Spécialité NSI Première	**DEVOIR SURVEILLE DE**	Jeudi 6 octobre 2022
Lycée d'Avesnières	**NSI**	Durée : 55 mn
Année scolaire 2022-2023	**N° 2**	Calculatrice interdite

NOM : .....

Prénom : .....

Rendre l'énoncé avec la copie.

Exercice 1 (12 points)

En faisant une recherche avec les mots clés *schéma demi additionneur* Léo a trouvé le schéma suivant :

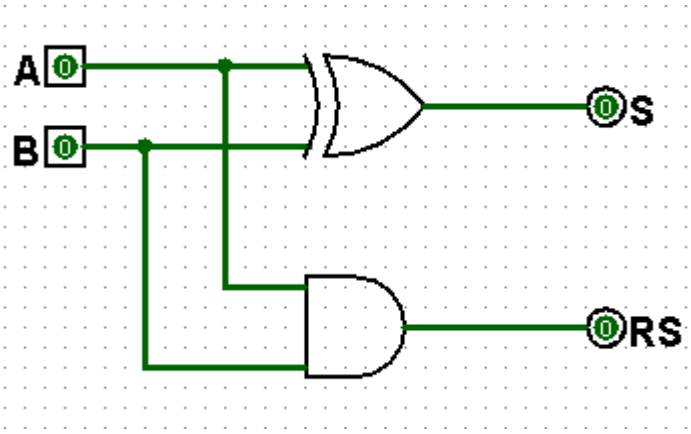


FIGURE 1. *demi additionneur 1 bit*

1. A l'aide d'un logiciel simulateur de circuits logiques, il essaie différentes combinaisons de 0 (`False`) et de 1 (`True`) sur les entrées A et B. Il observe les valeurs (0 ou 1) que prennent les variables somme (S) et retenue sortante (RS).

<u>Complétez directement sur l'énoncé</u> la table de vérité suivante avec les valeurs obtenues pour S et RS.

A	B	S	RS
0	0		
0	1		
1	0		
1	1		

2. Léo veut faire maintenant un additionneur complet (*full adder*) puisqu'il *accepte une retenue entrante* (RE).

La table de vérité de l'additionneur complet comporte 8 cas :

Quatre cas où la retenue de la colonne précédente, c'est-à-dire la "retenue entrante" vaut 0 :	Quatre cas où la retenue de la colonne précédente, c'est-à-dire la "retenue entrante" vaut 1 :
Retenue sortante 0 0 A 0 B + 0 Somme S 0	Retenue sortante 0 1 A 0 B + 0 Somme S 1
Retenue sortante 0 0 A 0 B + 1 Somme S 1	Retenue sortante 1 1 A 0 B + 1 Somme S 0
Retenue sortante 0 0 A 1 B + 0 Somme S 1	Retenue sortante 1 1 A 1 B + 0 Somme S 0
Retenue sortante 1 0 A 1 B + 1 Somme S 0	Retenue sortante 1 1 A 1 B + 1 Somme S 1

Expliquez le rôle de la retenue entrante.

3. On s'intéresse maintenant aux tables de vérité de l'additionneur complet.
   a. En entrant des 0 ou des 1 sur A, B et RE, **<u>complétez directement sur l'énoncé</u>** la table de vérité de S ci-après :

A	B	RE	S
0	0	0	
0	1	0	
1	0	0	
1	1	0	
0	0	1	
0	1	1	
1	0	1	
1	1	1	

b. En entrant des `0` ou des `1` sur A, B et RE, **<u>complétez directement sur l'énoncé</u>** la table de vérité de RS ci-dessous :

A	B	RE	RS
0	0	0	
0	1	0	
1	0	0	
1	1	0	
0	0	1	
0	1	1	
1	0	1	
1	1	1	

4. Léo a trouvé un schéma d'additionneur complet sur lequel **les carrés sont des entrées** et **les cercles des sorties** :

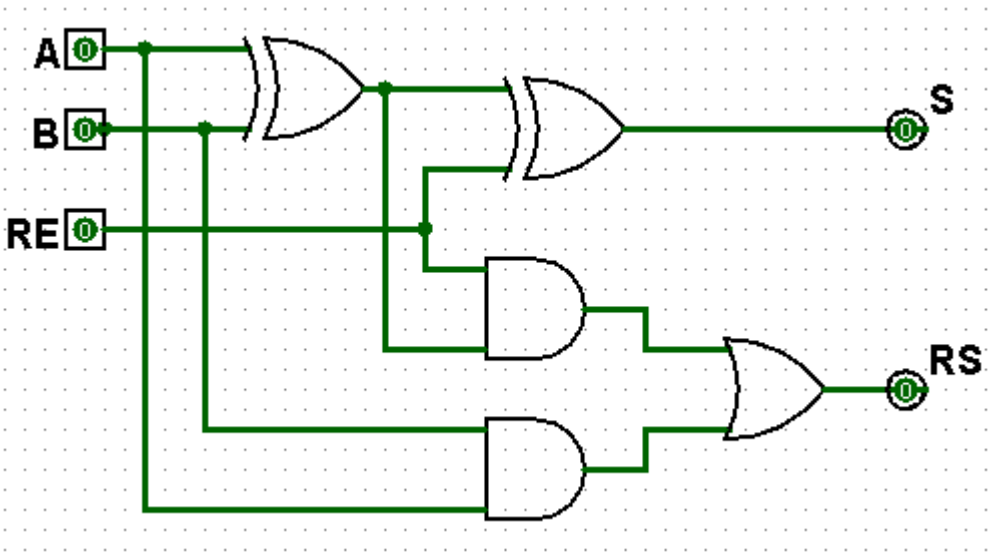


FIGURE 5. *Additionneur complet 1 bit*

Donnez les expressions logiques des variables S et RS en fonction des entrées A, B, RE. Vous utiliserez pour cela des connecteurs de la logique choisis parmi `and`, `or`, `xor` et `not`.

5. Pour plus de clarté les additionneurs complets 1 bit seront simplifiés selon le schéma ci-dessous :

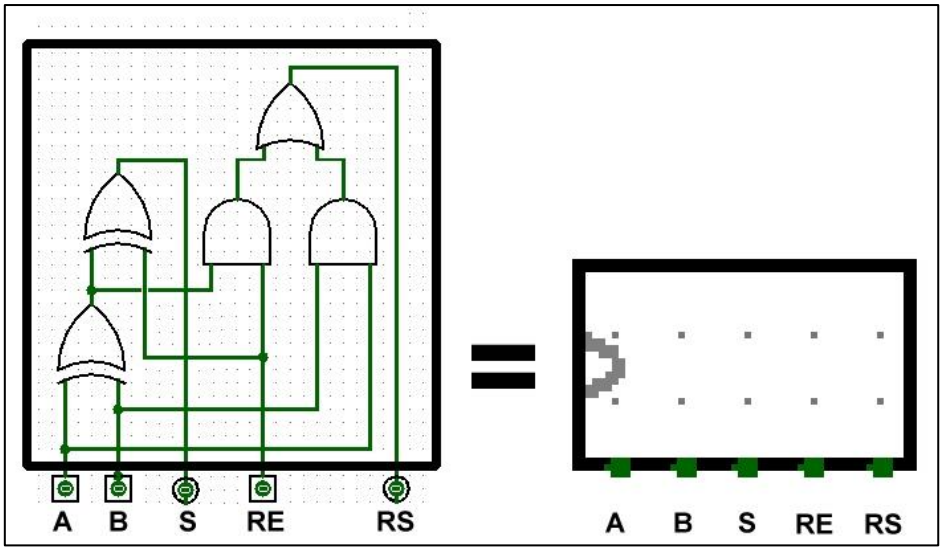


FIGURE 2. *Schéma simplifié de l'additionneur 4 bits*

Léo veut faire l'addition $A + B$ avec des nombres A et B de 4 bits. Pour cela il assemble quatre additionneurs 1 bit sur le schéma suivant. Les fils **vert foncé sont à la valeur 0** et les fils **vert clair sont à la valeur 1.**

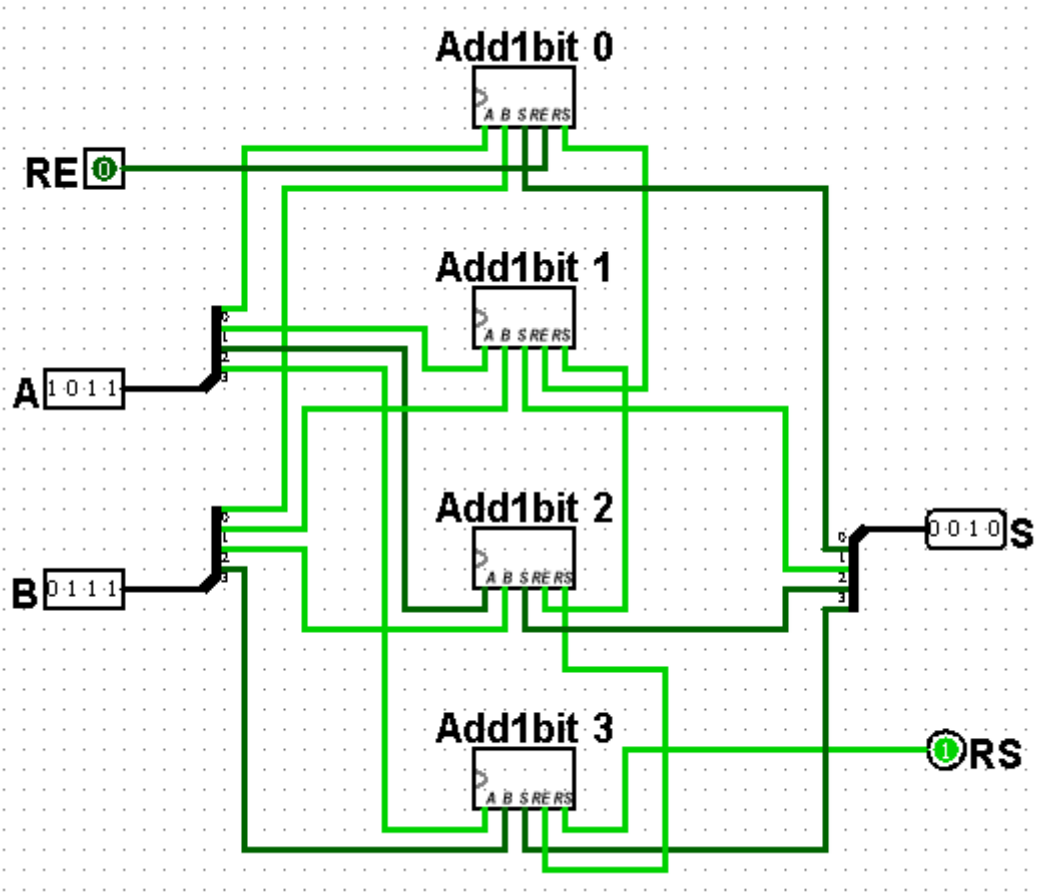


FIGURE 3. *Additionneur 4 bits*

Expliquez pourquoi la sortie S vaut `0010` et la sortie RS vaut `1`.

6. Léo a vu que, dans son simulateur de circuits logiques, il est possible de mettre des afficheurs hexadécimaux comme ci-dessous :

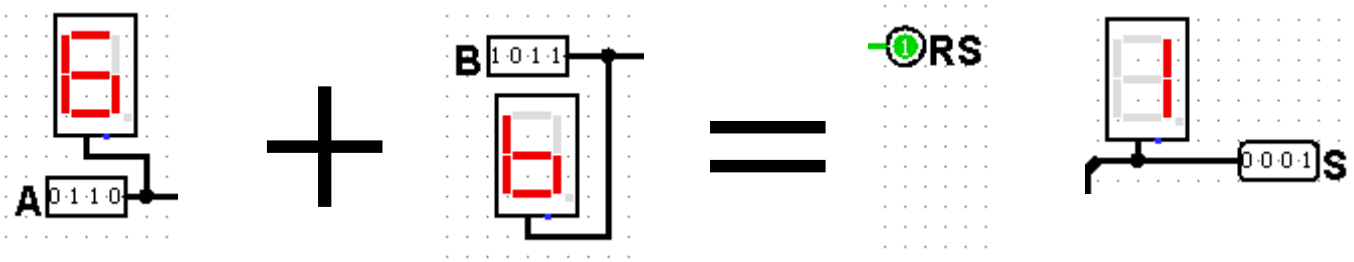


FIGURE 4. *Addition* $(0110)_2 + (1011)_2 = (10001)_2$ *c'est-à-dire* $(6)_{16} + (b)_{16} = (11)_{16}$

Il complète donc son schéma ainsi :

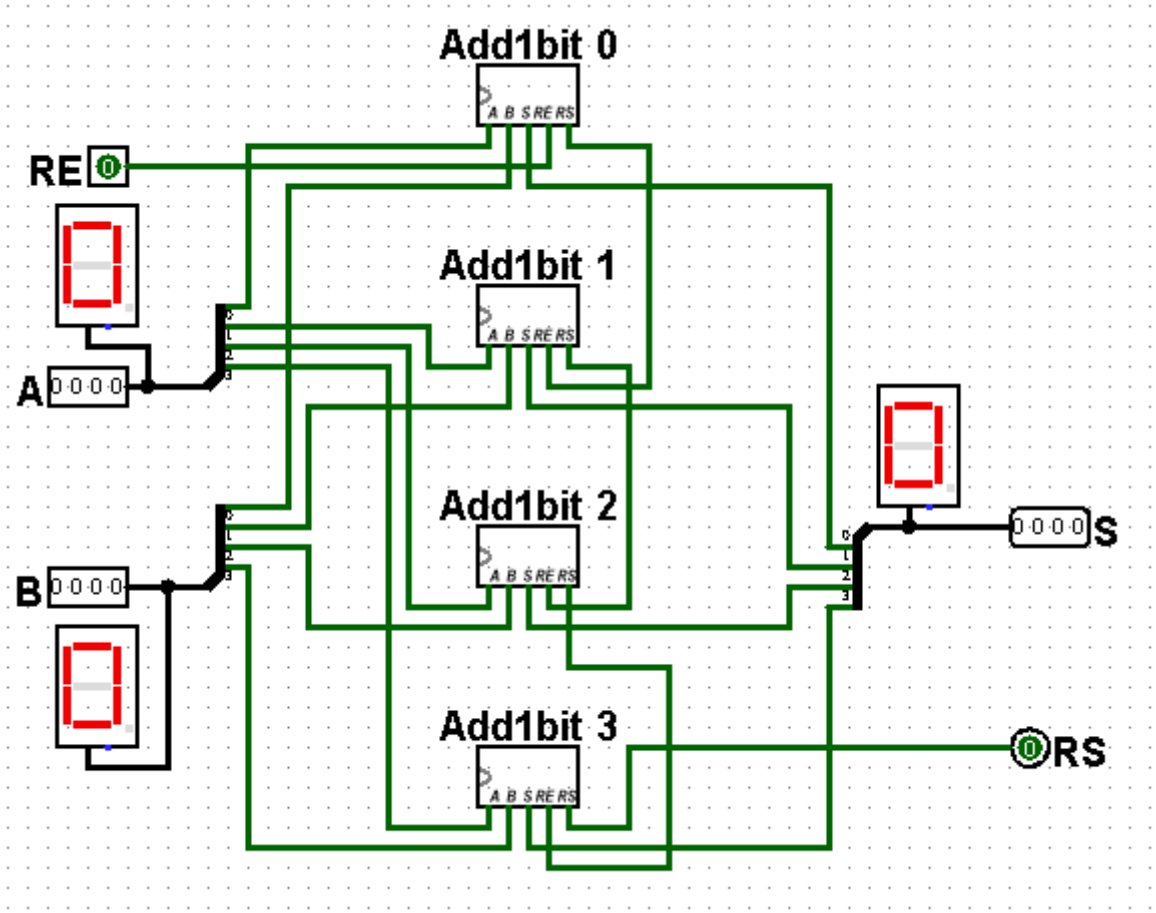


FIGURE 4. *Additionneur 4 bits avec afficheurs hexadécimaux*

S'il entre 1100 et 1111 comme valeurs respectives pour A et B :

a. Que voit-il affiché sur l'afficheur hexadécimal de l'entrée A ?
b. Que voit-il affiché sur l'afficheur hexadécimal de l'entrée B ?
c. Que voit-il affiché sur l'afficheur hexadécimal de la sortie S ?
d. Que vaut la retenue sortante RS ?

7. Léo veut maintenant réaliser la soustraction $A - B$ où $A \geq B$ et où les valeurs de A et B sont des entiers entre 0 et 7 inclus. Il ajoute quatre portes `not` pour B puisqu'il s'agit d'additionner l'opposé de B.

Il obtient le schéma suivant.

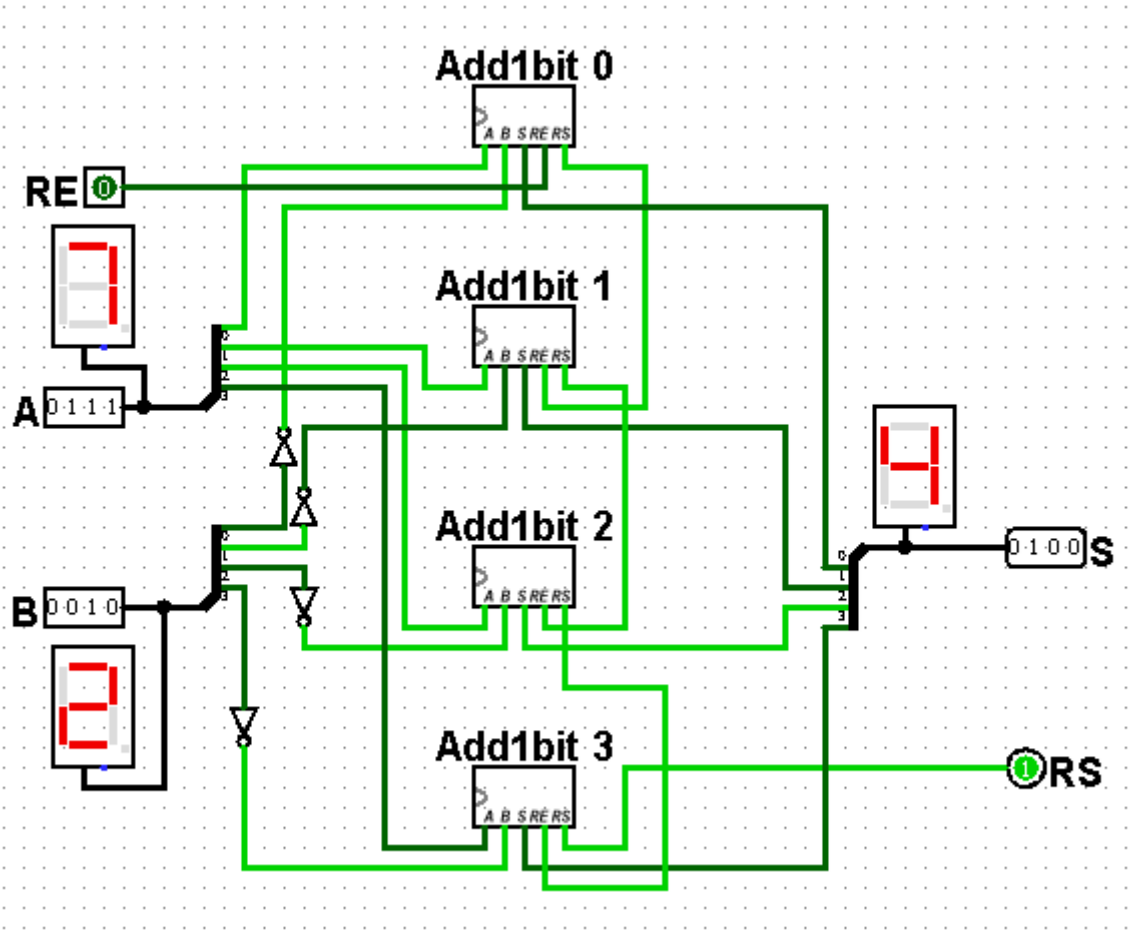


FIGURE 4. *Soustracteur de deux nombres A et B compris entre 0 et 7 avec $A \geq B$*

Léo essaye la soustraction $7 - 2$. Il ne comprend pas pourquoi il obtient 4. Pouvez-vous l'aider ?

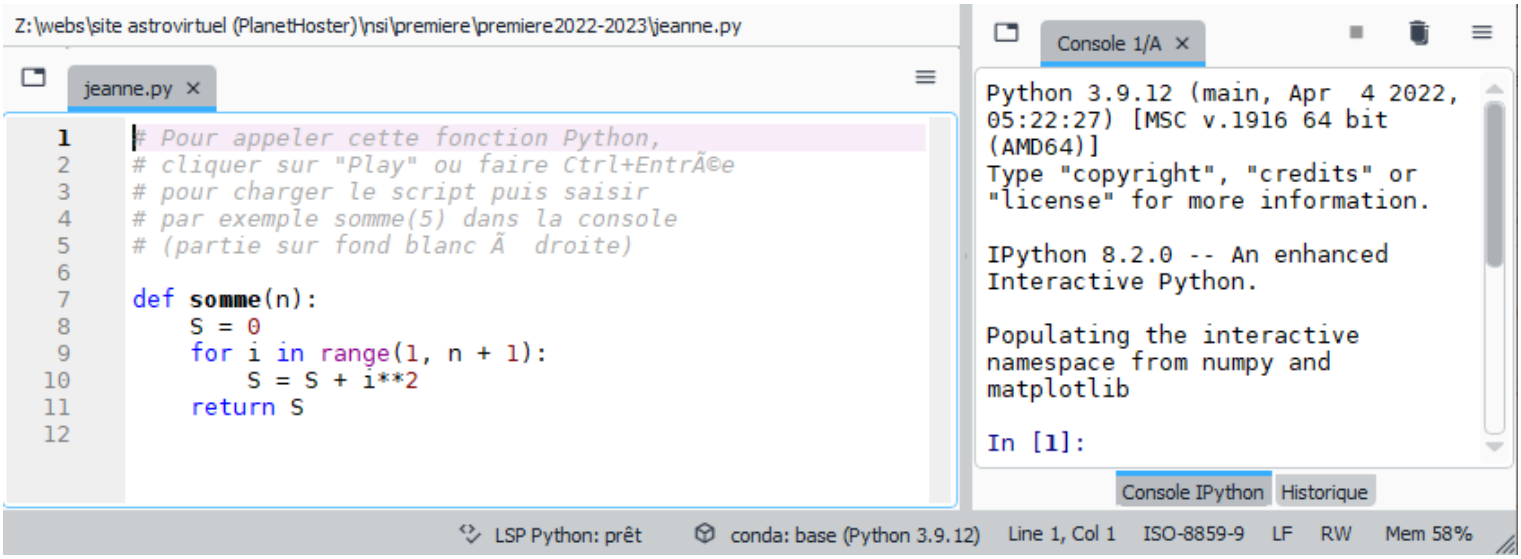
Exercice 2 (8 points)

1. Jeanne a trouvé un script sur son éditeur web Python favori :

```
01  # Pour appeler cette fonction Python,
02  # cliquer sur "Play" ou faire Ctrl+Entrée
03  # pour charger le script puis saisir
04  # par exemple somme(5) dans la console
05  # (partie sur fond blanc à droite)
06
07  def somme(n):
08      S = 0
09      for i in range(1, n + 1):
10          S = S + i**2
11      return S
```

Soucieuse de sauvegarder la fonction `somme(n)`, elle télécharge le script et l'enregistre dans son dossier personnel sur son ordinateur.
Elle nomme le fichier obtenu `jeanne.py`

Depuis l'environnement de développement Python installé sur son ordinateur, elle ouvre le fichier `jeanne.py` et elle voit l'affichage :



Que peut-elle remarquer ?

2. Jeanne a installé un éditeur hexadécimal qui lui permet de voir le contenu binaire de son fichier Python sous la forme d'une succession d'octets. L'octet 20 est le code du caractère "espace entre les mots".

jeanne.py

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Texte Décodé
00000000   23 20 50 6F 75 72 20 61 70 70 65 6C 65 72 20 63  # Pour appeler c
00000010   65 74 74 65 20 66 6F 6E 63 74 69 6F 6E 20 50 79  ette fonction Py
00000020   74 68 6F 6E 2C 0A 23 20 63 6C 69 71 75 65 72 20  thon,.# cliquer
00000030   73 75 72 20 22 50 6C 61 79 22 20 6F 75 20 66 61  sur "Play" ou fa
00000040   69 72 65 20 43 74 72 6C 2B 45 6E 74 72 C3 A9 65  ire Ctrl+EntrÃ©e
00000050   0A 23 20 70 6F 75 72 20 63 68 61 72 67 65 72 20  .# pour charger
00000060   6C 65 20 73 63 72 69 70 74 20 70 75 69 73 20 73  le script puis s
00000070   61 69 73 69 72 0A 23 20 70 61 72 20 65 78 65 6D  aisir.# par exem
00000080   70 6C 65 20 73 6F 6D 6D 65 28 35 29 20 64 61 6E  ple somme(5) dan
00000090   73 20 6C 61 20 63 6F 6E 73 6F 6C 65 0A 23 20 28  s la console.# (
000000A0   70 61 72 74 69 65 20 73 75 72 20 66 6F 6E 64 20  partie sur fond
000000B0   62 6C 61 6E 63 20 C3 A0 20 64 72 6F 69 74 65 29  blanc Ã  droite)
000000C0   0A 0A 64 65 66 20 73 6F 6D 6D 65 28 6E 29 3A 0A  ..def somme(n):.
000000D0   20 20 20 20 53 20 3D 20 30 0A 20 20 20 20 66 6F      S = 0.    fo
000000E0   72 20 69 20 69 6E 20 72 61 6E 67 65 28 31 2C 20  r i in range(1,
000000F0   6E 20 2B 20 31 29 3A 0A 20 20 20 20 20 20 20 20  n + 1):.
00000100   53 20 3D 20 53 20 2B 20 69 2A 2A 32 0A 20 20 20  S = S + i**2.
00000110   20 72 65 74 75 72 6E 20 53 0A                     return S.
```

   a. Écrivez les huit bits de l'octet qui sert à coder le caractère "espace".
   b. L'éditeur hexadécimal affiche le codage du fichier par des lignes contenant chacune seize octets. Il y a correspondance entre les lignes de codage et les lignes de texte affiché à droite. Écrivez la ligne d'octets, en hexadécimal, qui correspond au texte " blanc à droite) "
   c. Comment est codé le caractère 'à' dans le fichier ?

3. Jeanne a remarqué qu'en bas de la fenêtre de son environnement de développement Python, il est écrit ISO8859-9.

Voici la table d'encodage des caractères ISO 8859-9 :

« Not-Breaking Space ». C'est un caractère d'espace sans retour à la ligne.

ISO/CEI 8859-9:1999															
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	*Inutilisé*															
1x																
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8x	*Inutilisé*															
9x																
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ğ	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	İ	Ş	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ğ	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ı	ş	ÿ

En utilisant la table, justifiez l'affichage « Ã » visible dans l'environnement de développement.

4. Le fichier Python de Jeanne a-t-il été encodé selon la table d'encodage ISO8859-9 ?
5. Selon quel encodage de caractères est le fichier Python qui a été téléchargé ?
6. Vous avez trouvé sur Internet la documentation suivante :

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

texte = input("Voilà votre entrée texte : tapez votre texte :")
print(texte)
input("Pause. Appuyer sur ENTREE")
```

Ca n'a l'air de rien comme ça, ça date de vos chapitres 1 et 2 sur Python. Mais, en réalité, c'est plus complexe qu'on ne le pense.

La première ligne n'est pas obligatoire : `#!/usr/bin/env python` : on précise en Linux le chemin de l'interpréteur à utiliser.

La deuxième ligne `# -*- coding: utf-8 -*-` précise à l'interpréteur Python que le créateur du programme a utilisé l'encodage Utf-8 pour enregistrer les lignes de code. Si vous ne précisez pas cette information, Python va travailler par défaut en Utf-8. C'est pour cela que je disais qu'elle n'est pas nécessaire ici : on encode en Uft-8 via Notepad++ si on pense à changer l'encodage (qui de base est ANSI). C'est encore moins nécessaire avec IDLE qui encode de base en UTF-8. Alors pourquoi la mettre ? Après tout, cela ne sert à rien. Et bien, cela peut servir si votre utilisateur utilise un interpréteur Python ancien, qui ne travaille pas de base avec Utf-8 : il lui ordonne de changer son tableau de décodage. Ou pour l'avenir : un utilisateur futur de Python 4 va peut-être avoir un autre système d'encodage que l'Utf-8.

En conclusion, soyez précis et prenez l'habitude de toujours placer sur l'**une des deux premières lignes** l'encodage utilisé pour enregistrer votre fichier de code.

Que conseillez-vous à Jeanne pour corriger son problème ?