Université Lumière Lyon 2 Institut de la communication

Université Lumière Lyon 2

Licence 1 Informatique – 2022/2023 – S2

Algorithmique et programmation Python

TD 7 : Modules et Fonctions

Rappel des consignes :

a. Créez un dossier pour la séance.
b. Regroupez les fichiers Python « .py » dans ce dossier.
c. 1 exercice = 1 fichier Python
d. Le nom du fichier doit être en relation avec le numéro de l'exercice (« exercice_1.py », etc.)
e. Rédigez en parallèle les algorithmes pour chaque exercice en pseudo-code dans un fichier word unique.
f. A la fin de l'heure, vous regroupez les fichiers du dossier (word unique et .py pour chaque exercice) dans une archive (zip, 7z, rar, autre…)
g. A rendre sur votre espace Moodle.
h. N'oubliez pas de commenter abondamment votre code source Python.

NB : Cette séance est particulière.

1. Vous devez créer un module unique « **chaines.py** » au sein duquel vous insérerez toutes les fonctions que vous aurez à programmer.
2. Ensuite, pour chaque exercice, vous devez écrire un programme principal qui importe le module et fait appel à la fonction adéquate. Respectez la numération des exercices (exercice_1.py, etc.) pour faciliter le travail de correction.
3. Dans le fichier Word des pseudo-codes, vous ne mettez que les algorithmes des fonctions.

Exercice 1 – Recherche de lettre dans une chaîne

Dans le module « **chaines.py** », insérez la fonction : **compter(mot, lettre)**

- Mot représente une chaîne de caractères
- Lettre un caractère
- La fonction doit renvoyer le nombre d'apparition de la lettre dans le mot.
- Attention :
  - Vous devez vous appuyer sur votre propre implémentation (une boucle de parcours des lettres du mot peut-être ?) sans utiliser la méthode prédéfinie count() !
  - Le type chaîne de caractère **str** est pourvu d'un certain nombre d'attributs et de méthodes, voir https://www.w3schools.com/python/python_ref_string.asp.
  - Une chaîne est comme un tableau indicé, il est donc possible d'accéder aux lettres qui la compose, le premier indice est **0** (zéro) .

- On peut aussi voir une chaîne de caractères comme une séquence de lettres.

Dans votre programme principal (**exercice_1.py**), vous devez :

- Importer le module ci-dessus (il doit être dans le même dossier que votre programme principal)
- Faire saisir par l'utilisateur le mot et la lettre
- Les convertir en majuscule
- Faire appel à la fonction ci-dessus pour comptabiliser le nombre d'apparition de la lettre dans le mot.

Exercice 2 – Recherche de lettres distinctes

Rajouter dans le module « **chaines.py** » la fonction **distincts(mot)**

- Mot représente une chaîne de caractères
- La fonction doit renvoyer les lettres distinctes dans le mot sous la forme d'une chaîne de caractères (ex. pour « politologie », les lettres distinctes sont « politge »)

Dans votre programme principal (exercice2.py), vous devez :

- Importer le module
- Effectuer la saisie du mot
- Le convertir en majuscule
- Faire appel à la fonction pour afficher les lettres distinctes du mot

Exercice 3 - Comptage des lettres d'un mot

Rajouter dans le module « chaines.py » la fonction **frequences(mot)**

- Mot représente une chaîne de caractères
- La fonction retourne sous la forme d'un dictionnaire les lettres qui composent le mot (clé) et leur nombre d'apparition (valeur). Par ex. pour « totor », nous obtiendrons {'t' : 2, 'o' : 2, 'r' : 1}

Dans votre programme principal (exercice3.py), vous devez :

- Importer le module
- Effectuer la saisie du mot
- Le convertir en majuscule
- Faire appel à la fonction pour afficher le résultat sous la forme : 't' apparaît 2 fois, 'o' apparaît 2 fois, etc.

Remarque : Peut-être auriez-vous avantage à utiliser les 2 fonctions ci-dessus ?

Exercice 4 – Inclusion

Rajouter dans le module « chaines.py » la fonction **inclusion(mot_1, mot_2),** laquelle renvoie un booléen True ou False

- mot_1 et mot_2 représentent des chaînes de caractères

- La fonction doit indiquer si mot_1 est inclus dans mot_2 (True) ou non (False). On entend par inclusion ici le nombre d'apparition de lettres. Par exemple, « pates » est inclus dans « platanes » dans le sens où (a) chaque lettre du premier mot apparaît dans le second, (b) leur nombre d'apparition est inférieur ou égal ; en revanche, « papa » n'est pas inclus dans « paratonnerre » parce que « p » apparaît 2 fois dans le premier mot ; « pipa » non plus n'est pas inclus dans « paratonnerre » parce que « i » n'est pas présent dans le second.

Dans votre programme principal (exercice4.py), vous devez :

- Importer le module
- Effectuer la saisie des 2 mots
- Les convertir en majuscule
- Faire appel à la fonction pour la vérification de l'inclusion.

Remarque : peut-être auriez-vous avantage à exploiter les fonctions programmées précédemment ?

Exercice 5 – Anagrammes

Rajouter dans le module « chaines.py » la fonction **anagramme(mot_1, mot_2),** laquelle renvoie un booléen True ou False

- mot_1 et mot_2 représentent des chaînes de caractères
- La fonction doit indiquer si mot_1 est une anagramme de mot_2 c.-à-d. formé par les mêmes lettres, mais lesquelles sont positionnées différemment (ex. « CARTE » et « ECART » sont des anagrammes)

Dans votre programme principal (exercice5.py), vous devez :

- Importer le module
- Effectuer la saisie des 2 mots
- Les convertir en majuscule
- Faire appel à la fonction pour la vérification

Remarque : si les deux chaînes sont de longueurs différentes, il n'est même pas nécessaire de vérifier les lettres qui les composent

Exercice 6 – Cryptographie, chiffre de César

Rajouter dans le module « chaines.py » la fonction **chiffre_cesar(mot,decalage)**, laquelle renvoie une chaîne de caractères

- mot représente une chaîne de caractères
- décalage est un entier que l'on considère positif
- La fonction renvoie l'équivalent après cryptage du mot
- L'algorithme de cryptographie consiste à remplacer chaque lettre d'un mot par une autre lettre en opérant un décalage dans l'alphabet. On considère que toutes les lettres sont en majuscule (26 lettres en tout, de 'A' à 'Z'). Par exemple, « TOTO » avec un décalage de 2, deviendrait « VQVQ ».
- Attention, si le décalage nous fait déborder de 'Z', on recommence au début de l'alphabet. Par exemple, « ZAZA » avec un décalage de 3 devient « CDCD ».

- Remarque : (1) Une piste possible consiste à former une chaîne de référence avec toutes les lettres de l'alphabet (« ABCDEFGHIJKLMNOPQRSTUVWXYZ »), vous pourrez ainsi y trouver la lettre à coder et la valeur de substitution après décalage. (2) `find()` permet de trouver la position d'une lettre dans un mot (ex. « carte ».find(« a ») renvoie 1 [puisque les indices commencent à 0]).

Dans votre programme principal (exercice6.py), vous devez :

- Importer le module
- Effectuer la saisie du mot et le convertir en majuscule
- Effectuer la saisie du décalage
- Faire appel à la fonction pour produire le mot crypté