TD5. Séquences de caractères

Bases de la Programmation Impérative – Chapitre 2 : iterations
version du 25 décembre 2023

Ce TD a pour objectif de prendre conscience que la notion de complexité a aussi du sens pour l'utilisation mémoire.

Exercice 1 : séquence infinie de caractères

On considère l'analyse d'une séquence de caractères, potentiellement très, très, très longue. On dispose d'une fonction `recupere_prochain_caractere()` qui renvoie le prochain caractère non traité de la séquence. On considère que le caractère `@` indique la fin de la séquence.

On cherche à réaliser un ensemble d'opérations d'analyse sur une séquence donnée. Toutes les opérations doivent être réalisées en **une seule passe**. Vous devez donc compléter le code suivant afin de réaliser les analyses des questions 1 à 6. Le code que nous rajouterons doit uniquement se situer au niveau des `...`. Autrement dit, on s'interdit de rajouter du code à l'intérieur de la boucle **après** la ligne `car = recupere_prochain_caractere()`.

```
... # initialisation
car = recupere_prochain_caractere()
while car != '@':
    ... # analyse
    car = recupere_prochain_caractere()
... # affichage des résultats
```

Voici ce que doit être la sortie de notre programme pour la séquence `'a le a  pelle nouveaux 23 15 5  faux fi15n5@'` une fois que nous aurons réalisé toutes les opérations des questions 1 à 6 :

```
a : 4
le : 2
mots : 10
mots finissant par x : 2
moyenne des longueurs des mots : 3.2
somme des nombres: 63
```

Question 1

Ajouter le code nécessaire pour compter les `a`.

Question 2

Ajouter le code nécessaire pour compter les `le`.

Question 3

Ajouter le code nécessaire pour compter les mots. Un mot est une suite de caractères quelconques qui se termine par un ou plusieurs espaces ou par le caractère de fin de séquence `@`.

Question 4

Ajouter le code nécessaire pour compter les mots terminés par `x`.

Question 5

Ajouter le code nécessaire pour calculer la moyenne de la longueur des mots.

Question 6

Calculer la somme de tous les nombres de la séquence. Un nombre est l'interprétation en base 10 d'une séquence de chiffres (caractères entre `'0'` et `'9'`). Sur l'exemple précédent on obtient ainsi `23 + 15 + 5 + 15 + 5 = 63`.

Exercice 2 : mémoire constante, vraiment ?

Question 1

Sommes-nous vraiment certain de ne pas avoir de problème de mémoire ? Pourquoi ?

Question 2

En supposant que nous puissions traiter un caractère par nanoseconde (1GHz), combien de temps nous faudrait il pour utiliser 1 kilo octet de mémoire avec une séquence ne contenant que des `a` ?

Question 3

Écrire un programme qui sature la mémoire petit à petit de façon visible simplement avec un seul entier.

Exercice 3 : mais comment est-ce possible ? (pour aller plus loin)

Question 1

Comment l'interpréteur Python fait-il pour avoir des entiers arbitrairement grands alors que sur une machine 64 bits, le processeur supporte les opérations sur les entiers uniquement si ces derniers tiennent sur 64 bits ?