Manipulation de fichiers

Jean-Pierre Becirspahic
Lycée Louis-Le-Grand

Le module os

Les instructions permettant à l'interprète de dialoguer avec le système d'exploitation font partie du module os :

```
>>> import os
```

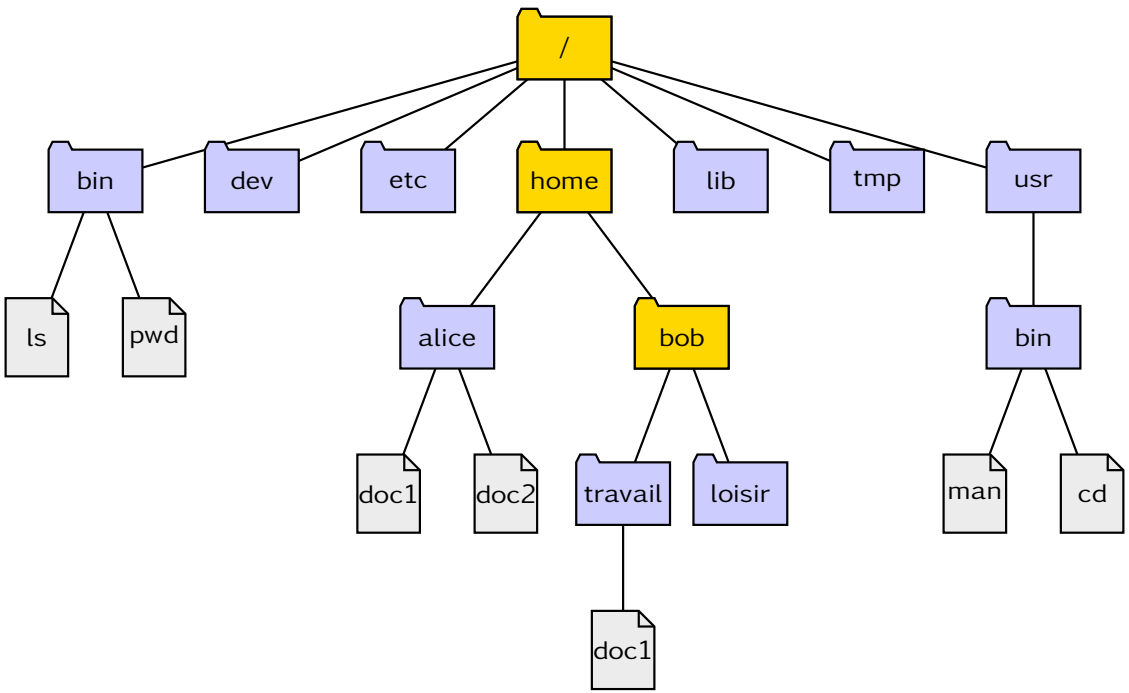
Le module os

Les instructions permettant à l'interprète de dialoguer avec le système d'exploitation font partie du module os :

```
>>> import os
```

La fonction `getcwd()` indique le répertoire courant :

```
>>> os.getcwd()
'/home/bob'
```



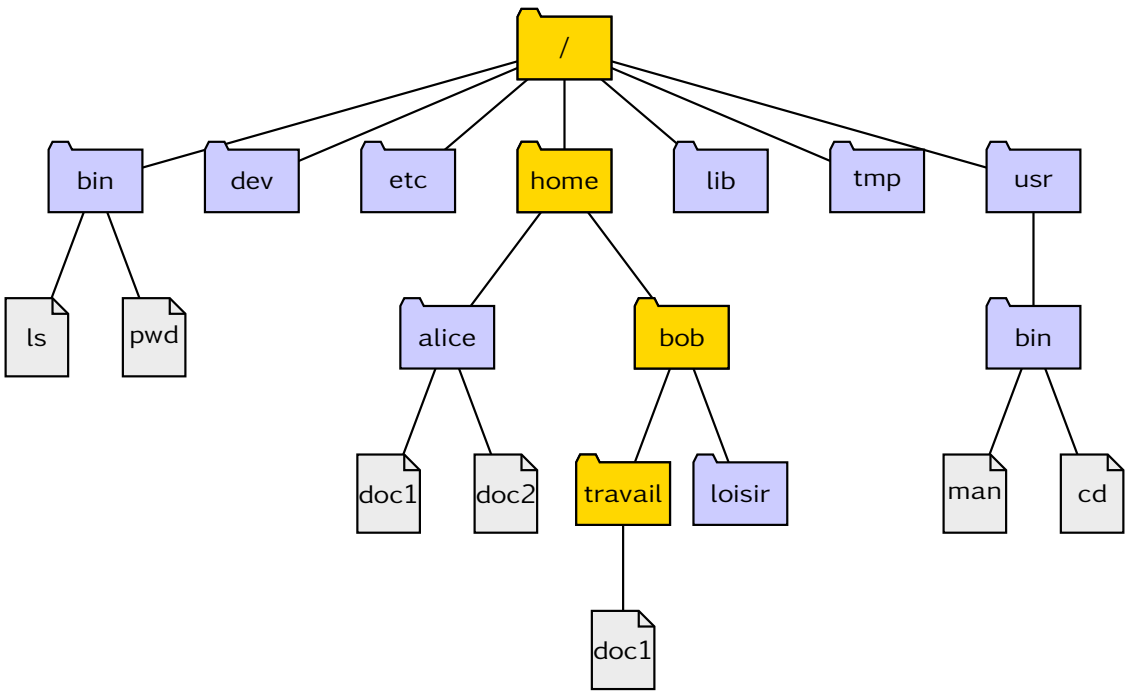
Le module os

Les instructions permettant à l'interprète de dialoguer avec le système d'exploitation font partie du module os :

```
>>> import os
```

La fonction chdir permet de changer le répertoire courant :

```
>>> os.chdir('/home/bob/travail')
```



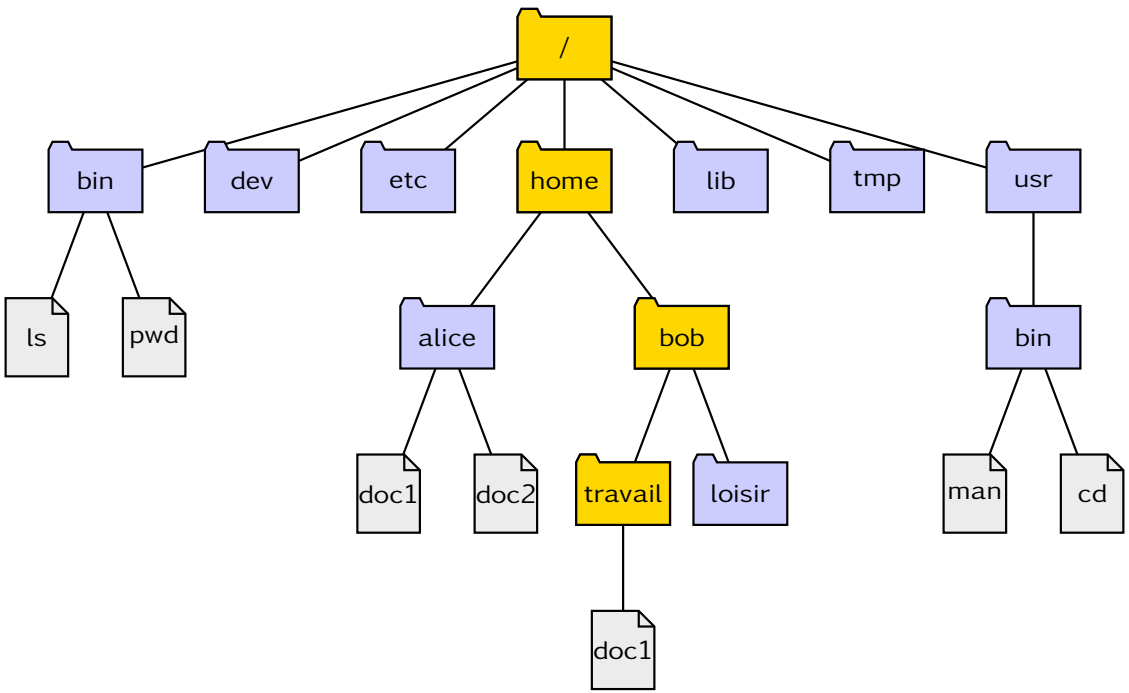
Le module os

Les instructions permettant à l'interprète de dialoguer avec le système d'exploitation font partie du module os :

```
>>> import os
```

La fonction `listdir` liste le contenu d'un répertoire :

```
>>> os.listdir('.')
['doc1']
```



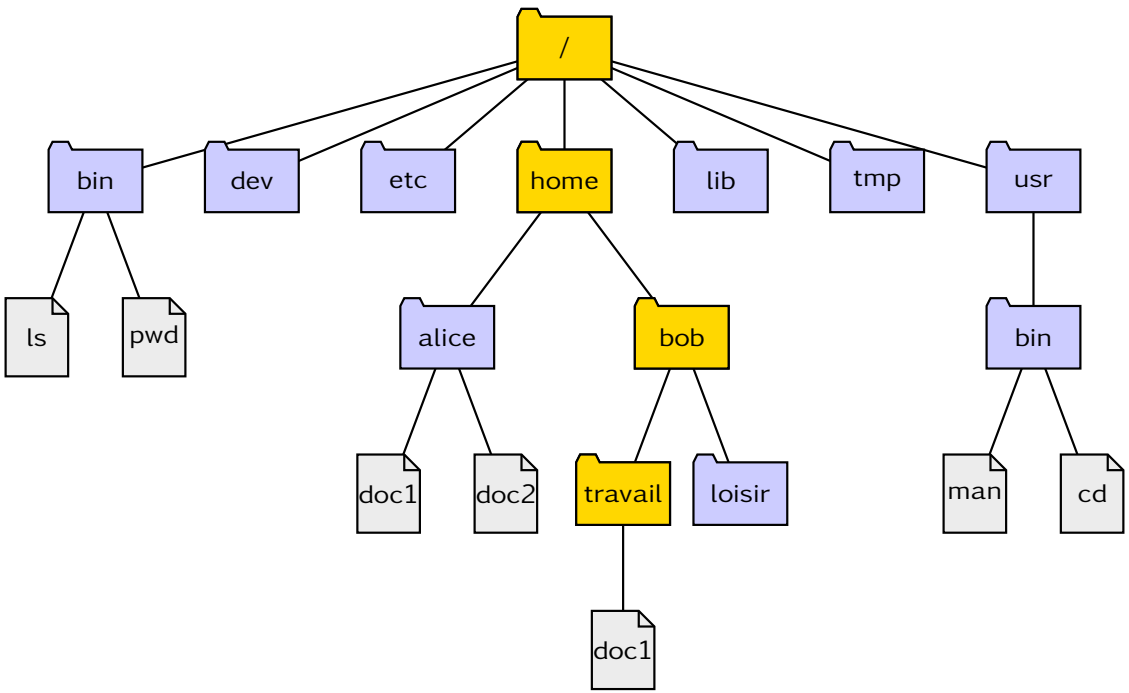
Le module os

Les instructions permettant à l'interprète de dialoguer avec le système d'exploitation font partie du module os :

```
>>> import os
```

La fonction `listdir` liste le contenu d'un répertoire :

```
>>> os.listdir('/home/bob/travail')
['doc1']
```



Lecture d'un fichier texte

La fonction **open** propose trois modes d'ouverture d'un fichier :

- en lecture (`'r'`) ;
- en écriture (`'w'`) ;
- en ajout (`'a'`).

Lecture d'un fichier texte

La fonction **open** propose trois modes d'ouverture d'un fichier :

- en lecture (`'r'`) ;
- en écriture (`'w'`) ;
- en ajout (`'a'`).

Pour ouvrir en lecture le fichier `exemple.txt` du répertoire courant :

```
>>> comptine = open('exemple.txt', 'r')
```

Lecture d'un fichier texte

La fonction **open** propose trois modes d'ouverture d'un fichier :

- en lecture (`'r'`) ;
- en écriture (`'w'`) ;
- en ajout (`'a'`).

Pour ouvrir en lecture le fichier `exemple.txt` du répertoire courant :

```
>>> comptine = open('exemple.txt', 'r')
```

Nous venons de créer un objet `comptine` faisant référence au fichier `exemple.txt` :

```
>>> comptine
<_io.TextIOWrapper name='exemple.txt' mode='r' encoding='UTF-8'>
```

Cet objet est un flux : les caractères sont lisibles uniquement les uns après les autres, sans possibilité de retour en arrière ni de saut en avant.

Lecture d'un fichier texte

La fonction **open** propose trois modes d'ouverture d'un fichier :

- en lecture (`'r'`) ;
- en écriture (`'w'`) ;
- en ajout (`'a'`).

Pour ouvrir en lecture le fichier `exemple.txt` du répertoire courant :

```
>>> comptine = open('exemple.txt', 'r')
```

Pour lire le fichier dans son entier : la méthode `read()`.

```
>>> comptine.read()
'Am, stram, gram,\nPic et pic et colégram,\nBour et bour et
 ratatam,\nAm, stram, gram.'
```

Lecture d'un fichier texte

La fonction **open** propose trois modes d'ouverture d'un fichier :

- en lecture (`'r'`);
- en écriture (`'w'`);
- en ajout (`'a'`).

Pour ouvrir en lecture le fichier `exemple.txt` du répertoire courant :

```
>>> comptine = open('exemple.txt', 'r')
```

Pour lire le fichier par groupe de 10 caractères :

```
>>> lst = []
>>> while True:
...     txt = comptine.read(10)
...     if len(txt) == 0:
...         break
...     lst.append(txt)
>>> lst
['Am, stram,', ' gram,\nPic', ' et pic et', ' colégram,', '\nBour
et b', 'our et rat', 'atam,\nAm, ', 'stram, gra', 'm.']
```

Lecture d'un fichier texte

La fonction **open** propose trois modes d'ouverture d'un fichier :

- en lecture (`'r'`) ;
- en écriture (`'w'`) ;
- en ajout (`'a'`).

Pour ouvrir en lecture le fichier `exemple.txt` du répertoire courant :

```
>>> comptine = open('exemple.txt', 'r')
```

Pour lire le fichier ligne par ligne : la méthode `readlines(n)`.

```
>>> comptine.readlines()
['Am, stram, gram,\n', 'Pic et pic et colégram,\n',
 'Bour et bour et ratatam,\n', 'Am, stram, gram.']
```

Lecture d'un fichier texte

La fonction **open** propose trois modes d'ouverture d'un fichier :

- en lecture (`'r'`) ;
- en écriture (`'w'`) ;
- en ajout (`'a'`).

Pour ouvrir en lecture le fichier `exemple.txt` du répertoire courant :

```
>>> comptine = open('exemple.txt', 'r')
```

Lecture par énumération des lignes :

```
>>> n = 0
>>> for l in comptine:
...     n += 1
...     print('{} :'.format(n), l, end='')
1 : Am, stram, gram
2 : Pic et pic et colégram,
3 : Bour et bour et ratatam,
4 : Am, stram, gram.
```

Lecture d'un fichier texte

La fonction **open** propose trois modes d'ouverture d'un fichier :

- en lecture (`'r'`) ;
- en écriture (`'w'`) ;
- en ajout (`'a'`).

Pour ouvrir en lecture le fichier `exemple.txt` du répertoire courant :

```
>>> comptine = open('exemple.txt', 'r')
```

Pour fermer un fichier : la méthode `close()`.

```
>>> comptine.close()
```

Fichiers CSV

Comma-Separated Value

On considère le fichier `planetes.txt` contenant le texte suivant :

```
Mercure, 2439, 3.7, 88
Vénus, 6052, 8.9, 225
Terre, 6378, 9.8, 365
Mars, 3396, 3.7, 687
```

Fichiers CSV

Comma-Separated Value

On considère le fichier `planetes.txt` contenant le texte suivant :

```
Mercure, 2439, 3.7, 88
Vénus, 6052, 8.9, 225
Terre, 6378, 9.8, 365
Mars, 3396, 3.7, 687
```

On ouvre le fichier et on découpe le texte en lignes :

```
>>> planetes = open('planetes.txt', 'r')
>>> lignes = planetes.readlines()
>>> planetes.close()
```

À cette étape, `lignes` est une liste de chaînes de caractères égale à :

```
['Mercure, 2439, 3.7, 88\n', 'Vénus, 6052, 8.9, 225\n', 'Terre, 6378, 9.8,
365\n', 'Mars, 3396, 3.7, 687\n']
```

Fichiers CSV

Comma-Separated Value

On considère le fichier `planetes.txt` contenant le texte suivant :

```
Mercure, 2439, 3.7, 88
Vénus, 6052, 8.9, 225
Terre, 6378, 9.8, 365
Mars, 3396, 3.7, 687
```

Chaque ligne est découpée en colonnes par la méthode `split` :

```
>>> tab = []
>>> for chn in lignes:
...     tab.append(chn.split(','))
```

À cette étape, `tab` est une liste de listes égale à :

```
[['Mercure', ' 2439', ' 3.7', ' 88\n'], ['Vénus', ' 6052', ' 8.9', ' 225\n'],
['Terre', ' 6378', ' 9.8', ' 365\n'], ['Mars', ' 3396', ' 3.7', ' 687\n']]
```

Fichiers CSV

Comma-Separated Value

On considère le fichier `planetes.txt` contenant le texte suivant :

```
Mercure, 2439, 3.7, 88
Vénus, 6052, 8.9, 225
Terre, 6378, 9.8, 365
Mars, 3396, 3.7, 687
```

On convertit les données numériques :

```
>>> for lst in tab:
...     lst[1] = int(lst[1])
...     lst[2] = float(lst[2])
...     lst[3] = int(lst[3])
```

La liste `tab` est maintenant prête à être utilisée :

```
[['Mercure', 2439, 3.7, 88], ['Vénus', 6052, 8.9, 225], ['Terre', 6378, 9.8,
365], ['Mars', 3396, 3.7, 687]]
```

Écrire dans un fichier texte

Deux modes d'ouverture possibles : le mode `'w'` (*write*) et le mode `'a'` (*append*).
Dans les deux cas, la méthode `write` permet d'enregistrer les chaînes de caractères passées en argument les unes à la suite des autres.

Écrire dans un fichier texte

Deux modes d'ouverture possibles : le mode `'w'` (*write*) et le mode `'a'` (*append*).
Dans les deux cas, la méthode `write` permet d'enregistrer les chaînes de caractères passées en argument les unes à la suite des autres.
Pour ajouter au fichier `planetes.txt` des données supplémentaires :

```
>>> planetes = open('planetes.txt', 'a')

>>> planetes.write('Jupiter, 71492, 24.8, 4335\n')
>>> planetes.write('Saturne, 60268, 10.4, 10757\n')

>>> planetes.close()
```

Écrire dans un fichier texte

Deux modes d'ouverture possibles : le mode `'w'` (*write*) et le mode `'a'` (*append*).
Dans les deux cas, la méthode `write` permet d'enregistrer les chaînes de caractères passées en argument les unes à la suite des autres.
Pour ajouter au fichier `planetes.txt` des données supplémentaires :

```
>>> planetes = open('planetes.txt', 'a')

>>> planetes.write('Jupiter, 71492, 24.8, 4335\n')
>>> planetes.write('Saturne, 60268, 10.4, 10757\n')

>>> planetes.close()
```

Le fichier `planetes.txt` contient maintenant le texte suivant :

```
Mercure, 2439, 3.7, 88
Vénus, 6052, 8.9, 225
Terre, 6378, 9.8, 365
Mars, 3396, 3.7, 687
Jupiter, 71492, 24.8, 4335
Saturne, 60268, 10.4, 10757
```

Encodage d'un fichier texte

Le jeu de caractères ASCII

Historiquement un caractère est codé sur 7 bits, ce qui donne $2^7 = 128$ caractères différents, qui constituent le jeu de caractères ASCII.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0																
1																
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Encodage d'un fichier texte

Le jeu de caractères ASCII

Historiquement un caractère est codé sur 7 bits, ce qui donne $2^7 = 128$ caractères différents, qui constituent le jeu de caractères ASCII.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0																
1																
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

L'espace a pour code ASCII $(20)_{16} = 32$; le caractère `'A'` a pour code ASCII $(41)_{16} = 65$; le caractère `'a'` a pour code ASCII $(61)_{16} = 97$.

Encodage d'un fichier texte

Le jeu de caractères ASCII

Historiquement un caractère est codé sur 7 bits, ce qui donne $2^7 = 128$ caractères différents, qui constituent le jeu de caractères ASCII.

Ce codage simple est insuffisant pour pouvoir représenter la diversité des caractères des langues autres que l'anglais, aussi un huitième bit a été utilisé pour ajouter au jeu de caractères ASCII 128 autres caractères codés entre $128 = (80)_{16}$ et $255 = (ff)_{16}$.

Encodage d'un fichier texte

Le jeu de caractères ASCII

Historiquement un caractère est codé sur 7 bits, ce qui donne $2^7 = 128$ caractères différents, qui constituent le jeu de caractères ASCII.

Ce codage simple est insuffisant pour pouvoir représenter la diversité des caractères des langues autres que l'anglais, aussi un huitième bit a été utilisé pour ajouter au jeu de caractères ASCII 128 autres caractères codés entre $128 = (80)_{16}$ et $255 = (ff)_{16}$.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
8																
9																
a		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬		®	¯
b	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
c	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
d	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
e	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
f	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Encodage d'un fichier texte

Le jeu de caractères ASCII

Historiquement un caractère est codé sur 7 bits, ce qui donne $2^7 = 128$ caractères différents, qui constituent le jeu de caractères ASCII.

Ce codage simple est insuffisant pour pouvoir représenter la diversité des caractères des langues autres que l'anglais, aussi un huitième bit a été utilisé pour ajouter au jeu de caractères ASCII 128 autres caractères codés entre $128 = (80)_{16}$ et $255 = (ff)_{16}$.

Chaque langue ayant des besoins spécifiques, ces extensions sont nombreuses et non compatibles entre elles : la norme LATIN-1 permet d'encoder les langues d'Europe occidentale, la norme LATIN-2 pour les langues d'Europe centrale, etc. Pas moins de 16 variantes existent pour le seul standard ISO 8859.
Mais les écritures idéographiques comme le chinois nécessitent plusieurs milliers de caractères et ne peuvent donc être codées sur un seul octet.

La norme Unicode

Elle attribue un identifiant numérique universel à chacun des milliers de caractères nécessaires à la transcription des différentes langues.
L'encodage le plus fréquent de l'Unicode est la norme UTF-8 ; c'est la norme utilisée par défaut par PYTHON.

La norme Unicode

Elle attribue un identifiant numérique universel à chacun des milliers de caractères nécessaires à la transcription des différentes langues.
L'encodage le plus fréquent de l'Unicode est la norme UTF-8 ; c'est la norme utilisée par défaut par PYTHON.

Le fichier `comptine.txt` ouvert au format UTF-8 :

```
>>> comptine = open('exemple.txt', 'r')
>>> print(comptine.read())
Am, stram, gram
Pic et pic et colégram,
Bour et bour et ratatam,
Am, stram, gram.
```

La norme Unicode

Elle attribue un identifiant numérique universel à chacun des milliers de caractères nécessaires à la transcription des différentes langues.
L'encodage le plus fréquent de l'Unicode est la norme UTF-8 ; c'est la norme utilisée par défaut par PYTHON.

Le fichier `comptine.txt` ouvert au format LATIN-1 :

```
>>> comptine = open('exemple.txt', 'r', encoding='latin1')
>>> print(comptine.read())
Am, stram, gram
Pic et pic et colÃ©gram,
Bour et bour et ratatam,
Am, stram, gram.
```

L'encodage choisi n'est manifestement pas le bon !

La norme Unicode

Elle attribue un identifiant numérique universel à chacun des milliers de caractères nécessaires à la transcription des différentes langues.
L'encodage le plus fréquent de l'Unicode est la norme UTF-8 ; c'est la norme utilisée par défaut par PYTHON.

La fonction **chr** retourne le caractère ASCII dont l'identifiant a été passé en paramètre mais aussi le caractère unicode associé à son identifiant :

```
>>> for i in range(945, 970):
...     print(chr(i), end=' ')
```

α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω

Fichiers image

Images en noir et blanc

Une image binaire peut être représentée par une matrice $p \times q$ dont les éléments, des 0 ou des 1 (plus exactement des booléens), indiquent la couleur du pixel : 0 pour le noir et 1 pour le blanc.

```
1111110111111111111111111110111111111111
1111100011111111111111111000111111111111
1111100001111111111111111000011111111111
1111100000111111111111110000011111111111
1111100000011111111111100000011111111111
1111100000000000000000000000011111111111
1111100000000000111111111000011001111111
1111100000000001111111111100011100111111
1111100000000011111111111110011110011111
1111000000000111111111111111001111101111
1111000000000111111111111111101111100101
1111000000001111111111000111011110001101
1110000000001111111111000011100111000100
1100000000001111111111000011100110000100
1000000000001111111111000011100111001110
1000000000000111111111000011100111111110
0000000000000111111111111111001010111110
1000000000000011111111111111101100000100
1100000111000111111111111100100000000010
1110001111111000111111111100111000000100
1100001110001100000000001110000000000000
1000001110001111100001111111100000000000
1111001111110011111111111111111111101011
1111100111111100011111111111111111001001
1111100111111100000111111111111111000111
1111111001111111000000000000000010111111
1111111100111111000111000000000010011111
1111111110011111011111101110010011111111
1111111100111111001111111001001111111111
1111111111000111100000000000001111111111
1111111111111000001111100000011111111111
1111111111111100000000000001111111111111
```

Fichiers image

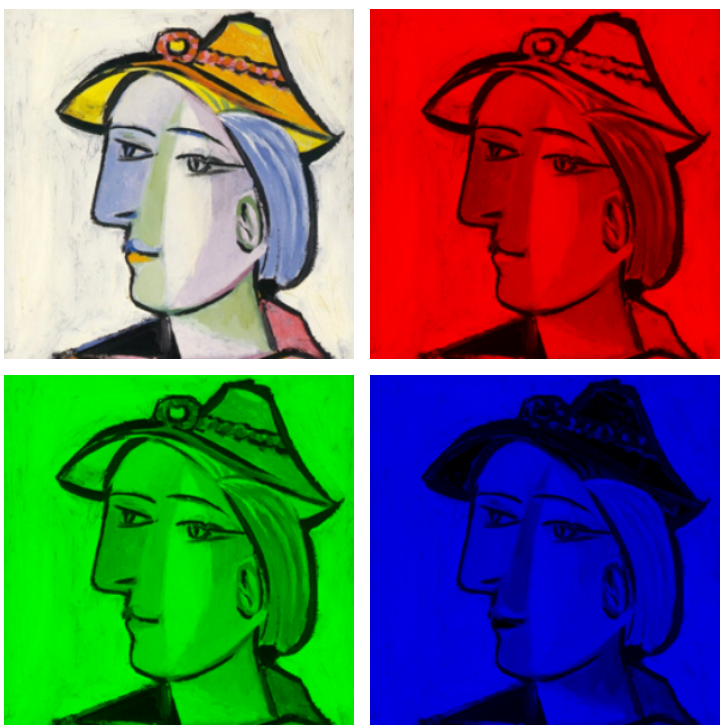
Images en gris

Une image en gris est aussi représentée par une matrice, mais chaque élément détermine la luminance du pixel correspondant (en général un entier non signé codé sur 8 bits).
Voici par exemple huit niveaux de gris différents :

Fichiers image

Images couleurs

Une image en couleur peut être représentée par trois matrices, chacune déterminant la quantité respective de rouge, de vert et de bleu qui constitue l'image (c'est le modèle RGB).
Les éléments de ces matrices sont des nombres entiers compris entre 0 et 255 (des entiers non signés sur 8 bits) qui déterminent la luminance de la couleur de la matrice pour le pixel correspondant.



Fichiers image

Images couleurs

Une image en couleur peut être représentée par trois matrices, chacune déterminant la quantité respective de rouge, de vert et de bleu qui constitue l'image (c'est le modèle RGB).
Les éléments de ces matrices sont des nombres entiers compris entre 0 et 255 (des entiers non signés sur 8 bits) qui déterminent la luminance de la couleur de la matrice pour le pixel correspondant.