TD 7 : Manipulation de chaines

Yann Salmon

10 – 19 janvier 2017

1 Manipulation de chaines

1.1 Opérations de base

En Python, les chaines de caractères sont des objets immuables (c'est également le cas dans les versions récentes de OCaml).

On peut créer une chaine en écrivant ses caractères dans l'ordre ; ces caractères doivent être entourés par des guillemets (").

Question 1.1. Dans la console Python, effectuez `s = "Travailleurs de tous les pays, unissez-vous !"`. Puis demandez quel est le type de `s` en évaluant `type(s)`.

En Python, on peut obtenir des informations à propos d'une chaine, et notamment lire ses caractères, avec les mêmes symboles d'opérations que pour les tableaux.

Question 1.2. Évaluez successivement

```
len(s)
s[0]
s[1]
s[3:6]
"topinambour"[1]
len("")
```

Question 1.3. Que va-t-il se passer si vous évaluez `len("s")` ?

On peut créer une nouvelle chaine en concaténant (mettant bout à bout) deux chaines existantes.

Question 1.4. Évaluez `s + " (1848)"`. Évaluez `"3" + "2"`.

1.2 Algorithmes classiques

La plupart des algorithmes d'analyse de tableaux se transposent aux chaines de caractères.

Question 1.5. Écrivez une fonction `compte` qui étant donné une chaine et un caractère, renvoie le nombre de fois que ce caractère apparait dans la chaine.

1.3 Recherche de facteur

On dit du mot u qu'il est un facteur du mot v si les lettres formant u apparaissent en bloc au sein du mot v : ainsi « top », « nam », « nambo » et « our » sont des facteurs de « topinambour ».

Le problème de la recherche de facteur consiste à déterminer si un mot apparait dans un texte, et à quel endroit. Nous allons voir un algorithme naïf pour résoudre ce problème.

Question 1.6. Écrivez la spécification et le code Python d'une fonction `facteur_ici` qui étant donné une chaine h, une chaine n et une position i dans h, indique si la chaine h contient la chaine n comme facteur **à la position** i.

Question 1.7. Écrivez la spécification et le code Python d'une fonction `facteur` qui étant donné une chaine h et une chaine n dans h, indique si la chaine h contient la chaine n comme facteur.

Question 1.8. Démontrez la terminaison et la correction de ces fonctions.

Question 1.9. Quelle est la complexité temporelle de la fonction `facteur` en fonction de la longueur de h et de celle de n ?

Dans le cours d'option informatique deuxième année (MP/MP*), l'utilisation des automates permettra de résoudre ce problème avec une complexité bien meilleure, surtout quand on voudra chercher une catégorie de mots (par exemple : trouver dans h les mots qui se terminent par « ment »).

2 Codage des caractères

2.1 Lettres usuelles et chiffres

Chaque caractère est représenté par un entier. Les fonctions **ord** et **chr** permettent de manipuler ces représentations.

Question 2.1. Évaluez `ord("a")`, `ord("b")`, `ord("A")`, `ord("B")`, `ord("0")`. Évaluez `chr(32)`, `chr(64)`.

Question 2.2. À l'aide de la fonction **chr**, écrivez une fonction qui étant donné un entier naturel renvoie son écriture en base 10 sous forme de chaine de caractères.

Question 2.3. À l'aide de la fonction **ord**, écrivez une fonction qui étant donné une chaine de caractères composées de chiffres, renvoie l'entier ainsi représenté en base 10. Que fait votre fonction si on l'appelle avec une chaine qui comprend le caractère `"a"` ?

Important. Python dispose déjà des deux fonctions ci-dessus : il s'agit de **str** et **int**.

2.2 Caractères spéciaux

Différents éléments de mise en forme de texte sont représentés par des caractères spéciaux. Ainsi, la tabulation est représentée par le caractère spécial de code 9.

Question 2.4. Évaluez successivement

```
s = "toto" + chr(9) + "tata"
len(s)
print(s)
s
s[4]
```

Le passage à la ligne est également représenté par des caractères. Le drame, c'est que plusieurs conventions existent. Sous Linux, chaque ligne est terminée par le caractère de code 10.

Question 2.5. Reprenez la question précédente en remplaçant le `chr(9)` par un `chr(10)`.

2.3 Séparation de lignes

Question 2.6. Écrivez une fonction `separer` qui étant donné une chaine t et un caractère de séparation *sep*, renvoie la liste des chaines obtenues en coupant t à chaque fois que *sep*. Ainsi `separer("tati", "t")` renvoie la liste `["", "a", "i"]`.

Cette fonction peut être particularisée pour séparer un texte ligne par ligne en utilisant le caractère 10 comme séparateur.

Important. Python dispose déjà de la fonction ci-dessus : t`.split(`*sep*`)` produit le résultat voulu. Pour la séparation de lignes, t`.splitlines()` a l'avantage de ne pas donner la ligne finale si elle est vide.