1 Code ASCII

1.1 Charset et code ASCII

Méthode *Principes du codage de caractères*

- Un texte est une suite de caractères visibles (lettres majuscules ou minuscules, chiffres, symboles mathématiques, symboles de ponctuation) ou invisibles (symboles de mise en page : retours charriots ...).
  Pour coder un texte, on code séparément tous ses caractères
- Après avoir défini les caractères qu'on souhaite coder, on fait d'abord correspondre à chaque caractère un unique code : c'est le **jeu de caractères** ou **charset** ou **table de codage** ou **code point**.
  Ensuite on définit une représentation binaire de tous les codes numériques du **charset**, c'est l'**encoding**.

Définition 1

Le charset ASCII (American Standard Code for Information Interchange) fut défini en 1963 et s'imposa rapidement. Il permet de coder 128 caractères sur 7 bits ($2^7 = 128$). Néanmoins, les ordinateurs manipulant des mots de 8 bits ,on a encodé les caractères ASCII sur un octet en laissant le huitième bit pour des contrôles de parité (contrôle d'erreur). L'identification entre un caractère et un octet permet de plus une gestion économe de la mémoire.
Jeu de caractères du code ASCII :

- 95 caractères : les 26 lettres minuscules, les 26 lettres majuscules, les 10 chiffres, 32 symboles divers (dont ceux de ponctuations) et 1 signe d'espace ;
- 33 autres symboles de mise en page non affichables (les numéros 0 à 31 et 127), par exemple, le retour chariot et le saut de page.

Le jeu de caractères ASCII est limité et ne permet pas d'encoder les caractères accentués .

1.2 Exercices

Exercice 1

La taille d'un fichier se mesure en octets (1 octet = 8 bits), dont les multiples sont préfixés comme pour le système métrique par `kilo` (pour 10^3 octets), mega (10^6 octets), `giga` (10^9 octets), `tera` (10^{12} octets) ...
Traditionnellement les systèmes d'exploitation comme Windows utilisent ces préfixes pour représenter les multiples binaires les plus proches (`kilo` pour $2^{10} = 1024$ octets), mega pour $2^{20} = 1048576$ octets ,), `giga` pour $2^{30} = 1073741824$ octets ...). Il est préconisé d'utiliser plutôt les préfixes `kibi`, `mébi`, `gibi` pour cette sémantique.

1. Dans un éditeur de texte, taper le texte « Aujoud'hui, il pleut » puis enregistrer le fichier au format texte brut avec l'extension .txt

Accéder à la taille de ce fichier en octet avec le bouton droit de la souris dans l'onglet Propriétés.
Que remarque-t-on ?

2. Créer un fichier Word ou LibreOffice contenant le même texte. Comparer sa taille avec celle du fichier précédent. Comment expliquer la différence ?

Exercice 2

On utilisera la table ASCII qu'on trouvera à l'adresse :
`http://fr.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange`.
On pourra vérifier ses résultats avec le convertisseur qu'on trouvera à l'adresse :
`http://nickciske.com/tools/binary.php`

1. A l'aide de la table ASCII, coder en binaire les phrases suivantes : « Je pense, donc je suis. » et « Cet exercice est passionnant ».

2. Voici maintenant une exclamation codée en binaire : `01000010 01110010 01100001 01110110 01101111`
Retrouver cette exclamation !

3. Procéder de même avec la suite de bits :
`0100001100100111011001010111001101110100001000000110011001100001011000110110100101101100011`

4. Peut-on coder en binaire la phrase « Un âne est-il passé par là ? » à l'aide de la table ASCII ? (Justifier la réponse)

Exercice 3 *un peu de Python*

En Python la primitive `ord` retourne le code numérique d'un caractère pour le **charset** par défaut qui est l'Unicode pour Python 3.2.La primitive `chr` est la réciproque de la précédente et retourne le caractère correspondant à un code (en Unicode).

1. Tester les commandes suivantes :

```
1 >>> ord('A'); chr(65)
2 >>> for i in range(33,127): #caractères Unicode et ASCII égaux entre 33
      et 127
3         print(chr(i))
```

conversions binaire décimale

2. Déterminer les plages d'ordinaux retournées par `ord()` pour les caractères alphabétiques latins minuscules puis majuscules.

3. Ecrire une fonction `rot13(chaine)` qui applique à une chaîne de caractères un codage `rot13` en décalant de 13 modulo 26 l'ordinal du caractère s'il s'agit d'un caractère alphabétique latin minuscule (['a'-'z']) ou majuscule (['A'-'Z']).
Tester cette fonction sur une chaîne sans accents (en Anglais par exemple).
Faut-il encore beaucoup travailler pour écrire une fonction de décodage d'un texte codé avec `rot13` ?

2 Extensions du code ASCII

2.1 Les jeux de caractères ISO 8859

Le code ASCII était à l'origine conçu pour représenter des textes en Anglais mais il n'est pas adapté pour les représentation de textes dans d'autres langues comme le Français qui utilisent des symboles accentués, des cédilles ....
En conservant l'identification entre un caractère et un octet, on a utilisé le huitième bit laissé libre par le code ASCII pour rajouter de nouveaux caractères.
Le jeu de caractères **ISO-8859-1** appelée aussi **Latin-1** ou Europe occidentale fait partie de la famille des charset **ISO- 8859** (16 normes de codage), il permet de coder sur 8 bits (soit $2^8 = 256$ possibilités) presque tous les caractères des langues européennes occidentales. La norme **ISO-8859-1** permet de coder 191 caractères de l'alphabet latin mais omet quelques caractères (comme la ligature 'oe' en Français).
La norme **ISO 8859-15** complète la norme **ISO -859-1** en rajoutant le symbole €et il existe des normes **ISO-8859-2** pour les langues d'Europe de l'Est, **ISO-8859-7** pour le grec moderne,**ISO-8859-8** pour l'hébreu ....
La norme **Windows-1252** (ou `ANSI`) est une variante de **l'ISO-8859-1** qui utilise de caractères imprimables, plutôt que des caractères de contrôle, dans les codes 128 à 159.

Avec l'expansion de l'informatique hors du monde anglo-saxon, la nécessaire extension du code ASCII a conduit à une multiplicité de normes ce qui pose des problèmes de codage / décodage (voir ci-dessous la comparaison des tables **ISO-8859-15** et **Windows CP1252 (ANSI)**).
On pourra consulter les différentes tables ISO 8859 aux adresses suivantes :

`http://fr.wikipedia.org/wiki/ISO_8859`

`http://fr.wikipedia.org/wiki/ISO_8859-1`

`http://fr.wikipedia.org/wiki/ISO_8859-15`

`http://fr.wikipedia.org/wiki/Windows-1252`

ISO/CEI 8859-15

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	*non utilisé*															
1x																
2x		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
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	*non utilisé*															
9x																
Ax		¡	¢	£	€	¥	Š	§	š	©	ª	«	¬		®	¯
Bx	°	±	²	³	Ž	µ	¶	·	ž	¹	º	»	Œ	œ	Ÿ	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Windows-1252 (CP1252)																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
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	{	|	}	~	DEL
8x	€		‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ		Ž	
9x		‘	’	“	”	•	–	—	˜	™	š	›	œ		ž	Ÿ
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬		®	¯
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	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Exercice 4 *compatibilité des différentes normes*

Se connecter à l'adresse `https://wiki.inria.fr/sciencinfolycee/Convertisseur_texte/binaire/hexa_en_ligne`

1. A l'aide du logiciel, retrouver le texte contenu dans le code :

```
01000001 01110101 00100000 01100100 11101001 01100010 01110101 01110100
00100000 01101001 01101100 00100000 01111001 00100000 01100001 01110110
01100001 01101001 01110100 00100000 01101100 01100101 00100000 01100011
01101111 01100100 01100101 00100000 01000001 01010011 01000011 01001001
01001001
```

2. Ce logiciel est-il compatible avec la norme **ISO 8859-1**, **ISO 8859-15** ou **Windows 1252** ? (justifier la réponse).

3. Trouver une astuce pour savoir laquelle des trois est utilisée !

2.2 Problèmes d'encodage/décodage

Exercice 5 *dans une page web*

1. Ouvrir le fichier `encodageweb.html` avec un navigateur web comme Firefox.

2. La page s'affiche-t-elle correctement ?

   Dans la barre d'outils on peut voir à « `Affichage` », « `Encodage des caractères` » que c'est Windows-1252.

   Modifier l'encodage en UTF-8 à partir du menu affichage de Firefox. Que se passe-t-il ?

3. Tester la lecture de la page avec les autres encodages proposés dans le menu Affichage du navigateur : Grec (ISO-8859-2), Occidental avec le symbole euro (ISO-8859-15) ...

```
1  <!DOCTYPE html>
2  <head>
3  <title>Encodage</title>
4  <meta http-equiv="Content-Type" content="text/html; charset=Windows-1252">
5  </head>
6  <body>
7  <p>
8  blabla
9  </p>
10 </body>
```

encodageweb.html

Pour plus d'informations sur le problème de l'encodage des pages web on pourra consulter cette page : `http://www.alsacreations.com/astuce/lire/69-declarer-encodage-des-caracteres.html`.

Exercice 6 *en Python*

En Python, les textes sont des chaînes de caractères de type `str` qui sont des séquences de caractères du charset Unicode. La méthode encode permet de convertir une chaîne dans le charset passé en argument. L'objet retourné n'est plus de type `str` mais de type `bytes`. Un objet de type bytes est une séquence d'octets où chaque octet est représenté par le caractère ASCII correspondant ou par son codage en hexadécimal dans le charset spécifié (par exemple `'\xa4'` pour 164 soit le caractère € en ISO 8859-15).
Tester les instructions suivantes et interpréter les résultats affichés.

```
>>> chaine = 'ceci est un euro : ?'
>>> chaine.encode('cp1252')
>>> chaine.encode('iso-8859-1')
>>> chaine.encode('iso-8859-15')
```

changements de charset

Exercice 7 *problèmes d'encodage dans un email*

SMTP (*Simple Mail Transfer Protocol*) est le protocole chargé d'acheminer les courriels (ou email) d'un expéditeur vers un destinataire. Les messages transportés sont encodés en ASCII brut (7 bits par caractère) Il existe des standards d'encodage des données de toutes sortes (textes dans des encodages non ACSII, données binaires comme des images ou des sons) selon leur type (il s'agit des types MIME pour Multipurpose Internet Mail Extension) en données ASCII grâce à un codage dit en base 64. Voir le document donné complément pour plus de détails sur le codage en base 64. Il faut savoir que ce codage est surtout intéressant pour encoder en ASCII des données binaires car 'il présente deux inconvénients : il rend illisible le message initial s'il s'agit d'un texte et il augmente la taille du message transmis de 1/3 environ. Le codage Quoted-Printable permet de remédier au problème de lisibilité de l'encodage des messages textes.
Python propose deux modules `email` et `smtplib` qui de créer des emails et de les envoyer. Le module `email` prend en charge tous les types MIME et permet d'envoyer des emails avec des pièces jointes de tout type ...

1. Créer avec Geany un fichier `message.txt` contenant des caractères accentués et le symbole €.Choisir l'encodage Européen de L'Ouest `ISO-8859-1`

2. Créer un script Python `emailfromtext.py` en complétant le code Python ci-dessous avec l'adresse de son serveur `smtp`, son adresse mail pour l'expéditeur et celle d'un camarade pour le destinataire.
   Exécuter le script et vérifier auprès du destinataire qu'il a bien reçu le mail.

3. Modifier le fichier `message.txt` en choisissant l'encodage `Windowscp1252` et en y insérant le mot oeuvre (avec la ligature obtenue avec `alt o`).
   Exécuter le script `emailfromtext.py`. Que constate-t-on dans le message réceptionné ?
   Changer la spécification de l'encodage à la ligne 13 en remplaçant 'Latin_1' par 'cp1252'.
   Exécuter à nouveau le script. Le message a-t-il été correctement transmis ?

```
1  #-*-coding:Utf-8-*
2  """
3  Envoi d'un email de type texte
4  Pour la doc du module email :
5   http://docs.python.org/library/email.html#module-email
6  """
7  import smtplib
8  from email.mime.text import MIMEText
9
10 #ouverture du fichier contenant le corps du message
```

```
#on précise l'encodage du fichier source si ce n'est pas de l'UTF-8
#en interne Python encodera en Unicode UTF-8
myfile = open('message.txt','r',encoding='latin_1')
#creation de l'objet courriel de type MIME /text
#on précise le _charset UTF-8 afin de pouvoir envoyer un message avec
#des carctères accentués, sinon le _charset par défaut est us-ascii
#et les caractères accentués ne sont pas pris en charge
myemail = MIMEText(myfile.read(),_charset='utf-8')
#on ferme proprement le fichier contenant le message
myfile.close()
#on remplit l'en-tete du courriel
myemail['Subject'] = 'Un email contenant du texte'
myemail['From'] = 'expediteur@messagerie.fr'
myemail['To'] = 'destinataire@messagerie.fr'
#on se connecte à notre serveur smtp (exemple avec celui de yahoo en
    connexion sécurisée par le protocole TLS
smtpserver = smtplib.SMTP('smtp.mail.yahoo.com',587)
smtpserver.set_debuglevel(True)
#activation du protocole de chaiffrement TLS
smtpserver.starttls()
#il faut s'authentifier
smtpserver.login('login','password')
#on envoie le message
smtpserver.send_message(myemail)
#on ferme proprement la connexion avec le serveur smtp
smtpserver.quit()
```

3 L'Unicode

3.1 Présentation

Avec la globalisation des échanges et la mondialisation d'internet, la diversité des normes d'encodage, l'absence de codage pour certains caractères ont posé problème. Depuis 1991, le consortium Unicode composé d'informaticiens, de chercheurs, de linguistes et de personnalités représentant les États et les entreprises, construit un **jeu de caractères universel** dont le but est d'associer un nom et un numéro uniques à tous les caractères des langues connues.
Unicode est un **charset**, le nom et l'ordinal (son numéro) seront les mêmes pour toutes les plate-forme informatiques ou logiciel utilisés. L'encodage d'Unicode est prévu sur 32 bits mais il existe plusieurs variantes d'**encoding** : l'**UTF-8** est l'encodage le plus répandu et tend à s'imposer comme une norme (navigateurs internet, langages de programmation ...). Il existe aussi les encodages UTF-16 et UTF-32 ; contrairement aux encodages ISO 8859, ce sont des encodages sur plusieurs octets.

Exercice 8 *comparaison de tables de caractères*

Pour comparer les tables de différents jeux de caractères il existe de petits utilitaires :

- Sous Windows aller dans « Démarrer », « Exécuter », taper « charmap ». Cocher « Affichage avancé », sélectionner « Windows Occidental » (c'est le charset ANSI ou cp1252) dans « Jeu de caractères ».
  Regarder le nombre de caractères proposés, puis sélectionner « Unicode » et comparer.

- Sous Linux on utilisera « gucharmap » pour obtenir la table des caractères Unicode avec leur encodage en UTF-8, et pour OSX (Apple) il faut chercher la « palette de caractères ».
- On peut connaître le plus grand codage entier possible d'un caractère Unicode disponible sur une plateforme avec le module `sys` de Python :

```
>>> import sys
>>> sys.maxunicode #plus grand codage entier d'un caractère Unicode
1114111
>>> chr(sys.maxunicode) #représentation formelle préfixe \U puis
    notation hexadécimale du code entier
'\U0010ffff'
>>> hex(sys.maxunicode)
'0x10ffff'
>>> for i in range(1050003,1050011): #à la recherche du plus grand
    caractère codé
       print(chr(sys.maxunicode-i),end=',')
#commande à tester en console
>>> sys.maxunicode - 1050005
64106
```

On peut constater que 64 106 caractères alphabétiques sont déjà codés en Unicode sur cette machine et qu'on peut aller jusqu'à 1 114 111 . On peut encore inventer (ou découvrir) de nouveaux alphabets !!!

3.2 Exercices

Exercice 9 *encodage en UTF-8*

L'encodage UTF-8 utilise 1, 2, 3 ou 4 octets en respectant certaines règles :

- Un texte en ASCII de base est codé de manière identique en UTF-8. On utilise un octet commençant par un bit 0 à gauche (bit de poids fort).
- Les octets ne sont pas remplis entièrement. Les bits de poids fort du premier octet forment une suite de 1 indiquant le nombre d'octets utilisés pour coder le caractère. Les octets suivants commencent tous par le bloc binaire 10.

Par exemple, dans la norme ISO-8859-1 le « é » est codé `1110 1001`, en UTF-8 on le code sur deux octets. Les bits imposés sont en gras, le code du « é » est écrit en commençant par la droite et l'octet de gauche est rempli par des zéros (en italique). On obtient : **110***000*11 **10**101001. Le codage ISO s'inscrit bien dans le codage UTF-8.

1. Déterminer l'ordinal Unicode du symbole €avec la primitive `ord` de Python.
2. Convertir cette valeur en binaire.
3. Combien d'octets doit-on utiliser en UTF-8 pour coder ce nombre convenablement (les moitiés d'octet sont interdites) ?
4. Donner le codage UTF-8 correspondant.
5. Faire de même pour les caractères « j » et « à ».

Exercice 10 *Encodage et décodage en Python*

En Python le jeu de caractères par défaut est l'Unicode. La primitive `ord` retourne l'ordinal d'un caractère en Unicode et `chr` est sa fonction réciproque. L'interpréteur Python décode le texte qui lui est transmis à travers un **codec**. A partir de Python3, le codec par défaut est le codec Utf-8 mais on peut écrire un script dans un autre encodage qu'on précisera à la première ligne d'un programme (pour assurer la portabilité du programme) avec le pseudo-commentaire `#-*- coding: latin1 -*-` par exemple. (`#-*- coding: utf-8 -*-` conseillé). Si le script est en `utf-8` cette ligne est inutile pour une exécution sous Python3 mais pas sous Python2 dont le codec par défaut est `ascii`.

1. Pour obtenir les informations sur le système et en particulier l'encodage, on peut utiliser le module `sys`.

```
>>> import sys #voir help(sys) pour la documentation
>>> sys.getdefaultencoding() #encodage par défaut de Python
>>> sys.getfilesystemencoding() #encodage du système d'exploitation
>>> sys.stdin.encoding #encodage du flux d'entrée standard en mode
    console (saisi clavier)
```

module sys

2. Ecrire le programme suivant en Python3.2 sans pseudo-commentaire d'encodage puis exécuter ce programme avec Python3.2 puis Python2.7. Que remarquez-vous ?

   Déterminer l'encodage par défaut de Python2.7 puis insérer un pseudo-commentaire d'encodage pour que le programme soit lisible par l'interpréteur de Python2.7. Dans les codes suivants, remplacer le symbole '?' par '€'.

```
chaine = 'à un euro ? près'
print(chaine)
```

changements de charset

3. Tester les instructions suivantes pour examiner les différences d'encodage entre les encodages sur un octet (`cp1252, ISO 8859`) et un encodage multi-octet comme l'`Utf-8`.

```
>>> chaine = 'à un euro ? près'
>>> chaine_latin15 = chaine.encode('iso-8859-15') ; print(chaine_latin15
    )
>>> chaine_cp1252 = chaine.encode('cp1252') ; print(chaine_cp1252)
>>> chaine_utf8 = chaine.encode('utf-8') ; print(chaine_utf8)
```

changements de charset

Exercice 11 *écriture et lecture dans des fichiers*

En Python, l'accès à un fichier se fait par l'intermédiaire d'un objet-fichier créé à l'aide de la primitive `open(nom,mod`

Fonctions	Rôle
`f = open('nom_fichier.txt','w')`	accès en écriture avec création d'un nouveau fichier
`f = open('nom_fichier.txt','a')`	accès en écriture à un fichier existant en mode ajout
`f = open('nom_fichier.txt','r')`	accès en écriture à un fichier existant en mode lecture
`f.write('texte')`	ajout de 'texte' dans le fichier
`f.writelines(liste)`	ajout d'une liste de lignes dans un fichier
`f.read()`	lecture de tout le fichier
`f.readlines()`	lecture de toutes les lignes stockées dans une liste
`f.close()`	fermeture du fichier

Les problèmes d'encodage/décodage ne se posent pas vraiment dans la manipulation des chaînes au sein d'un programme (sauf si l'encodage choisi ne permet pas d'encoder certains caractères). En revanche, pour les chaînes utilisées en entrées/sorties, il est important de préciser les encodages utilisés.

1. Tester le programme `encodage1.py` avec la chaîne "Un euro est 1 € est une vérité".
   Que remarque-t-on ? Interpréter ce résultat et corriger le programme.

2. Remplacer la valeur de l'argument `encoding` à la ligne 5 par 'utf-8'.
   Que se passe-t-il ? Interpréter ce résultat et corriger le programme.

3. Remplacer la valeur de l'argument `encoding` à la ligne 2 par 'utf-8' et l'argument `encoding` à la ligne 5 par 'cp1252' .
   Que se passe-t-il ? Interpréter ce résultat.

4. Pour insérer un caractère dont on connaît l'ordinal Unicode (par exemple 666) dans une chaîne, on convertit d'abord l'ordinal en hexadécimal (`hex(666)` retourne `'0x29a'`) puis on insère le caractère ainsi `\u029a`, c'est-à-dire un symbole d'échappement suivi de l'ordinal écrit en hexadécimal avec quatre chiffres.

   a. La ligature [1] est le caractère d'ordinal Unicode 339, elle ne figure pas dans jeu de caractères ISO-8859-1.
   Tester les instructions suivantes pour comparer deux écritures du mot 'oeuvre'.

```
>>> a = 'oeuvre' ; len(a)
>>> chr(339) ; hex(339) ; '\u0153' ; ord('\u0153')
>>> b = '\u0153uvre' ; len(b)
```

   b. Utiliser le même procédé pour écrire "c'est Noël" sans utiliser de combinaisons de touches pour obtenir le caractère 'ë'.

```
#-*- coding: Utf-8 -*-
entree = input('Entrez une chaîne de caractères : \n')
f = open('test.txt','w',encoding='cp1252')
f.write(entree)
f.close()
g = open('test.txt','r',encoding='Latin-1')
sortie = g.read()
g.close()
print(sortie)
```

encodage1.py

1. Ce document étant encodé en Latin1, je ne peux pas l'afficher ...

Table des matières