

# Cours sur le codage des caractères

Thème représentation des données : types et valeurs de base



## Introduction

Le but de ce chapitre est de comprendre comment les textes sont représentés dans un ordinateur et d'apprendre à manipuler le type `str` de Python qui leur est dédié.



## Crédits



Le contenu de ce cours est directement inspiré du chapitre 21 du *Manuel de première NSI* chez Ellipses.  
La plupart des exercices sont tirés de la plateforme <https://codex.forge.apps.education.fr/>.

## 1 Chaînes de caractères en Python



### Méthode

- Une chaîne de caractères est une séquence éventuellement vide de caractères. Chaque caractère est une chaîne de longueur 1. Nous étudierons la représentation informatique d'un caractère un peu plus loin.

En Python, les chaînes de caractères définissent le type `str`. La longueur d'une chaîne de caractères s'obtient avec la fonction `len`.

```
In [6]: a = 1

In [7]: type(a)
Out[7]: int

In [8]: b = str(a)

In [9]: b, type(b)
Out[9]: ('1', str)

In [10]: len(b), len('un'), len('')
Out[10]: (1, 2, 0)
```

- Les chaînes de caractères partagent certaines propriétés avec les tableaux de type `list` et les *p*-uplets de type `tuple` :
  - Une chaîne de caractères `chaine` peut être vue comme un tableau de caractères indexés de 0 à `len(chaine) - 1` de gauche à droite et de `-1` à `-len(chaine)` de droite à gauche. L'opérateur crochet permet d'accéder au caractère d'index `i` avec la syntaxe `chaine[i]` ou à la tranche de caractères d'index appartenant à l'intervalle `[i; j[` avec la syntaxe `chaine[i:j]` si  $0 \leq i \leq j$  ou à l'intervalle `]j; i]` avec la syntaxe `chaine[i:j:-1]` si  $-1 \geq i \geq j$ .



Le découpage en tranches et les index négatifs peuvent être pratiques pour manipuler les

chaînes de caractères mais ne sont pas au programme de première NSI.

```
In [26]: chaine = 'XYZT'

In [27]: chaine[0], chaine[1], chaine[len(chaine)-1]
Out[27]: ('X', 'Y', 'T')

In [28]: chaine[-1], chaine[-2], chaine[-len(chaine)]
Out[28]: ('T', 'Z', 'X')

In [29]: chaine[1:3], chaine[-1:-len(chaine):-1]
Out[29]: ('YZ', 'TZY')
```

- On peut itérer sur une chaîne de caractères avec une boucle for.

```
In [18]: for c in 'XY':
...:     print(c)
...:
X
Y
```

- On peut concaténer deux chaînes pour créer une nouvelle chaîne avec l'opérateur +.

```
In [19]: a, b, c = 'belle', '-', 'ile'

In [20]: a + b + c
Out[20]: 'belle-ile'
```

👉 Une chaîne de caractères d'identifiant `chaine` est un objet Python qui intègre de nombreuses méthodes accessibles avec la notation pointée `chaine.methode()`. On peut les découvrir en saisissant `dir(chaine)` ou dans la documentation en ligne :

<https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str>

```
In [24]: a = 'un beau marin'

In [25]: a.find('beau')
Out[25]: 3

In [26]: a.replace('b', 'v')
Out[26]: 'un veau marin'

In [27]: a.upper()
Out[27]: 'UN BEAU MARIN'

In [28]: 'un beau marin'.replace(' ', '')
Out[28]: 'unbeaumarin'

In [29]: dir(a)
Out[29]: ['__add__', '__class__', '__contains__', ... ]
```

 **Exercice 1**

Traiter dans le carnet Capytale de code **9aac-5273384** l'exercice [https://codex.forge.apps.education.fr/exercices/liste\\_longueurs/](https://codex.forge.apps.education.fr/exercices/liste_longueurs/) de la plateforme Codex.

 **Exercice 2**

Traiter dans le carnet Capytale de code **9aac-5273384** l'exercice <https://codex.forge.apps.education.fr/exercices/channiv/> de la plateforme Codex.

 **Exercice 3**

Traiter dans le carnet Capytale de code **9aac-5273384** l'exercice [https://codex.forge.apps.education.fr/exercices/anonymat\\_1/](https://codex.forge.apps.education.fr/exercices/anonymat_1/) de la plateforme Codex.

 **Exercice 4**

Traiter dans le carnet Capytale de code **9aac-5273384** l'exercice [https://codex.forge.apps.education.fr/exercices/renverse\\_chaine/](https://codex.forge.apps.education.fr/exercices/renverse_chaine/) de la plateforme Codex.

 **Exercice 5**

Traiter dans le carnet Capytale de code **9aac-5273384** l'exercice [https://codex.forge.apps.education.fr/exercices/est\\_nb\\_palindrome/](https://codex.forge.apps.education.fr/exercices/est_nb_palindrome/) de la plateforme Codex.

 **Exercice 6**

Traiter dans le carnet Capytale de code **9aac-5273384** l'exercice <https://codex.forge.apps.education.fr/exercices/dentiste/> de la plateforme Codex.

 **Exercice 7**

Traiter dans le carnet Capytale de code **9aac-5273384** l'exercice [https://codex.forge.apps.education.fr/exercices/validation\\_mdp/](https://codex.forge.apps.education.fr/exercices/validation_mdp/) de la plateforme Codex.

 **Exercice 8**

Traiter dans le carnet Capytale de code **9aac-5273384** l'exercice [https://codex.forge.apps.education.fr/exercices/est\\_inclus/](https://codex.forge.apps.education.fr/exercices/est_inclus/) de la plateforme e-nsi.

## 2 Du code ASCII à l'Unicode

*Humans use text. Computers speak bytes.*

Esther Nam and Travis Fischer, *Character encoding and Unicode in Python*

### 2.1 Table de codage et encodage



#### Point de cours 1

Une représentation informatique des caractères nécessite d'associer à chaque caractère une unique séquence d'octets. Cette conversion s'effectue à l'aide d'une **table de codage** et d'un **encodage**.

- ☞ Une **table de codage** ou **jeu de caractères** ou **charset**, associe un entier nommé **point de code** à un caractère.
- ☞ Un **encodage** associe à un **point de code** une séquence d'octets.



#### Remarque 1

- **Table de codage** et **encodage** sont des bijections : deux caractères distincts ont des points de code et des encodages distincts.
- Il peut exister plusieurs encodages pour une même table de codage. Par exemple la table Unicode admet les encodages UTF-8, UTF-16, UTF-32 ...

Par exemple le caractère '€' dont le point de code Unicode est 8364 a pour encodages (voir <https://www.compart.com/fr/unicode/U+20AC>) :

- 0xE2 0x82 0xAC en UTF-8 sur 3 octets notés en hexadécimal
- 0x20 0xAC en UTF-16 sur 2 octets notés en hexadécimal
- 0x00 0x00 0x20 0xAC en UTF-32 sur 4 octets notés en hexadécimal

- Avec l'internationalisation des échanges, on a besoin d'un encodage *compact* (économique en octets), *portable* (compatibilité entre les différents programmes) et *universel* (représentation de tous les caractères possibles).

### 2.2 Table de codage ASCII



#### Point de cours 2

Au début des années 1960, devant la multiplication des encodages de caractères et leurs incompatibilités mutuelles, l'ANSI (*American National Standards Institute*), définit une norme appelée **ASCII** (*American Standard Code for Information Interchange*).

Ce standard permet de coder 128 caractères sur 7 bits ( $2^7 = 128$ ), le bit de poids fort étant réservé à des contrôles d'erreur : ce **bit de parité** est choisi pour que le nombre total de bits à 1 dans l'octet soit toujours pair.

La table de codage **ASCII** représentée en Annexe 9 contient :

- les lettres de l'alphabet latin en majuscule et minuscule et les chiffres de 0 à 9;
- des signes de ponctuation et des opérateurs arithmétiques;
- des caractères spéciaux (tabulation, retour chariot, nouvelle ligne ...) et des caractères de contrôle non imprimables pour des protocoles de communication et des contrôles de périphériques (Fin de transmission, EOT, demande de transmission ENQ ...).

## Méthode

- ☞ En Python, la fonction `ord` associe à un caractère son point de code ASCII (en fait Unicode) et la fonction `chr` est sa réciproque :

```
In [39]: ord('a'), ord('z'), ord('A'), ord('Z')
Out[39]: (97, 122, 65, 90)

In [40]: chr(65), chr(90)
Out[40]: ('A', 'Z')

In [41]: [chr(ord('a') + k) for k in range(26)]
Out[41]: ['a', 'b', 'c', '...', 'y', 'z']

In [42]: ''.join([chr(ord('a') + k) for k in range(26)])
Out[42]: 'abcdefghijklmnopqrstuvwxyz'
```

- ☞ Les opérateurs de comparaison entre caractères `==`, `<` et `>` comparent les points de code.

```
In [3]: 'a' < 'A'
Out[3]: False

In [4]: ord('a'), ord('A')
Out[4]: (97, 65)
```

- ☞ Pour les chaînes de caractères, la comparaison s'effectue de gauche à droite caractère par caractère. En cas d'inclusion d'une chaîne dans l'autre, la plus petite est celle qui est incluse. On parle d'**ordre lexicographique**.



Si on compare deux entiers convertis en chaînes de caractères l'ordre lexicographique peut ne pas donner le même résultat que l'ordre numérique.

```
In [7]: 'abc' < 'abcd'
Out[7]: True

In [8]: 'bbc' > 'ab'
Out[8]: True

In [9]: '2' > '10'
Out[9]: True
```

 **Exercice 9**

Traiter dans le carnet Capytale de code **9aac-5273384** l'exercice [https://codex.forge.apps.education.fr/exercices/conversion\\_chaines/](https://codex.forge.apps.education.fr/exercices/conversion_chaines/) de la plateforme Codex.

 **Exercice 10**

1. Définir le chiffrement d'une chaîne de caractères par l'algorithme rot13.

Ressources : <https://fr.wikipedia.org/wiki/ROT13>

.....  
.....

2. L'image par rot13 du caractère 'R' est 'E'.

- on calcule d'abord le rang alphabétique d'un caractère :

```
In [56]: ord('R') - ord('A')  
Out[56]: 17
```

- puis on ajoute 13 à ce rang et on prend le reste dans la division euclidienne par 26

```
In [59]: (ord('R') - ord('A') + 13) % 26  
Out[59]: 4
```

- enfin on retrouve le rang du caractère associé au rang alphabétique calculé :

```
In [60]: chr(ord('A') + 4)  
Out[60]: 'E'
```

Écrire une fonction `rot13(chaine)` qui chiffre ou déchiffre la chaîne (en majuscules ou convertie en majuscules avec `chaine.upper()`) passée en paramètre, avec l'algorithme rot13.

.....  
.....  
.....  
.....  
.....

3. Dans un interpréteur Ipython, importer le module `this`, lire le *Zen de Python*, méditer puis afficher le code source du module avec la commande `%psource this`.

## 2.3 Tables de codages ISO-8859



### Point de cours 3

La table **ASCII** possédait les propriétés de compacité et de portabilité mais pas celle d'universalité. Par exemple les caractères accentués de l'alphabet latin ne figurent pas dans la table **ASCII**.

Pour y remédier, l'**ISO** (*Organisation internationale de normalisation*) a proposé la norme **ISO-8859** qui étend la table **ASCII** en utilisant le huitième bit. Sur un octet, on peut ainsi représenter deux fois plus de caractères, soit  $2^8 = 256$ .

La norme **ISO 8859** comprend seize tables de codages, certes compatibles entre elles et avec ASCII sur les 128 premiers caractères, mais avec 256 caractères il n'est pas possible d'écrire un texte avec un mélange de caractères arabes et russes.

La table de codage **ISO 8859-1** ou **Latin1** est la plus utilisée en Occident. Les applications Microsoft utilisent une variante de Latin1 nommé **cp1252** où les caractères de contrôle de point de code entre 128 et 159 ont été remplacés par des caractères imprimables comme le symbole de l'euro €.

La multiplicité des encodages a posé des problèmes d'**interopérabilité** avec l'internationalisation des échanges informatiques au-delà des aires de proximité linguistique. Qui n'a jamais reçu de courriel avec d'étranges caractères comme ci-dessous ?

Je transf??re le message ?? Valentin

L'encodage du message n'étant pas reconnu par le client de messagerie, les caractères accentués è et à ont été remplacés par des signes ?.



### Exercice 11 QCM E3C 2

1. Sur une page web qui s'affiche sur notre navigateur on peut lire :

« En consÃ©quence, l'AssemblÃ©e Nationale reconnaÃ®t et dÃ©clare, en prÃ©sence »

Quelle peut être la cause des affichages étranges de cette page ?

- a. l'encodage des caractères n'est pas celui attendu par le navigateur
  - b. le texte original est en japonais
  - c. la taille des caractères n'est pas celle attendue par le navigateur
  - d. la connexion à Internet présente des coupures
2. Le code ASCII permet de représenter en binaire les caractères alphanumériques. Quel est son principal inconvénient ?
- a. Il utilise beaucoup de bits.
  - b. Il ne différencie pas les majuscules des minuscules.
  - c. Il ne représente pas les caractères accentués.
  - d. Il n'est pas compatible avec la plupart des systèmes informatiques.

## 2.4 Table de codage universelle et encodages Unicode



### Point de cours 4

Pour assurer l'universalité de la représentation des caractères, l'**ISO** a défini une table de codage universelle, l'*Universal Character Set* sous la norme **ISO-10646**. Cette norme a défini une plage de **points de codes** pouvant être associés à 1 114 112 caractères. Les points de code sont codés sur 21 bits. Depuis 1991, le Consortium Unicode collabore avec l'ISO pour développer le Standard Unicode qui partage la même plage de points de codes que la norme **ISO-10646** avec des informations complémentaires (normalisation de forme). En mars 2020, 137 929 caractères sont recensés.

Chaque point de code est de la forme  $U+xx \dots xx$  où chaque chiffre  $x$  est un caractère hexadécimal (base 16) avec au moins quatre chiffres.

Il serait possible d'encoder tous les points de code Unicode (21 bits) sur 3 octets, mais des encodages avec un nombre d'octets variable ont été imaginés : entre 1 et 4 octets car certains bits sont utilisés pour distinguer les caractères. En effet, il suffit d'un octet pour stocker les caractères **ASCII** et de deux octets pour les plus courants dont le point de code est compris entre 0 et  $2^{16} - 1 = 65\,535$ .

Les encodages **Unicode** sont notés **UTF- $n$**  où  $n$  est le nombre minimal de bits pour représenter un point de code.

L'encodage **Unicode** le plus courant est **UTF-8** : les caractères **ASCII** y sont encodés sur 1 octet.



### Remarque 2



En Python 3, l'encodage par défaut des fichiers sources est **UTF-8**. Si le fichier n'est pas encodé en **UTF-8** mais par exemple en **cp1252**, on peut avoir un message d'erreur à l'exécution :

```
File "script-encodage-cp1252.py", line 1
SyntaxError: Non-UTF-8 code starting with '\xe9' in file script-encodage-
cp1252.py on line 1, but no encoding declared; see http://python.org
/dev/peps/pep-0263/ for details
```

Pour éviter ce type d'erreur, on peut placer un commentaire spécial précisant l'encodage sur la première ligne du fichier :

```
# coding: cp1252
```

La meilleure solution est tout de même de convertir le fichier source en **UTF-8**, ce que permet un bon éditeur de texte comme **Notepad++**.



### Exercice 12 QCM E3C 2

1. Parmi les caractères ci-dessous, lequel ne fait pas partie du code ASCII ?

a. a

b. B

c. @

d. é

2. Laquelle de ces affirmations concernant le codage UTF-8 des caractères est vraie ?

a. le codage UTF-8 est sur 7 bits

c. le codage UTF-8 est sur 1 à 4 octets

b. le codage UTF-8 est sur 8 bits

d. le codage UTF-8 est sur 8 octets

### 3. Quel est un avantage du codage UTF-8 par rapport au codage ASCII?

- il permet de coder un caractère sur un octet au lieu de deux
- il permet de coder les majuscules
- il permet de coder tous les caractères
- il permet de coder différentes polices de caractères

## ANNEXE

### Table de codage ASCII

| Hex  | Dec | Char                       | Hex  | Dec | Char  | Hex  | Dec | Char | Hex  | Dec | Char |
|------|-----|----------------------------|------|-----|-------|------|-----|------|------|-----|------|
| 0x00 | 0   | NULL null                  | 0x20 | 32  | Space | 0x40 | 64  | @    | 0x60 | 96  | `    |
| 0x01 | 1   | SOH Start of heading       | 0x21 | 33  | !     | 0x41 | 65  | A    | 0x61 | 97  | a    |
| 0x02 | 2   | STX Start of text          | 0x22 | 34  | "     | 0x42 | 66  | B    | 0x62 | 98  | b    |
| 0x03 | 3   | ETX End of text            | 0x23 | 35  | #     | 0x43 | 67  | C    | 0x63 | 99  | c    |
| 0x04 | 4   | EOT End of transmission    | 0x24 | 36  | \$    | 0x44 | 68  | D    | 0x64 | 100 | d    |
| 0x05 | 5   | ENQ Enquiry                | 0x25 | 37  | %     | 0x45 | 69  | E    | 0x65 | 101 | e    |
| 0x06 | 6   | ACK Acknowledge            | 0x26 | 38  | &     | 0x46 | 70  | F    | 0x66 | 102 | f    |
| 0x07 | 7   | BELL Bell                  | 0x27 | 39  | '     | 0x47 | 71  | G    | 0x67 | 103 | g    |
| 0x08 | 8   | BS Backspace               | 0x28 | 40  | (     | 0x48 | 72  | H    | 0x68 | 104 | h    |
| 0x09 | 9   | TAB Horizontal tab         | 0x29 | 41  | )     | 0x49 | 73  | I    | 0x69 | 105 | i    |
| 0x0A | 10  | LF New line                | 0x2A | 42  | *     | 0x4A | 74  | J    | 0x6A | 106 | j    |
| 0x0B | 11  | VT Vertical tab            | 0x2B | 43  | +     | 0x4B | 75  | K    | 0x6B | 107 | k    |
| 0x0C | 12  | FF Form Feed               | 0x2C | 44  | ,     | 0x4C | 76  | L    | 0x6C | 108 | l    |
| 0x0D | 13  | CR Carriage return         | 0x2D | 45  | -     | 0x4D | 77  | M    | 0x6D | 109 | m    |
| 0x0E | 14  | SO Shift out               | 0x2E | 46  | .     | 0x4E | 78  | N    | 0x6E | 110 | n    |
| 0x0F | 15  | SI Shift in                | 0x2F | 47  | /     | 0x4F | 79  | O    | 0x6F | 111 | o    |
| 0x10 | 16  | DLE Data link escape       | 0x30 | 48  | 0     | 0x50 | 80  | P    | 0x70 | 112 | p    |
| 0x11 | 17  | DC1 Device control 1       | 0x31 | 49  | 1     | 0x51 | 81  | Q    | 0x71 | 113 | q    |
| 0x12 | 18  | DC2 Device control 2       | 0x32 | 50  | 2     | 0x52 | 82  | R    | 0x72 | 114 | r    |
| 0x13 | 19  | DC3 Device control 3       | 0x33 | 51  | 3     | 0x53 | 83  | S    | 0x73 | 115 | s    |
| 0x14 | 20  | DC4 Device control 4       | 0x34 | 52  | 4     | 0x54 | 84  | T    | 0x74 | 116 | t    |
| 0x15 | 21  | NAK Negative ack           | 0x35 | 53  | 5     | 0x55 | 85  | U    | 0x75 | 117 | u    |
| 0x16 | 22  | SYN Synchronous idle       | 0x36 | 54  | 6     | 0x56 | 86  | V    | 0x76 | 118 | v    |
| 0x17 | 23  | ETB End transmission block | 0x37 | 55  | 7     | 0x57 | 87  | W    | 0x77 | 119 | w    |
| 0x18 | 24  | CAN Cancel                 | 0x38 | 56  | 8     | 0x58 | 88  | X    | 0x78 | 120 | x    |
| 0x19 | 25  | EM End of medium           | 0x39 | 57  | 9     | 0x59 | 89  | Y    | 0x79 | 121 | y    |
| 0x1A | 26  | SUB Substitute             | 0x3A | 58  | :     | 0x5A | 90  | Z    | 0x7A | 122 | z    |
| 0x1B | 27  | FSC Escape                 | 0x3B | 59  | ;     | 0x5B | 91  | [    | 0x7B | 123 | {    |
| 0x1C | 28  | FS File separator          | 0x3C | 60  | <     | 0x5C | 92  | \    | 0x7C | 124 |      |
| 0x1D | 29  | GS Group separator         | 0x3D | 61  | =     | 0x5D | 93  | ]    | 0x7D | 125 | }    |
| 0x1E | 30  | RS Record separator        | 0x3E | 62  | >     | 0x5E | 94  | ^    | 0x7E | 126 | ~    |
| 0x1F | 31  | US Unit separator          | 0x3F | 63  | ?     | 0x5F | 95  | _    | 0x7F | 127 | DEL  |