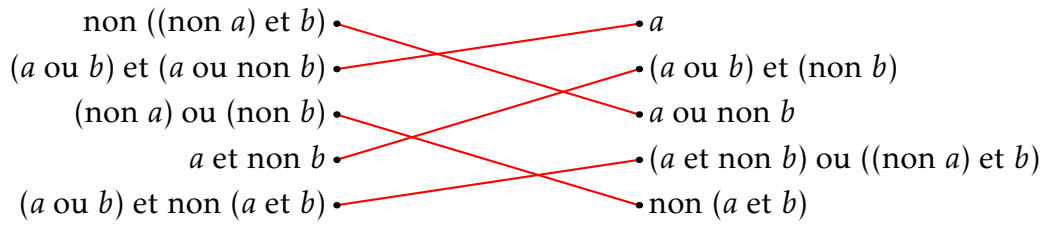
3)
$134 : 6 = 22$ reste 2
$22 : 6 = 3$ reste 4
$3 : 6 = 0$ reste 3
$3\,4\,2$

4)
$184 : 16 = 11$ reste 8
$11 : 16 = 0$ reste 11
$B\,8$

Exercice 3 : (1,5pt) Chacune des questions suivantes est indépendante. Les réponses doivent être justifiées brièvement.

1) Quel est le plus grand entier, donné en base 10, qui peut être représenté en binaire avec 4 bits ?
   Solution : Le plus grand nombre pouvant être écrit avec 4 bits est $1111_2 = 15_{10}$.
2) Donner le plus petit et le plus grand nombre, exprimés en base 10, qui s'écrivent en binaire avec 10 chiffres, donc le premier (celui de gauche) est un 1 ? Indication, on pourra réfléchir aux colonnes nécessaires dans le tableau de conversion.
   Solution : En haut de la 10^e colonne du tableau de conversion, il y aura 512. Dans la 11^e colonne, il y aura 1024. Avec 9 chiffres, on peut aller de 512 à 1023.
3) Combien de lignes faut-il dans la table de vérité d'une expression booléenne avec 3 booléens, comme "a ou b ou c", pour décrire toutes les valeurs possibles de chaque variable ? On ne demande pas d'écrire ces lignes.
   Solution : Il y a 8 combinaisons possibles, donc 8 lignes.

Exercice 4 : (2pt) Relier chacune des expressions booléennes de gauche avec celle de droite qui est équivalente.



Exercice 5 : (2pt) On souhaite réaliser des fonctions Python `et`, `ou`, `non` et `xor` (dont on rappelle la table de vérité ci-contre) qui correspondent aux opérations booléennes de même nom. On a déjà écrit le code de ces fonctions mais le nom de chacune a été effacé. Rajouter `et(a, b)`, `ou(a, b)`, `non(a)` et `xor(a, b)` au début de la fonction correspondante. Attention, une cinquième fonction est venue se glisser parmi les autres. Vous n'avez pas à déterminer ce qu'elle fait.

a	b	a xor b
0	0	0
0	1	1
1	0	1
1	1	0

```
def non(a):
    if a:
        return False
    else:
        return True
```

```
def et(a, b):
    if a:
        return b
    else:
        return False
```

```
def ou(a, b):
    if a:
        return True
    else:
        return b
```

```
def equivalent(a, b):
    if a:
        return b
    else:
        if b:
            return False
        else:
            return True
```

```
def xor(a, b):
    if a:
        if b:
            return False
        else:
            return True
    else:
        return b
```

Exercice 6 : (2pt) On considère les 8 instructions Python ci-dessous :

```
for s in texte:
```

```
c = 0
```

```
return c
```

```
return True
```

```
if s == symbole:
```

```
c = c + 1
```

```
return -1
```

```
return False
```

À l'aide de certaines de ces instructions, et aucune autre, compléter le code des fonctions `compter` et `appartient` telles que :

- `compter(symbole, texte)` renvoie le nombre de fois où `symbole` apparaît dans `texte`.
- `appartient(symbole, texte)` renvoie **`True`** si `symbole` apparaît dans `texte` et **`False`** sinon.

```
def compter(symbole, texte):
    c = 0
    for s in texte:
        if s == symbole:
            c = c + 1
    return c
```

```
def appartient(symbole, texte):
    for s in texte:
        if s == symbole:
            return True
    return False
```

Exercice 7 : (3pt) On considère la fonction ci-dessous.

```
def mystere1(n):
    res = 0
    for i in range(n):
        if i%2 == 0:
            res = res + i
        else:
            res = res - i
    return res
```

i	i%2	i%2==0	res
			0
0	0	True	0
1	1	False	−1
2	0	True	1
3	1	False	−2
4	0	True	2
5	1	False	−3

1) On considère l'appel `mystere1(6)`. Remplir le tableau ci-contre. La première ligne correspond à l'état de la mémoire après l'exécution de la ligne 2. Les lignes suivantes correspondent à l'état de la mémoire après la ligne 7.
2) Encadrer le résultat renvoyé pour cet appel.
3) Quelle est la valeur renvoyée par `mystere1(20)` ? -10

Exercice 8 : (3pt) On considère la fonction ci-dessous.

```
def mystere2(n):
    u = 1
    while u < n:
        u = 2*u + 1
    return u
```

u	u < n
1	True
3	True
7	True
15	True
31	True
63	False

1) Compléter le tableau ci-contre pour l'exécution de `mystere2(50)` et entourer la valeur renvoyée à la fin.
2) Quelle est la valeur renvoyée par `mystere2(1000)` ? On pourra comparer les valeurs successives de u avec des puissances de 2.
Solution : On remarque que u est toujours égale à une puissance de 2, moins 1. La plus petite puissance de 2 supérieure à 1000 est 1024. Le résultat est donc 1023.

Exercice 9 : (1pt) Pour chacune des questions suivantes, entourer la bonne réponse.

1) Déterminer quelle version de la fonction `mirroir` permet de renvoyer un texte correspondant au texte passé en paramètre écrit à l'envers.
Par exemple `mirroir("NSI")` doit renvoyer `"ISN"`.

a)
```
def miroir(texte):
    res = ""
    for s in texte:
        res = s + res
    return res
```

b)
```
def miroir(texte):
    res = ""
    for s in texte:
        res = res + s
    return res
```

c)
```
def miroir(texte):
    res = ""
    for s in texte:
        res = s
    return res
```

d)
```
def miroir(texte):
    res = ""
    for s in texte:
        res = res
    return res
```

2) On considère la fonction `decompte` définie ci-contre.
Que va afficher `decompte(3)` ?
On rappelle que `range(k)` va renvoyer successivement tous les entiers entre 0 et `k` – 1 inclus.

```
def decompte(n):
    for i in range(n):
        print(n-i)
    print("Partez !")
```

a)
```
0
1
2
Partez !
```

b)
```
1
2
3
Partez !
```

c)
```
2
1
0
Partez !
```

d)
```
3
2
1
Partez !
```

Exercice 10 : (3pt) Lorsqu'on essaie de parler la bouche grande ouverte chez le dentiste, il ne reste que les voyelles.
Nous allons écrire une fonction `dentiste` qui prend une chaîne de caractères `texte` et qui renvoie un texte ne contenant que les voyelles de `texte`, dans le même ordre que dans `texte`. Les voyelles sont `a`, `e`, `i`, `o`, `u` et `y`. Les ponctuations et espaces sont également supprimés.

```
>>> dentiste("j'ai vraiment mal")
'aiaiea'
>>> dentiste("c'est bientot fini ?")
'eieoii'
```

1) Donner le résultat des instructions suivantes :

```
>>> dentiste("il fait froid")
"iaioi"
>>> dentiste("")
""
```

2) Pour écrire le code de la fonction `dentiste`, on pourra s'inspirer de la fonction `recopier` qui prend une chaîne de caractère `texte` et qui renvoie une copie de ce texte. On part du texte vide et on parcourt tous les symboles de `texte` en les rajoutant à droite de `resultat`.

```
def recopier(texte):
    resultat = ""
    for symbole in texte:
        resultat = resultat + symbole
    return resultat
```

On suppose qu'on dispose d'une fonction `voy` qui prend un symbole `symbole` et qui renvoie un booléen indiquant si `symbole` est une voyelle ou non.

```
>>> voy("a")
True
>>> voy("m")
False
```

Il n'est pas demandé d'écrire le code de `voy`.

Compléter le code de la fonction `dentiste` ci-dessous, en utilisant `voy` :

```
def dentiste(texte):
    resultat = ""
    for symbole in texte:
        if voy(symbole):
            resultat = resultat + symbole
    return resultat
```

Lycée Les 3 Sources — **Bourg-Lès-Valence**
NSI 1re — Année **2025-26**

Devoir surveillé n°1 – correction

Pour les deux exerices suivants, on pourra utiliser les éléments ci-dessous.

$4^3 = 64$ $4^2 = 16$ $4^1 = 4$ $4^0 = 1$
$6^2 = 36$ $6^1 = 6$ $6^0 = 1$
$16^1 = 16$ $16^0 = 1$

Décimal	128	64	32	16	8	4	2	1
214	1	1	0	1	0	1	1	0
203	1	1	0	0	1	0	1	1

Exercice 1 : (2pt) Convertir en base 10 les nombres ci-dessous :
1) $1101\,0110_2$ = 214_{10} 2) $2213_4 = 167_{10}$ 3) $531_6 = 199_{10}$ 4) $5C_{16} = 92_{10}$

Solution :
1) Voir le tableau.
2) $2 \times 64 + 2 \times 16 + 1 \times 4 + 3 \times 1 = 167$
3) $5 \times 36 + 3 \times 6 + 1 \times 1 = 199$
4) $5 \times 16 + 12 = 92$

Exercice 2 : (2pt) Convertir dans la base demandée les nombres suivants :
1) 203_{10} en binaire 2) 121_{10} en base 4 3) 124_{10} en base 6 4) 194_{10} en base 16

Solution :
1) $203_{10} = 11001011_2$ d'après le tableau.

2)
$121 : 4 = 30$ reste 1
$30 : 4 = 7$ reste 2
$7 : 4 = 1$ reste 3
$1 : 4 = 0$ reste 1
$1\,3\,2\,1$

3)
$124 : 6 = 20$ reste 4
$20 : 6 = 3$ reste 2
$3 : 6 = 0$ reste 3
$3\,2\,4$

4)
$194 : 16 = 12$ reste 2
$12 : 16 = 0$ reste 12
$C\,2$

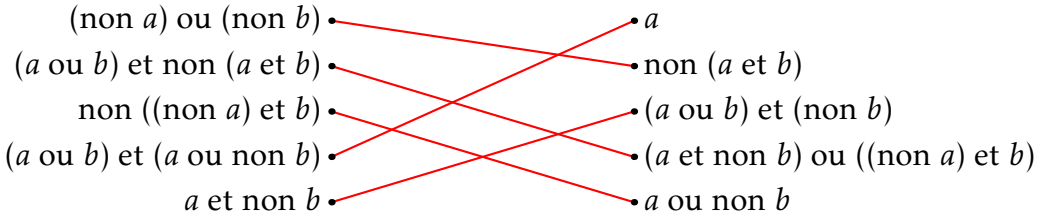
Exercice 3 : (1,5pt) Chacune des questions suivantes est indépendante. Les réponses doivent être justifiées brièvement.

1) Quel est le plus grand entier, donné en base 10, qui peut être représenté en binaire avec 4 bits ?
Solution : Le plus grand nombre pouvant être écrit avec 4 bits est $1111_2 = 15_{10}$.

2) Donner le plus petit et le plus grand nombre, exprimés en base 10, qui s'écrivent en binaire avec 10 chiffres, donc le premier (celui de gauche) est un 1 ? Indication, on pourra réfléchir aux colonnes nécessaires dans le tableau de conversion.
Solution : En haut de la 10^e colonne du tableau de conversion, il y aura 512. Dans la 11^e colonne, il y aura 1024. Avec 9 chiffres, on peut aller de 512 à 1023.

3) Combien de lignes faut-il dans la table de vérité d'une expression booléenne avec 3 booléens, comme "a ou b ou c", pour décrire toutes les valeurs possibles de chaque variable ? On ne demande pas d'écrire ces lignes.
Solution : Il y a 8 combinaisons possibles, donc 8 lignes.

Exercice 4 : (2pt) Relier chacune des expressions booléennes de gauche avec celle de droite qui est équivalente.



Exercice 5 : (2pt) On souhaite réaliser des fonctions Python `et`, `ou`, `non` et `xor` (dont on rappelle la table de vérité ci-contre) qui correspondent aux opérations booléennes de même nom. On a déjà écrit le code de ces fonctions mais le nom de chacune a été effacé. Rajouter `et(a, b)`, `ou(a, b)`, `non(a)` et `xor(a, b)` au début de la fonction correspondante. Attention, une cinquième fonction est venue se glisser parmi les autres. Vous n'avez pas à déterminer ce qu'elle fait.

a	b	a xor b
0	0	0
0	1	1
1	0	1
1	1	0

```
def non(a):
    if a:
        return False
    else:
        return True
```

```
def ou(a, b):
    if a:
        return True
    else:
        return b
```

```
def et(a, b):
    if a:
        return b
    else:
        return False
```

```
def equivalent(a, b):
    if a:
        return b
    else:
        if b:
            return False
        else:
            return True
```

```
def xor(a, b):
    if a:
        if b:
            return False
        else:
            return True
    else:
        return b
```

Exercice 6 : (2pt) On considère les 8 instructions Python ci-dessous :

```
for s in texte:
```

```
c = 0
```

```
return c
```

```
return True
```

```
if s == symbole:
```

```
c = c + 1
```

```
return -1
```

```
return False
```

À l'aide de certaines de ces instructions, et aucune autre, compléter le code des fonctions `compter` et `appartient` telles que :

- `compter(symbole, texte)` renvoie le nombre de fois où `symbole` apparaît dans `texte`.
- `appartient(symbole, texte)` renvoie `True` si `symbole` apparaît dans `texte` et `False` sinon.

```
def appartient(symbole, texte):
    for s in texte:
        if s == symbole:
            return True
    return False
```

```
def compter(symbole, texte):
    c = 0
    for s in texte:
        if s == symbole:
            c = c + 1
    return c
```

Exercice 7 : (3pt) On considère la fonction ci-dessous.

```
def mystere1(n):
    res = 0
    for i in range(n):
        if i%2 == 0:
            res = res - i
        else:
            res = res + i
    return res
```

i	i%2	i%2==0	res
			0
0	0	True	0
1	1	False	1
2	0	True	−1
3	1	False	2
4	0	True	−2
5	1	False	3

1) On considère l'appel `mystere1(6)`. Remplir le tableau ci-contre. La première ligne correspond à l'état de la mémoire après l'exécution de la ligne 2. Les lignes suivantes correspondent à l'état de la mémoire après la ligne 7.
2) Encadrer le résultat renvoyé pour cet appel.
3) Quelle est la valeur renvoyée par `mystere1(20)` ? 10

Exercice 8 : (3pt) On considère la fonction ci-dessous.

```
def mystere2(n):
    u = 1
    while u < n:
        u = 2*u + 1
    return u
```

u	u < n
1	True
3	True
7	True
15	True
31	True
63	False

1) Compléter le tableau ci-contre pour l'exécution de `mystere2(50)` et entourer la valeur renvoyée à la fin.
2) Quelle est la valeur renvoyée par `mystere2(1000)` ? On pourra comparer les valeurs successives de u avec des puissances de 2.
Solution : On remarque que u est toujours égale à une puissance de 2, moins 1. La plus petite puissance de 2 supérieure à 1000 est 1024. Le résultat est donc 1023.

Exercice 9 : (1pt) Pour chacune des questions suivantes, entourer la bonne réponse.

1) Déterminer quelle version de la fonction `mirroir` permet de renvoyer un texte correspondant au texte passé en paramètre écrit à l'envers.
Par exemple `mirroir("NSI")` doit renvoyer `"ISN"`.

a)
```
def miroir(texte):
    res = ""
    for s in texte:
        res = res + s
    return res
```

b)
```
def miroir(texte):
    res = ""
    for s in texte:
        res = s + res
    return res
```

c)
```
def miroir(texte):
    res = ""
    for s in texte:
        res = res
    return res
```

d)
```
def miroir(texte):
    res = ""
    for s in texte:
        res = s
    return res
```

2) On considère la fonction `decompte` définie ci-contre.
Que va afficher `decompte(3)` ?
On rappelle que `range(k)` va renvoyer successivement tous les entiers entre 0 et `k` – 1 inclus.

```
def decompte(n):
    for i in range(n):
        print(n-i)
    print("Partez !")
```

a)
```
3
2
1
Partez !
```

b)
```
2
1
0
Partez !
```

c)
```
1
2
3
Partez !
```

d)
```
0
1
2
Partez !
```

Exercice 10 : (3pt) Lorsqu'on essaie de parler la bouche grande ouverte chez le dentiste, il ne reste que les voyelles.
Nous allons écrire une fonction `dentiste` qui prend une chaîne de caractères `texte` et qui renvoie un texte ne contenant que les voyelles de `texte`, dans le même ordre que dans `texte`. Les voyelles sont `a`, `e`, `i`, `o`, `u` et `y`. Les ponctuations et espaces sont également supprimés.

```
>>> dentiste("j'ai vraiment mal")
'aiaiea'
>>> dentiste("c'est bientot fini ?")
'eieoii'
```

1) Donner le résultat des instructions suivantes :

```
>>> dentiste("il fait chaud")
"iaiau"
>>> dentiste("")
""
```

2) Pour écrire le code de la fonction `dentiste`, on pourra s'inspirer de la fonction `recopier` qui prend une chaîne de caractère `texte` et qui renvoie une copie de ce texte. On part du texte vide et on parcourt tous les symboles de `texte` en les rajoutant à droite de `resultat`.

```
def recopier(texte):
    resultat = ""
    for symbole in texte:
        resultat = resultat + symbole
    return resultat
```

On suppose qu'on dispose d'une fonction `voy` qui prend un symbole `symbole` et qui renvoie un booléen indiquant si `symbole` est une voyelle ou non.

```
>>> voy("a")
True
>>> voy("m")
False
```

Il n'est pas demandé d'écrire le code de `voy`.

Compléter le code de la fonction `dentiste` ci-dessous, en utilisant `voy` :

```
def dentiste(texte):
    resultat = ""
    for symbole in texte:
        if voy(symbole):
            resultat = resultat + symbole
    return resultat
```