Les chaînes de caractères - parcours

1 Parcours

Définition

Une chaîne de caractères est itérable : cela signifie qu'une boucle bornée (boucle `for`) permet de parcourir chacun des caractères d'une chaîne `s` selon la syntaxe ci-dessous.

```
for char in s:
    # à chaque tour de boucle, char prend successivement la
    # valeur des caractères de s
    print(char) # par exemple
```

Exercice 10

Quels sont les affichages réalisés par le script suivant ?

```
s = "abcde"
for idx in range(len(s)):
    print(idx, s[idx])
```

Exercice 11

Écrire une fonction `nb_e(s)` qui renvoie le nombre d'occurrences de la lettre `"e"` dans une chaîne de caractères `s`.

Exercice 12

Écrire une fonction `est_binaire_valide` qui prend en argument une chaîne de caractères `s` et renvoie :

- `True` si `s` ne contient que les caractères `0` et `1` ;
- `False` sinon.

Exercice 13

1. Écrire une fonction `valeur_decimale` qui prend en argument une chaîne de caractères représentant un nombre écrit en binaire, et renvoie sa valeur décimale.
2. Même question pour une chaîne de caractères représentant un nombre écrit en hexadécimal.

Exercice 14

Écrire une fonction `est_palindrome` qui prend en argument une chaîne de caractères et renvoie `True` si elle est un palindrome, `False` sinon. (Il s'agit d'utiliser ce qu'on vient de voir sur les propriétés des chaînes de caractères et de ne pas recopier le DM précédent bien sûr.)

Exercice 15

1. Concevoir une fonction `somme_chiffres(n)` qui renvoie la somme des chiffres d'un entier positif `n`.
   → *Utiliser les fonctions de conversions* `str` *et* `int`.

2. Utiliser la fonction précédente pour écrire une fonction `somme_de_tous_les_chiffres(n)` qui renvoie la somme de tous les chiffres de tous les entiers inférieurs à `n`.

   On doit avoir `somme_de_tous_les_chiffres(14)==55` et `somme_de_tous_les_chiffres(100)==900`.

2 Reconstruction d'une chaîne de caractères

Une chaîne peut être construite à l'aide d'une variable cumulative de la manière suivante.

```
chaine = "exemple"
s = ""
for char in chaine~:
    s = s+char
```

Exercice 16

Quelle est la chaîne de caractères affichée par le script suivant ?

```
chaine = "abc"
s = ""
for char in chaine~:
    s = s+char+s
print(s)
```

Exercice 17

Écrire une fonction `str_filter(s, mask)` qui renvoie la chaîne composée des caractères de `s` à l'exception de ceux compris dans la chaîne `mask`. Exemple : l'appel `str_filter("un exemple", " e")` doit renvoyer `"unxmpl"`.

→ L'expression `c in m` est un booléen qui vaut `True` si la chaîne `c` est une sous-chaîne de `m`, et `False` sinon.

Exercice 18

Écrire une fonction `a_l_envers(s)` qui prend en argument une chaîne de caractères `s` et renvoie la chaîne composée des caractères de `s` lus de droite à gauche. Exemple : l'appel `a_l_envers("abcd")` doit renvoyer `"dcba"`.

Exercice 19

Le chiffrement de Vigenère consiste à décaler les lettres contenues dans un message selon une clef déterminée.

Prenons l'exemple du message `"lyceelangevinwallon"`, que l'on souhaite chiffrer avec la clef `"nsi"`.

Les décalages fournis par la clef ont pour valeur 13, 18 et 8. Le chiffrement s'effectue alors de la manière suivante :

- la lettre `"l"` est décalée de 13 rangs (`"y"`) ;
- la lettre `"y"` est décalée de 18 rangs (`"q"`) ;
- la lettre `"c"` est décalée de 8 rangs (`"k"`) ;
- la lettre `"e"` est décalée de 13 rangs (`"r"`) ;
- et ainsi de suite...

Le message codé est alors `"yqkrwtnfornqaoiydwa"`.

1. Concevoir une fonction `chiffre_vigenere(msg, key)` qui chiffre le message `msg` avec la clef `key`.
2. Concevoir une fonction `dechiffre_vigenere(msg, key)` qui déchiffre le message `msg`, connaissant la clef de chiffrement `key`.

3 Les fichiers texte

La lecture ou l'écriture de fichiers texte se fait en Python grâce à la fonction `open`.
Les paramètres de la fonction `open` sont :

- le nom d'un fichier situé dans le même répertoire que le fichier `.py` (ou son chemin relatif/absolu);
- la chaîne `"r"`, `"w"` ou `"a"`, selon que l'on souhaite un accès en lecture ou écriture ;
- le format d'encodage (par le mot-clef `encoding`).

La fonction `open` renvoie un objet de type `_io.TextIOWrapper` pour lequel sont définies un certain nombre de méthodes (fonctions) dont : `write`, `read` et `close`.

La méthode `close` permet de libérer la mémoire accordée à la lecture/écriture du fichier : il est recommandé de l'appeler de manière systématique lorsque la lecture ou l'écriture d'un fichier est terminée.

- Écriture d'un fichier texte au format UTF-8

  Le script ci-dessous crée un fichier texte nommé `data.txt` contenant la chaîne de caractère `s`.
  Si le fichier `data.txt` existe déjà, son contenu est irréversiblement écrasé.

```
s = "Une chaîne de caractères\nsur deux lignes"
writer = open("data.txt", "w", encoding="utf-8")
writer.write(s)
writer.close()
```

- Ajout d'un texte à la fin d'un fichier

  Il est possible d'ajouter du texte à la fin d'un fichier grâce au caractère `"a"` (*append* = ajouter à la fin).

```
s = "\nEncore du texte"
writer = open("data.txt", "a", encoding="utf-8")
writer.write(s)
writer.close()
```

- Lecture d'un fichier texte encodé en Latin-1

  Après exécution du script ci-dessous, le contenu du fichier `data.txt` est copié dans la chaîne de caractère `s`. Une fois que la méthode `read` a été appelée, l'objet `reader` n'est plus utilisable : tout autre appel de `read` renvoie une chaîne vide.

```
reader = open("data.txt", "r", encoding="latin-1")
s = reader.read()
reader.close()
print(s)
```

- Lecture ligne par ligne d'un fichier texte

  En mode lecture, l'objet `_io.TextIOWrapper` est itérable : à chaque tour de la boucle `for` dans le script suivant, la variable `line` prend successivement la valeur des lignes du fichier `data.txt`.

```
reader = open("data.txt", "r", encoding="utf-8")
for line in reader:
    print(line)
reader.close()
```

Exercice 20

Télécharger le fichier `exemple.txt`, qui a été encodé au format Latin-1, et écrire un script qui enregistre le contenu de ce fichier au format UTF-8 dans un fichier `exemple-utf8.txt`.