LES CHAÎNES DE CARACTÈRES

Point de cours :

Prise en main

On considère l'algorithme suivant :

```
DEBUT
  n ← 5
  m ← 'bonjour'
  p ← '10'
  m ← m + p
FIN
```

Ici les variables *n* et *p* n'ont pas le même statut. *n* contient une valeur numérique alors que *p* contient une valeur alphanumérique. Après exécution de l'algorithme, la variable *m* contiendra la chaîne de caractères '*bonjour*10'. L'instruction $n \leftarrow n + p$ n'aurait aucun sens.

CE QU'IL FAUT SAVOIR

Une chaîne de caractères est un regroupement de caractères issus d'un alphabet.

Remarque

Ne pas confondre une chaîne de caractères et un tableau. Comme les tableaux, les chaînes de caractères ont une longueur : le nombre de caractères qui les composent. Contrairement aux tableaux, il n'est pas possible de modifier partiellement les caractères d'une chaîne déjà existante directement, même si les caractères peuvent être considérés individuellement grâce à l'indice.

Délimitation d'une chaîne de caractères

On utilise les guillemets en pseudo-code, les apostrophes ou les guillemets en Python :

Pseudo-code	Python
message1 ← « "Oui", a-t-elle répondu »	message1 = '"Oui", a-t-elle répondu,'
message2 ← « j'aime beaucoup l'algo ! »	message2 = "j'aime beaucoup l'algo !"
Afficher message1, message2	**print**(message1, message2)

L'algorithme affiche : Oui , a-t-elle répondu, j'aime beaucoup l'algo !

Concaténation de deux chaînes de caractères

Pour joindre deux chaînes de caractères, on utilise l'opérateur '+'

Pseudo-code	Python
message3 ← message1 + message2	message3 = message1 + message2
Afficher message3	**print**(message3)

La variable alphanumérique message3 contient "Oui", a-t-elle répondu, j'aime beaucoup l'algo !

Accès à des éléments d'une chaîne de caractères

Python permet une manipulation aisée des chaînes de caractères : on peut utiliser l'index vu dans la partie sur les tableaux, ou encore tester si une lettre apparaît dans une chaîne de caractères.

Pour accéder à un caractère en particulier ou à un regroupement de caractères, on utilise le nom de la variable qui contient la chaîne de caractères et l'indice des caractères concernés dans la chaîne. Les indices commencent à 0.

```
Python
c="Bonjour tout le monde !"
print(len(c))
print(c[0])
print(c[2 : 15])
```

La chaîne de caractères est de longueur 23. $c[0] = "B"$ et $c[2:15] = "njour\ tout\ le"$

```
Python
>>>c="Bonjour tout le monde !"
>>>c.index('m')   #position d'un caractère
17
>>>c.index('jour')   #position d'une sous-chaîne
3
>>> 'b' in c   # permet de tester si une sous-chaîne appartient à la chaîne
False
#Attention, la casse des caractères est importante
>>> 'B' in c
True
```

Modification d'une chaîne de caractères

Pour modifier une chaîne de caractères déjà existante, il faudra créer une nouvelle variable. Par exemple :

```
Python
m = "bonjour"
M = "B"+ m[1:]
print (M)
```

Cet algorithme affichera le mot « Bonjour ».

Exemple 1 : Création d'une chaîne à partir d'une chaîne déjà existante

L'algorithme suivant recopie un mot saisi au clavier par l'utilisateur en insérant entre chaque caractère des dièses #.

Par exemple, si l'utilisateur saisit "Algorithmique", l'algorithme affichera : "A#l#g#o#r#i#t#h#m#i#q#u#e"

```
Algorithme : plagiat modifié
Variables numériques : i
Variables alphanumériques : mot
DEBUT
   Saisir mot
   nouveau_mot← "" #Le nouveau mot ne contient aucun caractère au début
   Pour i allant de 0 à longueur(mot) – 2
      nouveau_mot← nouveau_mot + mot[i] + "#"
   Fin pour
   nouveau_mot← nouveau_mot + mot[longueur(mot) – 1]
   Afficher nouveau_mot
FIN
```

Python

```
mot=input("Veuillez saisir un mot")
nouveau_mot=""
for i in range (len(mot) -1):
    nouveau_mot=nouveau_mot+mot[i]+"#"
nouveau_mot=nouveau_mot+mot[len(mot)-1]
print(nouveau_mot)
```

Exemple 2 : Nombre de mots dans une chaîne de caractères

On souhaite écrire un algorithme qui, à partir d'une chaîne de caractères saisie au clavier par l'utilisateur, affiche le nombre de mots de la phrase. On considère que l'utilisateur saisit des chaînes de caractères sans ponctuation.

Idée : compter le nombre d'espaces dans la phrase saisie.

Algorithme : nombre de mots

Variables numériques : *i*, *cpt*
Variables alphanumériques : *chaine*
DEBUT
  Afficher "Veuillez saisir une phrase sans ponctuation, commençant et se terminant par une lettre"
  Saisir *chaine*
  cpt ← 1
  Pour *i* **allant de** 0 **à** *longueur*(*chaine*) − 1
    Si *chaine*[*i*] = " " **alors**
      cpt ← *cpt* + 1
  Fin Pour
  Afficher *cpt*
FIN

Python

```
chaine=input("Veuillez saisir une phrase sans ponctuation, commençant et se terminant par une lettre")
cpt=1
for i in range (len(chaine)) :
    if chaine[i]==" " :
        cpt=cpt+1
print(cpt)
```

Transtypage

Il arrive que l'on ait besoin de transformer une chaîne de caractères en nombre et réciproquement. Cette action s'appelle une opération de transtypage.

En Python :

- On peut transformer un nombre en chaîne de caractères en utilisant la fonction de transtypage **str(chaine)**.
- On peut transformer une chaîne de caractères en nombre avec la commande **int(chaine)** ou **float(chaine).**

Python

```
>>> str(52) # transforme le nombre 52 en la chaîne de caractères '52'.
>>> int('52') #transforme la chaîne de caractères '52' en le nombre entier 52.
>>> float("3.14") #transforme la chaîne de caractères "3.14" en le nombre réel 3.14.
```

Remarque

Lors du transtypage d'une chaîne de caractères en nombre, il convient de s'assurer que le contenu de la chaîne de caractères convienne au type que l'on souhaite lui donner. Par exemple, le transtypage de la chaîne "4.15" en nombre entier engendrera une erreur.

Exemple 3 : Conversion décimal à binaire

On souhaite écrire un algorithme dans lequel l'utilisateur saisit un nombre entier écrit en base décimale et où l'algorithme affiche ce nombre écrit en base 2.

Algorithme : conversion base 10 à base 2

```
Variables numériques : i, n
Variable alphanumérique : nombre
DEBUT
  Afficher "Veuillez saisir un nombre entier en base 10"
  Saisir n
  r ← n
  nombre ← ""
  Tant que PartieEntiere(r / 2) ≠ 0
    nombre ← str(r%2) + nombre
    r ← PartieEntiere(r / 2)
  Fin Tant que
  nombre ← str(r) + nombre
  Afficher int(nombre)
FIN
```

Python

```
n=int(input())
r=n
nombre=""
while(r//2)!=0:
    nombre=str(r%2)+nombre
    r=r//2
nombre=str(r)+nombre
print(int(nombre))
```

La notation str() ci-contre traduit la transformation d'un nombre en chaîne de caractères (et inversement pour int()).

Applications :

Application 1 : voyelle ou consonne ?

1. Écrire un algorithme qui demande à l'utilisateur de saisir une lettre minuscule puis affiche si la lettre saisie est une consonne ou une voyelle.

Remarque

On pourra utiliser une variable alphanumérique voyelle='aeiouy'

2. Modifier l'algorithme précédent pour qu'il affiche également si le caractère saisi est une majuscule ou une minuscule.

Application 2 : réécriture

Écrire un algorithme qui transforme une chaîne de caractères écrite en minuscules (sans accents et sans ponctuation), en une chaîne identique mais écrite en majuscules.

Application 3 : dans une entreprise

Dans une entreprise, chaque employé possède un code personnel alphanumérique composé de huit caractères.

- Les deux premiers représentent l'année d'embauche.
- Les quatre suivants le numéro d'embauche.
- Le septième le titre de l'employé :
  - 1 pour Madame
  - 2 pour Monsieur
- Le huitième désigne le domaine dans lequel travaille l'employé :
  - 0 pour la direction
  - 1 pour le secrétariat
  - 2 pour la gestion
  - 3 pour l'informatique
  - 4 pour la communication
  - 5 pour l'entretien
  - 6 pour la fabrication

Écrire un programme qui demande à l'utilisateur son nom, prénom et son code personnel et qui renvoie une phrase donnant sa situation.

Exemple

L'utilisateur saisit les nom et prénom DURAND Estelle, et le code "05201413".

On veut obtenir la phrase suivante :
« Madame Estelle DURAND travaille au service informatique. Elle travaille dans l'entreprise depuis 2005 et son numéro d'embauche est le 2014. »

Remarque

Le programme peut prendre en compte le fait que la personne peut avoir été recrutée avant ou après l'an 2000. Par exemple : si les deux premiers chiffres sont 12, il semble difficile de penser que la personne a été recrutée en 1912...

Application 4 : conversion hexadécimal à décimal

Écrire un algorithme qui demande à l'utilisateur de saisir, sous forme d'une chaîne de caractères, un nombre entier écrit en base hexadécimale puis qui affiche ce nombre en base décimale.

Remarque

On pourra utiliser la variable suivante :

$H =$ "0123456789*ABCDEF*"

Application 5 : conversion binaire à hexadécimal

Écrire un algorithme qui demande à l'utilisateur de saisir un nombre entier en base binaire puis qui affiche ce nombre en base hexadécimale sous forme d'une chaîne de caractères.

1. Écrire un algorithme qui décompose une chaîne de caractères constituée de '0' et de '1' en groupes de 4 caractères en partant de la droite.

Remarques

- On pourra créer un tableau contenant des chaînes de caractères de longueur 4.
- On ajoutera les '0' nécessaires pour que le dernier groupe formé contienne bien 4 caractères.

2. Modifier l'algorithme précédent pour qu'il convertisse le nombre en base hexadécimale.

Remarque

On pourra utiliser les deux tableaux suivants :

$H =$ ["0","1","2","3","4","5","6","7","8","9","*A*","*B*","*C*","*D*","*E*","*F*"]

$B =$ ["0000","0001","0010","0011","0100","0101","0110","0111",
"1000","1001","1010","1011","1100","1101","1110","1111"]

ainsi que la commande index qui permet de connaître le premier indice d'apparition d'une valeur donnée dans un tableau.