Chaînes de caractères – Analyse d'un texte

Tu vas faire quelques activités amusantes en manipulant les chaînes de caractères.

Vidéo ■ Chaîne de caractères - partie 1
Vidéo ■ Chaîne de caractères - partie 2

Cours 1 (Caractère et chaîne)**.**

1. Un ***caractère*** est un symbole unique, par exemple une lettre minuscule `"a"`, une lettre majuscule `"B"`, un symbole spécial `"&"`, un symbole représentant un chiffre `"7"`, une espace `" "` que l'on notera aussi `"␣"`.

   Pour désigner un caractère, il faut le mettre entre guillemets simples `'z'` ou entre guillemets doubles `"z"`.
2. Une ***chaîne de caractères*** est une suite de caractères, comme un mot `"Bonjour"`, une phrase `'Il fait beau.'`, un mot de passe `"N[w5ms}e!"`.
3. Le type d'un caractère ou d'une chaîne est `str` (pour *string*).

Cours 2 (Opérations sur les chaînes)**.**

1. La ***concaténation***, c'est-à-dire la mise bout à bout de deux chaînes, s'effectue à l'aide de l'opérateur `+`. Par exemple `"para"+"pluie"` donne la chaîne `"parapluie"`.
2. La chaîne vide `""` est utile lorsque l'on veut initialiser une chaîne avant d'y ajouter d'autres caractères.
3. La ***longueur*** d'une chaîne est le nombre de caractères qu'elle contient. Elle s'obtient par la fonction `len()`. Par exemple `len("Hello␣World")` renvoie 11 (l'espace compte comme un caractère).
4. Si `mot` est une chaîne alors on peut récupérer chaque caractère par `mot[i]`. Par exemple si `mot = "avion"` alors :
   - `mot[0]` est le caractère `"a"`,
   - `mot[1]` est le caractère `"v"`,
   - `mot[2]` est le caractère `"i"`,
   - `mot[3]` est le caractère `"o"`,
   - `mot[4]` est le caractère `"n"`.

Lettre	a	v	i	o	n
Rang	0	1	2	3	4

Note qu'il y a 5 lettres dans le mot `"avion"` et qu'on y accède par les indices en commençant par 0. Les indices sont donc ici 0, 1, 2, 3 et 4 pour la dernière lettre. De façon plus générale, si `mot` est une chaîne, les caractères s'obtiennent par `mot[i]` pour `i` variant de 0 à `len(mot)-1`.

Cours 3 (Sous-chaînes)**.**
On peut extraire plusieurs caractères d'une chaîne à l'aide de la syntaxe `mot[i:j]` qui renvoie une chaîne formée des caractères numéro i à $j-1$ (attention le caractère numéro j n'est pas inclus !).
Par exemple si `mot = "vendredi"` alors :

- `mot[0:4]` renvoie la sous-chaîne `"vend"` formée des caractères de rang 0, 1, 2 et 3 (mais pas 4),
- `mot[3:6]` renvoie `"dre"` correspondant aux rangs 3, 4 et 5.

Lettre	v	e	n	d	r	e	d	i
Rang	0	1	2	3	4	5	6	7

Autre exemple : `mot[1:len(mot)-1]` renvoie le mot privé de sa première et dernière lettre.

Activité 1 (Pluriels des mots)**.**

Objectifs : écrire petit à petit un programme qui renvoie le pluriel d'un mot donné.

1. Pour une chaîne `mot`, par exemple `"chat"`, affiche le pluriel de ce mot en rajoutant un `"s"`.
2. Pour un mot, par exemple `"souris"`, affiche la dernière lettre de cette chaîne (ici `"s"`). Améliore ton programme de la première question, en testant si la dernière lettre est déjà un `"s"` :
   - si c'est le cas, il n'y a rien à faire pour le pluriel,
   - sinon il faut ajouter un `"s"`.
3. Teste si un mot se termine par `"al"`. Si c'est le cas, affiche le pluriel en `"aux"` (le pluriel de `"cheval"` est `"chevaux"`). (Ne tiens pas compte des exceptions.)
4. Rassemble tout ton travail des trois premières questions dans une fonction `met_au_pluriel()`. La fonction n'affiche rien, mais renvoie le mot au pluriel.

`met_au_pluriel()`

Usage : `met_au_pluriel(mot)`
Entrée : un mot (une chaîne de caractères)
Sortie : le pluriel du mot

Exemples :
- `met_au_pluriel("chat")` renvoie `"chats"`
- `met_au_pluriel("souris")` renvoie `"souris"`
- `met_au_pluriel("cheval")` renvoie `"chevaux"`

5. Écris une fonction `affiche_conjugaison()` qui conjugue un verbe du premier groupe au présent.

`affiche_conjugaison()`

Usage : `affiche_conjugaison(verbe)`
Entrée : un verbe du premier groupe (une chaîne de caractères se terminant par `"er"`)
Sortie : pas de résultat mais l'affichage de la conjugaison du verbe au présent

Exemple :
- `affiche_conjugaison("chanter")`, affiche `"je chante, tu chantes,..."`
- `affiche_conjugaison("choisir")`, affiche `"Ce n'est pas un verbe du premier groupe."`

Cours 4 (Un peu plus sur les chaînes)**.**

1. Une boucle `for ... in ...` permet de parcourir une chaîne, caractère par caractère :
```
for carac in mot:
    print(carac)
```
2. On peut tester si un caractère appartient à une certaine liste de caractères. Par exemple :
```
if carac in ["a", "A", "b", "B", "c", "C"]:
```
permet d'exécuter des instructions si le caractère `carac` est l'une des lettres a, A, b, B, c, C.
Pour éviter certaines lettres, on utiliserait :
```
if carac not in ["X", "Y", "Z"]:
```

Activité 2 (Jeux de mots)**.**

Objectifs : manipuler des mots de façon amusante.

1. **Distance entre deux mots.**
La distance de Hamming entre deux mots de même longueur est le nombre d'endroits où les lettres sont différentes.
Par exemple :

JAPON **SAVON**

La première lettre de **JAPON** est différente de la première lettre de **SAVON**, les troisièmes aussi sont différentes. La distance de Hamming entre **JAPON** et **SAVON** vaut donc 2.
Écris une fonction `distance_hamming()` qui calcule la distance de Hamming entre deux mots de même longueur.

`distance_hamming()`

Usage : `distance_hamming(mot1,mot2)`
Entrée : deux mots (des chaînes de caractères)
Sortie : la distance de Hamming (un entier)

Exemple : `distance_hamming("LAPIN","SATIN")` renvoie 2

2. **Latin-cochon.**

   On transforme un mot commençant par une consonne selon la recette suivante :
   - on déplace la première lettre à la fin du mot ;
   - on rajoute le suffixe **UM**.

   Par exemple **VITRE** devient **ITREVUM** ; **BLANCHE** devient **LANCHEBUM** ; **CARAMEL** devient **ARAMELCUM**. Les mots commençant par une voyelle ne changent pas. Écris une fonction `latin_cochon()` qui transforme un mot selon ce procédé.

   `latin_cochon()`

   Usage : `latin_cochon(mot)`
   Entrée : un mot (une chaîne de caractères)
   Sortie : le mot transformé en latin-cochon, s'il commence par une consonne.

   Exemple : `latin_cochon("BONJOUR")` renvoie `"ONJOURBUM"`

3. **Verlan.**

   Écris une fonction `verlan()` qui renvoie un mot à l'envers : **SALUT** devient **TULAS**.

   `verlan()`

   Usage : `verlan(mot)`
   Entrée : un mot (une chaîne de caractères)
   Sortie : le mot à l'envers

   Exemple : `verlan("TOCARD")` renvoie `"DRACOT"`

4. **Palindrome.**

   Déduis-en une fonction qui teste si un mot est un palindrome ou pas. Un *palindrome* est un mot qui s'écrit indifféremment de gauche à droite ou de droite à gauche ; par exemple **RADAR** est un palindrome.

   `est_un_palindrome()`

   Usage : `est_un_palindrome(mot)`
   Entrée : un mot (une chaîne de caractères)
   Sortie : « vrai » si le mot est un palindrome, « faux » sinon.

   Exemple : `est_un_palindrome("KAYAK")` renvoie `True`

Activité 3 (ADN)**.**

*Une molécule d'ADN est formée d'environ six milliards de nucléotides. L'ordinateur est donc un outil indispensable pour l'analyse de l'ADN. Dans un brin d'ADN il y a seulement quatre types de nucléotides qui sont notés **A**, **C**, **T** ou **G**. Une séquence d'ADN est donc un long mot de la forme : **TAATTACAGACCTGAA...***

1. Écris une fonction `presence_de_A()` qui teste si une séquence contient le nucléotide **A**.

`presence_de_A()`

Usage : `presence_de_A(sequence)`
Entrée : une séquence d'ADN (une chaîne de caractères parmi A, C, T, G)
Sortie : « vrai » si la séquence contient « A », « faux » sinon.

Exemple : `presence_de_A("CTTGCT")` renvoie `False`

2. Écris une fonction `position_de_AT()` qui teste si une séquence contient le nucléotide **A** suivi du nucléotide **T** et renvoie la position de la première occurrence trouvée.

`position_de_AT()`

Usage : `position_de_AT(sequence)`
Entrée : une séquence d'ADN (une chaîne de caractères parmi A, C, T, G)
Sortie : la position de la première séquence « AT » trouvée (commence à 0) ; `None` si n'apparaît pas

Exemple :
- `position_de_AT("CTTATGCT")` renvoie 3
- `position_de_AT("GATATAT")` renvoie 1
- `position_de_AT("GACCGTA")` renvoie `None`

Indication. `None` est affecté à une variable pour signifier l'absence de valeur.

3. Écris une fonction `position()` qui teste si une séquence contient un code donné et renvoie la position de la première occurrence.

`position()`

Usage : `position(code,sequence)`
Entrée : un code et une séquence d'ADN
Sortie : la position du début du code trouvé ; `None` si n'apparaît pas

Exemple : `position("CCG","CTCCGTT")` renvoie 2

4. Un crime a été commis dans le château d'Adéno. Tu as récupéré deux brins d'ADN, provenant de deux positions éloignées de l'ADN du coupable. Il y a quatre suspects, dont tu as séquencé l'ADN. Sauras-tu trouver qui est le coupable ?

   Premier code du coupable : **CATA**

   Second code du coupable : **ATGC**

   ADN du colonel Moutarde :
   CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGC
   ADN de Mlle Rose :
   CTCCTGATGCTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGG
   ADN de Mme Pervenche :
   AAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGTACTCCGCGCGCCGGGACAGAATGCC
   ADN de M. Leblanc :
   CTGCAGGAACTTCTTCTGGAAGTACTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAG

Cours 5 (Codage des caractères)**.**
Un caractère est stocké par l'ordinateur sous la forme d'un entier. Pour le codage ASCII/unicode, la lettre majuscule « A » est codé par 65, la lettre minuscule « h » est codée par 104, le symbole « # » par 35.
Voici la table des premiers caractères. Les numéros 0 à 32 ne sont pas des caractères imprimables. Cependant le numéro 32 est le caractère espace " ".

33 !	43 +	53 5	63 ?	73 I	83 S	93]	103 g	113 q	123 {
34 "	44 ,	54 6	64 @	74 J	84 T	94 ^	104 h	114 r	124 |
35 #	45 -	55 7	65 A	75 K	85 U	95 _	105 i	115 s	125 }
36 $	46 .	56 8	66 B	76 L	86 V	96 ‘	106 j	116 t	126 ~
37 %	47 /	57 9	67 C	77 M	87 W	97 a	107 k	117 u	127 -
38 &	48 0	58 :	68 D	78 N	88 X	98 b	108 l	118 v	
39 ’	49 1	59 ;	69 E	79 O	89 Y	99 c	109 m	119 w	
40 (	50 2	60 <	70 F	80 P	90 Z	100 d	110 n	120 x	
41)	51 3	61 =	71 G	81 Q	91 [	101 e	111 o	121 y	
42 *	52 4	62 >	72 H	82 R	92 \	102 f	112 p	122 z	

1. La fonction `chr()` est une fonction `Python` qui renvoie le caractère associé à un code.

python : chr()

Usage : `chr(code)`
Entrée : un code (un entier)
Sortie : un caractère

Exemple :
- `chr(65)` renvoie `"A"`
- `chr(123)` renvoie `"{"`

2. La fonction `ord()` est une fonction `Python` correspondant à l'opération inverse : elle renvoie le code associé à un caractère.

python : ord()

Usage : `ord(carac)`
Entrée : un caractère (une chaîne de longueur 1)
Sortie : un entier

Exemple :
- `ord("A")` renvoie 65
- `ord("*")` renvoie 42

Activité 4 (Majuscules/minuscules).

Objectifs : convertir un mot en majuscules ou en minuscules.

1. Décode à la main le message chiffré sous les codes suivants :
   80-121-116-104-111-110 101-115-116 115-121-109-112-64
2. Écris une boucle qui affiche les caractères codés par les entiers de 33 à 127.
3. Que renvoie la commande `chr(ord("a")-32)` ? Et `chr(ord("B")+32)` ?
4. Écris une fonction `lettre_majuscule()` qui transforme un lettre minuscule en sa lettre majuscule.

`lettre_majuscule()`

Usage : `lettre_majuscule(carac)`
Entrée : un caractère minuscule parmi `"a"`,...,`"z"`
Sortie : la même lettre en majuscule

Exemple : `lettre_majuscule("t")` renvoie `"T"`

5. Écris une fonction `majuscules()` qui à partir d'une phrase écrite en minuscules renvoie la même phrase écrite en majuscules. Les caractères qui ne sont pas des lettres minuscules restent inchangés.

`majuscules()`

Usage : `majuscules(phrase)`
Entrée : une phrase
Sortie : la même phrase en majuscules

Exemple : `majuscules("Bonjour le monde !")` renvoie `"BONJOUR LE MONDE !"`

Fais le travail semblable pour une fonction `minuscules()`.

6. Écris une fonction `formate_prenom_nom()` qui renvoie le prénom et le nom formatés suivant le style **Prenom NOM**.

`formate_prenom_nom()`

Usage : `formate_prenom_nom(personne)`
Entrée : le prénom et le nom d'une personne (sans accent, séparés par une espace)
Sortie : le nom complet au format `"Prenom NOM"`

Exemple :
- `formate_prenom_nom("harry Potter")` renvoie `"Harry POTTER"`
- `formate_prenom_nom("LORD Voldemort")` renvoie `"Lord VOLDEMORT"`

Activité 5.

Objectifs : déterminer la langue d'un texte à partir de l'analyse des fréquences des lettres.

1. Écris une fonction `occurrences_lettre()` qui compte le nombre de fois où la lettre donnée apparaît dans une phrase (en majuscules et sans accents).

`occurrences_lettre()`
Usage : `occurrences_lettre(lettre,phrase)` Entrée : une lettre et une phrase en majuscules (une chaîne de caractères) Sortie : le nombre d'occurrences de la lettre (un entier) Exemple : `occurrences_lettre("E","ESPRIT ES TU LA")` renvoie 2

2. Écris une fonction `nombre_lettres()` qui compte le nombre total de lettres qui apparaissent dans une phrase (en majuscules et sans accents). Ne pas compter les espaces, ni la ponctuation.

`nombre_lettres()`
Usage : `nombre_lettres(phrase)` Entrée : une phrase en majuscules (une chaîne de caractères) Sortie : le nombre total de lettres de « A » à « Z » (un entier) Exemple : `nombre_lettres("ESPRIT ES TU LA")` renvoie 12

3. La ***fréquence d'apparition*** d'une lettre dans un texte ou une phrase est le pourcentage donné selon la formule :

$$\text{fréquence d'apparition d'une lettre} = \frac{\text{nombre d'occurrences de la lettre}}{\text{nombre total de lettres}} \times 100.$$

Par exemple, la phrase **ESPRIT ES TU LA** contient 12 lettres ; la lettre **E** y apparaît 2 fois. La fréquence d'apparition de **E** dans cette phrase est donc :

$$f_E = \frac{\text{nombre d'occurrences de } \mathbf{E}}{\text{nombre total de lettres}} \times 100 = \frac{2}{12} \times 100 \simeq 16.66$$

La fréquence est donc d'environ 17%.

Écris une fonction `pourcentage_lettre()` qui calcule cette fréquence d'apparition.

`pourcentage_lettre()`
Usage : `pourcentage_lettre(lettre,phrase)` Entrée : une lettre et une phrase en majuscules (une chaîne de caractères) Sortie : la fréquence d'apparition de la lettre (un nombre inférieur à 100) Exemple : `pourcentage_lettre("E","ESPRIT ES TU LA")` renvoie `16.66...`

Utilise cette fonction pour afficher proprement la fréquence d'apparition de toutes les lettres d'une phrase.

4. Voici la fréquence d'apparition des lettres selon la langue utilisée (source : en.wikipedia.org/wiki/Letter_frequency). Par exemple, la lettre la plus courante en français est le « e » avec une fréquence de plus de 16%. Le « w » représente environ 2% des lettres en anglais et en allemand, mais n'apparaît presque pas en français et en espagnol. Ces fréquences varient aussi en fonction du texte analysé.

Lettre	Anglais	Français	Allemand	Espagnol
a	8.167%	8.173%	7.094%	12.027%
b	1.492%	0.901%	1.886%	2.215%
c	2.782%	3.345%	2.732%	4.019%
d	4.253%	3.669%	5.076%	5.010%
e	12.702%	16.734%	16.396%	12.614%
f	2.228%	1.066%	1.656%	0.692%
g	2.015%	0.866%	3.009%	1.768%
h	6.094%	0.737%	4.577%	0.703%
i	6.966%	7.579%	6.550%	6.972%
j	0.153%	0.613%	0.268%	0.493%
k	0.772%	0.049%	1.417%	0.011%
l	4.025%	5.456%	3.437%	4.967%
m	2.406%	2.968%	2.534%	3.157%
n	6.749%	7.095%	9.776%	7.023%
o	7.507%	5.819%	3.037%	9.510%
p	1.929%	2.521%	0.670%	2.510%
q	0.095%	1.362%	0.018%	0.877%
r	5.987%	6.693%	7.003%	6.871%
s	6.327%	7.948%	7.577%	7.977%
t	9.056%	7.244%	6.154%	4.632%
u	2.758%	6.429%	5.161%	3.107%
v	0.978%	1.838%	0.846%	1.138%
w	2.360%	0.074%	1.921%	0.017%
x	0.150%	0.427%	0.034%	0.215%
y	1.974%	0.128%	0.039%	1.008%
z	0.074%	0.326%	1.134%	0.467%

D'après toi, dans quelles langues ont été écrits les quatre textes suivants (les lettres de chaque mot ont été mélangées).

TMAIER BERACUO RSU NU REBRA PRCEEH EIANTT NE ONS EBC NU GAOFREM EIMATR RERNAD APR L RDUOE LAHECLE UIL TTNI A EUP SREP EC LGNGAEA TE RBONUJO ERMNOUSI DU UBRACEO QUE OVSU EEST LIJO UQE OUVS EM MSZELBE BAEU ASNS MIERNT IS RVETO AGRAME ES PRARPTOE A OEVTR AMGUPLE VUOS SEET EL PNIHXE DSE OSHET ED CSE BIOS A ESC MSOT LE OUBRCEA NE ES ESTN ASP DE IEJO TE OUPR ERRNOTM AS BELEL XOVI IL OREVU NU RGLEA ECB ILESSA EBOMTR AS PIOER EL NRDAER S EN ISIAST TE ITD MNO NOB EUSRMNOI NRPEEAZP QEU UTOT EUTLRFTA IVT XUA SPNEDE DE UECIL UQI L TECEOU TECET NEOCL VATU BNEI UN GMAEORF SNAS TUOED LE EOABURC OHENTXU TE NSCOFU UJRA SMIA UN EPU TRDA UQ NO EN L Y ARRPEIDNT ULSP

WRE TREITE SO TSPA CUDHR AHNCT UND WIND SE STI RED AEVRT MTI ESEIMN IDNK RE ATH END NEABNK WLOH IN EMD AMR ER AFTSS HIN IHSERC RE AHTL HIN MRWA EINM SHNO SAW SRTIBG UD SO NGBA DNEI EIHSGTC ESISTH RAETV UD DEN LERNIOKG NITHC NDE LOENINKGRE TIM OKRN UDN CHWFSEI NEIM NSOH ES STI IEN BIFTRLSEEEN DU BILESE IKDN OMKM EHG MIT MIR RAG ECHNOS EPELSI EIPSL IHC ITM RDI HNCMA BEUTN MBLUNE DINS NA DEM TNDRAS NMIEE UTETMR AHT CAMHN UDNGEL GDAWEN MIEN EATRV MENI VEART DUN OSTHER DU CINTH SAW KNNOEIREGL RIM ILEES PRSTVRCIEH ISE IHGRU BEEILB RIGUH MNEI KNDI NI RDNEUR NATBRLET STAESUL EDR WNID

DSNOACAIF ORP ANU DAEDALRI DNAAEIMTI EQU NNCOSETE EL RSTEOUL SMA AACTFAITNS UQE LE TSVAO OINSRVUE DE US ANIGIICANOM EIORDP TOOD RTEIENS RPO LE ITOABOLRROA ED QIUAMALI

USOP A NSSRCAEAD LA TMREAAI NXTADAUEE ROP GOARLS EMESS DE NNAMICLUIAPO Y LOVOIV A RES LE RHMEOB EOMDNEERPRD DE LOS RSOPMRIE OMTSIPE UEQ CIIDADE LE RTDAAOZ ED LSA CELSAL Y LA NICOIOPS ED LAS UESVNA SSACA Y ES ITRMNEEOD QEU AERFU EL UEQIN IIIRDEGAR LA NAIORTREICP DE AL RRTEIA

IMTRUESMME DNA TEH LNGIIV SI EYAS SIFH REA GJPNUIM DNA HET TTNOCO IS GHIH OH OUYR DDADY SI IRHC DAN ROUY MA SI DOGO GKOILON OS USHH LTLIET BBYA NDOT OUY CYR NEO OF HESET GNSRONIM YUO RE NANGO SIER PU SNIGING NAD OULLY EPADRS YUOR GINSW DAN LYOLU KATE OT HET KSY TUB ITLL TATH MGNIRNO EREHT NATI INTGOHN ACN AHMR OYU TWIH DADYD NDA MYMMA NSTIDGAN YB