Les chaînes de caractères

Rappel : Déclaration des variables

```
Variables :
  Prix : Réel
  Nom : Chaîne
```

Une chaîne de caractères est une suite ordonnée de caractères. En algorithmique, une chaîne de caractères est considérée comme une variable scalaire. Par contre, en langage C, une chaîne est vue comme un tableau de caractères alors qu'en Java, une chaîne est un objet. En Python, les chaînes de caractères sont classées dans les séquences tout comme les listes et les tuples (*les tuples et les chaînes sont des **séquences immuables**, c.à.d. qu'elles ne peuvent pas être modifiées*).

1- "Une chaîne de caractères est une suite finie de caractères."

Exemples :

- La chaîne de caractères "`Baobab`" est constituée des ***caractères*** "`B`", "`a`", "`o`", "`b`", "`a`" et "`b`".
- La chaîne de caractères "`123`" n'est constituée que de ***chiffres***, mais en aucun cas il ne faut la confondre avec la grandeur numérique `123`.
- La chaîne de caractères "" représente une ***chaîne*** de caractères ***vide***.
- La chaîne de caractères " " est une chaîne de caractères ne contenant qu'un seul caractère qui est ici le caractère ***espace*** (*à ne pas confondre avec la chaîne de caractères vide*).

Les chaînes sont constituées de caractères. Chaque caractère (*qui n'est autre qu'un symbole*) se trouve à une position donnée dans la chaîne (*on parle alors de **rang***). Le rang d'un caractère est à la chaîne ce que l'indice d'un élément est au tableau. En algorithmique, le premier caractère se trouve au rang `1`, le deuxième caractère se trouve au rang `2` et ainsi de suite... En langage Python, le premier caractère d'une chaîne se trouve au rang `0`, le deuxième caractère se trouve au rang `1` et ainsi de suite...

Exemple : Chaîne "`Baobab`"

Rangs (Python)	*0*	*1*	*2*	*3*	*4*	*5*
Caractères	`"B"`	`"a"`	`"o"`	`"b"`	`"a"`	`"b"`
Rangs (Algo.)	*1*	*2*	*3*	*4*	*5*	*6*

Un caractère peut-être :

- une lettre (*caractère alphabétique – la distinction est faite entre les lettres minuscules et les lettres majuscules*),
- un chiffre (*caractère numérique*),
- un signe de ponctuation,
- ou tout autre symbole (*comme l'étoile, le tiret, le "slash"...*).

Dans un système informatique, à chaque caractère est associé une valeur numérique : son **code ASCII** (*American Standard Code for Information Interchange*). L'ensemble des codes est recensé dans une table nommée "***table des codes ASCII***". Quand on stocke un caractère en mémoire (*dans une variable*), on mémorise en réalité son code ASCII. Un code ASCII est codé sur un octet (*huit bits*). La table des codes ASCII est donnée à la fin de ce document.

On peut remarquer que les lettres minuscules se suivent dans la table des codes ASCII (*tout comme les lettres majuscules et les chiffres*). Ces trois "*classes*" de caractères forment trois plages de codes ASCII (*trois domaines de valeurs*) qu'il sera possible d'exploiter pour effectuer des conversions.

2- Relation d'ordre

Exemple : Comparaison de caractères
`"B" > "A"` car le code ASCII du caractère `"B"` (*66 en base 10*) est supérieur au code ASCII du caractère `"A"` (*65 en base 10*).

Pour comparer deux chaînes de caractères, on compare les caractères de même rang dans les deux chaînes en commençant par le premier caractère de chaque chaîne (*le premier caractère de la première chaîne est comparé au premier caractère de la seconde chaîne, le deuxième caractère de la première chaîne est comparé au deuxième caractère de la seconde chaîne, et ainsi de suite…*).

Exemples : Comparaison de deux chaînes

- `"baobab" < "sapin"` car le code ASCII de `"b"` est inférieur au code ASCII de `"s"`.
- `"baobab" > "banania"` car le code ASCII de `"o"` est supérieur au code ASCII de `"n"` (*la comparaison ne peut pas se faire sur les deux premiers caractères car ils sont identiques*).
- `"1999" > "1998"` car le code ASCII de `"9"` est supérieur au code ASCII de `"8"` (*la comparaison ne peut pas se faire sur les trois premiers caractères car ils sont identiques*). Attention, ici ce ne sont pas des valeurs numériques qui sont comparées, mais bien des caractères.
- `"333" > "1230"` car le code ASCII de `"3"` est supérieur au code ASCII de `"1"`.
- `"333" < "3330"` car la seconde chaîne a une longueur supérieure à celle de la première (*la comparaison ne peut pas se faire sur les trois premiers caractères car ils sont identiques*).
- `"Baobab" < "baobab"` car le code ASCII de `"b"` est supérieur au code ASCII de `"B"`.

Exercice d'application : Compléter le tableau ci-dessous en indiquant la relation d'ordre (<,=,>) ainsi que le nombre de caractères concernés par la comparaison.

`"python"`	`>`	`"Python"`	Le code ASCII de `"p"` est supérieur au code ASCII de `"P"`.
`"Bonsoir"`		`"Bonjour"`	
`"Hola"`		`"hola"`	
`"Ohé"`		`"Eho"`	
`"BTS1IG"`		`"BTS2IG"`	
`"27/10/00"`		`"27/10/99"`	
`"C & Java"`		`"C | Java"`	
`"> 15"`		`"< 15"`	
`"toi"`		`"toi"`	
`"5170"`		`"517"`	
`"318"`		`"1273"`	

3- Concaténation de chaînes de caractères

Le but de la concaténation est de créer des chaînes de caractères en juxtaposant deux (*ou plus*) chaînes de caractères. En algorithmique, l'opérateur de concaténation est représenté par **+** ou par **//**. Les deux notations sont possibles (*la première notation est utilisée en Java, Python ou encore en JavaScript*). Nous préférerons tout de même la seconde notation qui nous permettra lors de la lecture d'un algorithme, de bien voir qu'une instruction porte sur des opérandes de type `Chaîne` et non de type `Entier` ou `Réel`. L'opération de concaténation peut-être appliquée à une même variable (*tout comme pour les variables dites "de cumul" ou les variables "compteurs"*).

Exemple : Concaténation de deux chaînes

```
Algo Concaténations
Variable :
  catégorie : Chaîne
Début
  catégorie <- "développeur"
  catégorie <- catégorie // "s"
  Afficher(catégorie)                  # Affiche : développeurs
  catégorie <- "des " // catégorie
  Afficher(catégorie)                  # Affiche : des développeurs
Fin
```

4- Manipulation de chaînes de caractères

4.1. Copie d'un extrait

Algo.	**SSCHAINE**(*chaîne*,*position*,*nombre*):Chaîne
Python	*chaîne*[*position*] *chaîne*[*position_début*:*position_fin_exclue*]

Le rôle de la fonction `SSCHAINE()` est de retourner une "*sous-chaîne*" (*copie d'un extrait de la chaîne passée en premier paramètre*) de *`nombre`* caractères de la chaîne *`chaîne`* à partir du caractère qui se trouve en position *`position`* (*rappel : le premier caractère d'une chaîne de caractères se trouve en position `1` en algorithmique et en position `0` en langage Python*). En Python, une chaîne étant une séquence, l'utilisation des crochets permet d'accéder à un extrait constitué d'un ou plusieurs caractères.

4.2. Nombre de caractères

Algo.	**LONGUEUR**(*chaîne*):Entier
Python	**len**(*chaîne*)

La fonction `LONGUEUR()` a pour rôle de calculer et de retourner le nombre de caractères présents dans la chaîne de caractères passée en paramètre. En Python, on peut remarquer que la fonction est identique à celle utilisée avec les listes et les tuples. En fait, la fonction `len()` permet de connaître la longueur (*nombre d'éléments*) d'une séquence... que ce soit une liste, un tuple ou une chaîne.

4.3. Recherche d'un caractère ou d'une sous-chaîne

Algo	**RANG**(*chaîne*,*sous-chaîne*,*position*):Entier
Python	*chaîne*.**index**(*sous-chaîne*,*position*) *chaîne*.**find**(*sous-chaîne*,*position*)

La fonction `RANG()` recherche et retourne la première occurrence de la *`sous-chaîne`* dans la *`chaîne`*. La recherche commence à partir de *`position`*. Cette fonction retourne la position du premier caractère de la *`sous-chaîne`*, ou la valeur `0` si la "*`sous-chaîne`*" n'existe pas dans la *`chaîne`* (*à partir de `position`*). En Python, si aucune occurrence de la sous-chaîne n'est trouvée, la méthode `find()` retourne la valeur `-1` alors que la méthode `index()` génère une exception (*`ValueError`*). *Le concept de exceptions sera étudié dans un autre chapitre.*

4.4. Conversion en code ASCII

Algo	**CODE**(*caractère*):Entier
Python	**ord**(*caractère*)

La fonction `CODE()` retourne le code ASCII du caractère passé en paramètre.

4.5. Conversion en caractère

Algo	**CAR**(*code_ascii*):Caractère
Python	**chr**(*code_ascii*)

La fonction `CAR()` retourne le caractère dont le code ASCII est passé en paramètre.

4.6. Conversion en grandeur numérique

Algo	**CVNOMBRE**(*chaîne*):Entier **CVNOMBRE**(*chaîne*):Réel
Python	**int**(*chaîne*) **float**(*chaîne*)

La fonction `CVNOMBRE()` a pour rôle de convertir (*CV pour ConVersion*) la chaîne de caractères passée en paramètre, en grandeur numérique si la suite de caractères représente un nombre, puis de retourner la grandeur numérique.

Pour stocker en mémoire la valeur numérique `65534` (`0xFFFE`), il faut deux octets (*puisque qu'avec un seul octet on peut coder une valeur comprise entre 0 et 255 inclus*). Par contre, pour stocker la chaîne de caractères `"65534"`, il faut cinq octets en mémoire puisque cette chaîne contient cinq caractères, et qu'il faut un octet par caractère.

Exemple : Occupation mémoire

```
Variables :
  valeur_chaîne : Chaîne
  valeur_entier : Entier
Début
  valeur_chaîne <- "65534"
  valeur_entier <- 65534
  ...
Fin
```

valeur_chaîne

0x36	0x35	0x35	0x33	0x34

valeur_chaîne

0xFF	0xFE

Dans la réalité, les nombres entiers (*ainsi que les nombres réels*) sont tous codés sur le même nombre d'octets. Ce nombre d'octets dépend du système d'exploitation et du microprocesseur (*pour les entiers, on parle de "mot machine"*) et non de la valeur à mémoriser.

4.7. Conversion en chaîne de caractères

Algo	**CVCHAINE**(*entier*):Chaîne **CVCHAINE**(*réel*):Chaîne
Python	**str**(*entier*) **str**(*réel*)

La fonction `CVCHAINE()` a pour rôle de convertir en chaîne de caractères une valeur numérique passée en paramètre, puis de retourner la chaîne résultante.

Exercice d'application : Que retourne chacune des instructions ci-dessous ? Traduire chaque instruction en langage Python.

Algo.	***Algo.***	***Python***	***Python***
`LONGUEUR("Agenda")`			
`SSCHAINE("Almanach",2,3)`			
`SSCHAINE("Almanach",5,1)`			
`SSCHAINE("Almanach",5)`			

RANG("Almanach","an",1)			
RANG("Almanach","kan",1)			
RANG("Almanach","a")			
RANG("Almanach","a",4)			
RANG("Almanach","a",5)			
CODE("Z")			
CAR(118)			

Pour conclure...

En Python, il existe de nombreuses fonctions/méthodes de manipulation des chaînes de caractères comme qui n'ont pas été détaillées dans ce document. Certaines sont applicables aux séquences en général et d'autres sont spécifiques aux chaînes. Elles peuvent être classées en plusieurs catégories :

- Modification de la casse : lower(), `upper()`, `swapcase()`, `capitalize()`, `title()`
- Nombre d'occurrences : `count()`
- Suppression de caractères : `lstrip()`, `rstrip()`, `strip()`
- Alignement : `ljust()`, `rjust()`, `center()`
- Découper et reconstituer : `split()`, `join()`
- Remplacement : `replace()`
- Recherche inversée : `rindex()`, `rfind()`
- Valeur minimale et valeur maximale : `min()`, `max()`
- Classe des caractères : `isalnum()`, `isalpha()`, `isdigit()`, `islower()`, `isupper()`, `isspace()`, `istitle()`

Pour vous aider à utiliser au mieux toutes ces fonctions, il existe la documentation Python que vous pouvez consulter sur Internet, la commande `pydoc` ainsi que l'interpréteur en mode interactif.