Cours : CHAINES DE CARACTERES :

Absence d'accents :

Un ordinateur ne sait manipuler que des nombres, ainsi si on souhaite manipuler du texte, on va associer à chaque caractère (lettre, chiffres, ponctuation...) un entier en choisissant une convention (par exemple que la lettre "A" est représentée par l'entier 65). On appelle ces conventions des encodages, et il en existe un grand nombre, qui ne sont pas forcément compatibles entre eux !

On utilisera l'encodage UTF-8 qui est le plus générique. Il permet de gérer de la même manière les caractères de toutes les langues. Certains langages de programmation, comme Java ou Python, sont capables de manipuler des textes encodés en UTF-8, mais ce n'est pas le cas de C ou C++ par exemple.

I- CHAINE DE CARACTERES :

Lire une ligne de texte :

Vous avez déjà vu comment en Python il était possible d'afficher du texte. Par exemple :

```
print("Le Corbeau et le Renard")
```

```
Le Corbeau et le Renard
```

Mais il est possible de faire beaucoup plus de choses que simplement afficher du texte.

Il est possible de stocker du texte dans une variable, afin de l'afficher plus tard.

```
vers = "Maitre Corbeau, sur un arbre perche,"
print(vers)
```

```
Maitre Corbeau, sur un arbre perche,
```

On peut aussi demander une ligne de texte à l'utilisateur

```
vers = input()
print(vers)
```

```
Tenait en son bec un fromage.
```

```
Tenait en son bec un fromage.
```

On parlera également de *chaîne de caractères* pour désigner une suite de caractères, possiblement sur plusieurs lignes.

Comparer deux chaines de caractères :

En Python, il est possible de comparer deux chaînes de caractères selon l'ordre alphabétique (appelé également ordre lexicographique).

```
ligne1 = "Maitre Renard, par l'odeur alleche,"
ligne2 = "Lui tint a peu pres ce langage :"
print("Selon l'ordre lexicographique, ", end = "")
if ligne1 < ligne2:
   print("la ligne 1 est située avant la ligne 2")
if ligne1 == ligne2:
   print("la ligne 1 est égale à la ligne 2")
if ligne1 > ligne2:
   print("la ligne 1 est située après la ligne 2")
```

```
Selon l'ordre lexicographique, la ligne 1 est située après la ligne 2
```

On peut donc comparer directement deux chaînes de caractères et tous les opérateurs de comparaisons sont disponibles, c'est-à-dire `<`, `<=`, `==`, `!=`, `=>` et `>`.

Calculer la longueur d'une chaine de caractères :

Quand on nous donne une chaîne de caractères on a parfois besoin de connaître sa longueur, c'est-à-dire le nombre de caractères qu'elle contient. En Python, il existe une fonction pour cela, comme on peut le voir sur le code suivant :

```
vers = "Que vous etes joli! Que vous me semblez beau!"
longueur = len(vers)
print(longueur)
```

```
45
```

Rapidité du calcul d'une longueur

Calculer la longueur d'une chaîne de caractère est une opération **rapide** en Python, qui se fait en temps constant : quelle que soit la longueur de la chaîne, cela prend toujours le même temps.

MOTS, lire un mot individuellement :

Vous avez vu comment lire une ligne complète, mais parfois on veut simplement lire un seul mot, afin de pouvoir le manipuler tout seul, indépendamment de la ligne. On définit de manière naturelle un mot comme une suite de lettres ne contenant aucun espace, que ce soit un véritable espace, un retour à la ligne ou une tabulation. Imaginons, par exemple, qu'on ait à lire un entier *nb* et un mot (de longueur plus petite que 100), tout deux sur la même ligne :

```
5 TRUC
```

et à afficher ensuite *nb* fois le mot donné :

```
TRUC TRUC TRUC TRUC TRUC
```

Avant de voir comment résoudre cet exercice, il faut comprendre qu'en Python on ne peut pas faire autrement que de lire toute la ligne à la fois. Il faut donc être capable d'extraire les mots ou les nombres situés sur une ligne. Imaginons qu'on souhaite lire la ligne

```
Cette ligne comporte plusieurs mots
```

puis afficher les premier et quatrième mots. On va utiliser le code suivant :

```
# Lit toute la ligne
ligne = input()
# Coupe la ligne en un tableau de mots
mots = ligne.split(" ")
# Affiche la ligne puis les deux mots
print(ligne)
print(mots[0])
print(mots[3])
```

```
Cette ligne comporte plusieurs mots
Cette
plusieurs
```

La fonction `split` permet, à partir d'une chaîne de caractères, de la couper en petits morceaux, chaque morceau étant séparé des autres par une espace. La fonction renvoie un tableau de mots, auxquels on peut alors accéder de manière classique.

Il est aussi possible de lire la ligne de texte et de la couper en mot, en une seule ligne de code :

```
mots = input().split(" ")
```

Un programme possible pour résoudre le problème expliqué ci-dessus est le suivant :

```
elements = input().split(" ")
nb = int(elements[0])
mot = elements[1]
for id in range(nb):
    print(mot, end = ' ')
```

On a donc utilisé la fonction `split()` que nous avions présenté et nous avons converti le premier mot en un entier, car nous savons que ce mot représente un nombre. Si nous ne l'avions pas fait, nous aurions eu l'erreur suivante :

```
TypeError: 'str' object cannot be interpreted as an integer
```

Lire plusieurs choses sur la même ligne écriture simplifiée :

Lire plusieurs mots

On cherche à lire deux mots sur la même ligne, un nom de pays et un nom de ville, puis afficher un petit texte. En utilisant le code `input().split(" ")` que vous avez déjà vu on obtient le programme suivant :

```
mots =  input().split(" ")
pays = mots[0]
ville = mots[1]
print("Vous habitez à {} ({})".format(ville, pays))
```

```
France Paris
```

```
Vous habitez à Paris (France)
```

Ce code fonctionne très bien, mais il est possible de l'écrire de manière plus courte et plus agréable à lire :

```
pays, ville =  input().split(" ")
print("Vous habitez à {} ({})".format(ville, pays))
```

Ainsi, au lieu de créer un tableau `mots` contenant deux éléments, on affecte directement la valeur de `input().split(" ")` aux deux variables `pays` et `ville`.

Lire plusieurs entiers

De manière similaire, si on chercher à lire deux entiers sur la même ligne on peut utiliser `input().split(' ')` puis qu'on convertir chaque élément lu en un entier. Par exemple :

```
mots =  input().split(" ")
longueur = int(mots[0])
largeur = int(mots[1])
print(longueur * largeur)
```

```
5 10
```

```
50
```

Il est possible de faire plus court, sans devoir convertir les mots un par un en entiers !

```
longueur, largeur =  map(int, input().split(" "))
print(longueur * largeur)
```

```
5 10
```

```
50
```

La fonction `map()` va appeler la fonction `int()` (son premier argument) sur chacun des élements du tableau donné comme second argument, et va retourner le tableau contenant tous les entiers. On affecte alors les deux éléments de ce tableau aux variables `longueur` et `largeur`.

II- CARACTERES : ,Dans la suite, nous allons voir comment manipuler des caractères seuls ou au sein de chaines de caractères.

Accéder aux caractères d'une chaîne et les afficher :

Jusqu'à présent nous avons vu comment manipuler des chaînes de caractères dans leur ensemble, mais il est tout à fait possible de manipuler chacun des caractères de la chaîne. Par exemple, voici comment on peut lire et afficher le premier et le sixième caractère d'une chaîne de caractères :

```
prenom = "TINTIN"
print(prenom[0])
print(prenom[5])
```

```
T
N
```

On remarque que pour les chaînes de caractères les indices démarrent à 0. Ainsi, si *longueur* est la longueur de la chaîne de caractères, alors il est possible d'accéder aux caractères d'indices 0, 1, 2, ..., *longueur*-1. C'est donc pareil que pour les tableaux.

Comparer deux caractères :

Vous avez déjà vu comment comparer deux chaînes de caractères selon l'ordre alphabétique, et il est aussi possible de comparer uniquement deux caractères :

```
nom = "HADDOCK"
if nom[0] < nom[5]:
   print("La lettre d'indice 0 (H) est avant la lettre d'indice 5 (C) dans l'alphabet")
if nom[2] == nom[3]:
   print("La lettre d'indice 2 (D) est égale à la lettre d'indice 3 (D) dans l'alphabet")
if nom[0] > nom[5]:
  print("La lettre d'indice 0 (H) est après la lettre d'indice 5 (C) dans l'alphabet")
```

```
La lettre d'indice 2 (D) est égale à la lettre d'indice 3 (D) dans l'alphabet
La lettre d'indice 0 (H) est après la lettre d'indice 5 (C) dans l'alphabet
```

On peut donc comparer directement deux caractères et tous les opérateurs de comparaisons sont disponibles, c'est-à-dire `<`, `<=`, `==`, `!=`, `=>` et `>`.

Il est aussi possible de comparer un caractère d'une chaîne à un caractère directement indiqué dans le code, par exemple :

```
nom = "DI GORGONZOLA"
if nom[0] == "D":
   print("Le nom commence par la lettre D")
if nom[3] <= "M":
   print("La quatrième lettre (la lettre G) est située avant la lettre M dans l'alphabet")
if nom[2] == " ":
   print("La troisième lettre est un espace !")
```

```
Le nom commence par la lettre D
La quatrième lettre (la lettre G) est située avant la lettre M dans l'alphabet
La troisième lettre est un espace !
```

En Python, un caractère est simplement une chaîne de caractère de longueur 1, on utilise donc des guillemets doubles, comme pour les chaînes de caractères.

Déclarer et lire un caractère :

Jusqu'à présent, vous avez vu comment manipuler des caractères au sein de chaînes complètes, mais il est aussi possible de déclarer un caractère indépendamment d'une chaîne de caractères :

```
lettre = "T"
```

En Python, il n'existe pas de notion de caractère individuel, il s'agit donc simplement d'une chaîne de caractère de longueur 1 !.

Il est aussi possible de demander un caractère à l'utilisateur et d'afficher un caractère :

```
lettre = input()
print("La lettre donnée par l'utilisateur est :")
print(lettre)
print("Une autre lettre :")
print("W")
```

```
↳ E
```

```
↳ La lettre donnée par l'utilisateur est :
E
Une autre lettre :
W
```

En Python, encore une fois, tout se fait comme pour une chaîne de caractères classique.

Modifier une chaine existante :

Il est parfois nécessaire de modifier les caractères d'une chaîne de caractère. En Python les chaînes de caractères ne sont pas modifiables, ceci pour des raisons qu'il serait trop long d'expliquer ici. Il faut donc passer par d'autres objets que des chaînes de caractères.

En partant d'une chaîne de caractère existante

```
texte = "Exemple de texte"
```

l'objet par lequel on va passer, pour pouvoir faire des modifications, est un véritable tableau de caractères. Il nous faut donc d'abord convertir une chaîne de caractères en tableau de caractères. Pour cela, il faut utiliser le code suivant :

```
caracteres = list(texte)
```

Ensuite, on peut le modifier comme on le ferait de manière habituelle. Par exemple, pour changer le premier caractère on fait

```
caracteres[0] = "X"
```

On peut alors convertir le tableau de caractères en une chaîne de caractères, par exemple pour pouvoir facilement l'afficher ensuite

```
texte = "".join(caracteres)
```

De manière plus détaillée, cette ligne demande à construire une nouvelle chaîne en collant bout à bout tous les caractères du tableau `caracteres`.

Voilà ce que donne un programme complet :

```
texte = "Exemple de texte"
caracteres = list(texte)
caracteres[8] = "X"
caracteres[9] = "X"
texte = "".join(caracteres)
print(texte)
```

```
↳ Exemple XX texte
```