GESTION DE CARACTERES :

CONVERTIR UN CARACTERE VERS/DEPUIS UN NOMBRE :

Au sein d'un langage de programmation les caractères sont représentés (si on simplifie les choses) sous forme de nombres selon ce qu'on appelle le code ASCII (American Standard Code for Information Interchange). Le tableau ci-dessous indique cette correspondance, ce qui nous intéresse étant les colonnes "Dec" (code décimal) et "Char" (caractère associé).

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

La plupart des 31 premiers caractères sont ce qu'on appelle des "caractères de contrôle", on ne s'y intéressera pas. Remarquez tout le même qu'ils contiennent la tabulation (Dec = 9) ou les retours à la ligne (Dec = 10 par exemple).

Ce qui est intéressant c'est qu'il est possible de convertir un caractère vers son code ASCII, et inversement.

Caractère -> code

```
caractere = "U"
code = ord(caractere)
print(code)
```

```
85
```

Code -> Caractère

```
code = 111
caractere = chr(code)
print(caractere)
```

```
o
```

Encodage réel

En Python, les chaînes de caractères sont en réalité codées en UTF-8, un code (ou encodage) plus général que l'ASCII et qui permet de représenter plus de caractères, par exemples les caractères chinois. En UTF-8, Les caractères d'indices inférieurs à 128 sont exactement les mêmes que en ASCII : le tableau ci-dessus fonctionne donc.

Afficher la liste des caractères

Ci-dessous, vous trouverez un petit code permettant d'afficher la liste des caractères existants avec leur code associé. Les caractères d'indice inférieur à 32 sont affichés sous la forme d'un espace car les afficher normalement poserait des problèmes, étant donné qu'ils ont un sens spécial. Le code peut faire appel à des éléments de syntaxe que vous n'avez pas encore vu, il est simplement donné à titre d'exemple.

```
# Caractères 0 à 1023
for bloc in range(8):
    for lig in range(16):
        for col in range(8):
            code = 128 * bloc + 16 * col + lig
            caractere = chr(code)
            if code < 32:
                caractere = " "
            print("{:04d} {}  ".format(code, caractere), end = "")
        print()
    print()
```

CLASSES DE CARACTERES :

Convertir en majuscules/minuscules :

Convertir un caractère

```
caractereMin = "d"
caractereMaj = "J"
print(caractereMin.upper())
print(caractereMaj.lower())
```

```
D
j
```

Notez que ces fonctions ne passeront en majuscule (resp. minuscule) que les caractères qui sont en minuscule (resp. majuscule). Il n'y a donc pas pas besoin de tester la casse des caractères avant de pouvoir les utiliser !

Convertir une chaîne de caractère

```
texte = "Texte avec 1 majuscule et 29 minuscules !"
print(texte.upper())
print(texte.lower())
```

```
TEXTE AVEC 1 MAJUSCULE ET 29 MINUSCULES !
texte avec 1 majuscule et 29 minuscules !
```

On remarquera que les caractères spéciaux ne sont pas modifiés, uniquement les lettres.

Étant donné un caractère, on souhaiterait savoir s'il appartient à certaines classes de caractères (chiffre, lettre, minuscule, majuscule...) afin d'effectuer des opérations différentes au sein du programme. Nous allons voir comment faire pour quelques classes très courantes.

```
caractere = input()
if caractere.isdigit():
   print("Il s'agit d'un chiffre")
if caractere.islower():
   print("Il s'agit d'une lettre minuscule")
if caractere.isupper():
   print("Il s'agit d'une lettre majuscule")
if caractere.isalpha():
   print("Il s'agit d'une lettre")
```

```
Il s'agit d'une lettre majuscule
Il s'agit d'une lettre
```

Avec des comparaisons de caractères

Il est préférable d'utiliser les fonctions toutes faites, mais il est aussi possible de se baser directement sur des comparaisons de caractères.

```
caractere = input()
if "0" <= caractere and caractere <= "9":
   print("Il s'agit d'un chiffre")
if "a" <= caractere and caractere <= "z":
   print("Il s'agit d'une lettre minuscule")
if "A" <= caractere and caractere <= "Z":
   print("Il s'agit d'une lettre majuscule")
if ("a" <= caractere and caractere <= "z") or ("A" <= caractere and caractere <= "Z"):
   print("Il s'agit d'une lettre")
```

```
Il s'agit d'une lettre majuscule
Il s'agit d'une lettre
```

Attention, le code ci-dessus ne va fonctionner qu'avec des caractères non accentués, c'est-à-dire les seuls que nous manipuleront dans les exercices. Si vous souhaitez un jour manipuler des caractères accentués, utilisez les fonctions toutes faites, qui elles marcheront.

Sur une chaîne complète

Les mêmes fonctions sont disponibles pour des chaînes de caractères complètes :

```
texte = input()
if texte.isdigit():
   print("Le texte ne contient que des chiffres")
if texte.islower():
   print("Le texte ne contient que des lettres minuscules")
if texte.isupper():
   print("Le texte ne contient que des lettres majuscules")
if texte.isalpha():
   print("Le texte ne contient que des lettres")
```

CARACTERES SPECIAUX :

Imaginons que l'on souhaite afficher le texte suivant :

```
Il m'a dit "Bonjour" !
```

Ne pressentez-vous pas un problème ? Le texte contient des guillemets ; or, nous utilisons justement des guillemets pour délimiter nos chaînes de caractères.

Pour résoudre ce problème, on va utiliser le caractère `\` (une barre oblique inversée, généralement appelée antislash). Il est dit « caractère d'échappement » car, placé devant un autre caractère, il permet d'annuler son sens spécial. Pour un guillemet, il va donc annuler la fonction de délimiteur d'une chaîne de caractères :

```
print("Il m'a dit \"Bonjour\" !")
```

Remarquez que `\` peut également être utilisé pour donner un sens particulier à caractère ; par exemple, `\n` symbolise un retour à la ligne, et `\t` symbolise une tabulation.

Mais, puisque `\` a un sens spécial, comment faire si on souhaite afficher ce caractère ? Et bien on va utiliser `"\\"`, le premier annulant le caractère spécial du second pour le transformer en simple caractère. Pour en afficher deux, on écrit donc `"\\\\"`.

Autres notations

En Python, une chaîne de caractère peut aussi être délimitée par des apostrophes (ou guillemets simples), à la place des guillemets doubles. On peut donc écrire plus simplement :

```
print('Il a dit "Bonjour" !')
```

Comme les guillemets simples délimitent la chaîne, pas besoin de transformer les guillemets doubles, il n'y a pas de confusion. Si la chaîne contient à la fois des guillemets simples et des guillemets doubles, alors on peut utiliser des triples guillemets :

```
print("""Il m'a dit "Bonjour" !""")
```

Attention, cela ne marchera pas si le premier (ou dernier) caractère de la chaîne est lui aussi un guillemet. Ce n'est donc pas une solution miracle, mais elle permet de régler la plupart des situations qui arrivent en pratique.