Master SVS
Master Sc. Cog.

Intro à la bio-informatique par la programmation – Examen –

11/12/2020

Tout équipement de calcul, programmation, communication est interdit. Le sujet comporte 2 parties notées séparément, chacune d'elles étant composée de plusieurs exercices indépendants. Lire d'abord la totalité de l'énoncé, et commencer par ce qui vous semble le plus facile. Si besoin, on pourra supposer certaines questions résolues.

PARTIE 1

Exercice 1 : (**somme des longueurs**) Ecrivez en python une fonction `sum_length` qui prend en entrée une liste de chaînes de caractères et retourne la somme des longueurs des chaînes de la liste. Par exemple `sum_length(['truc','machin','chouette'])` retourne `18`. Par ailleurs, `sum_length([])` retourne `0`.

Exercice 2 : (**Recherche de mots dans un texte**) Ecrire une fonction `sousseq()` qui prend en argument deux chaînes de caractères, l'une courte et l'autre longue, qui représentent respectivement un mot et un texte, et qui renvoie la position de la première occurrence de la première chaîne de caractères dans la seconde. Exemple : `sousseq("ATG","GTTACGATGCGTATG")` renvoie 6.

Exercice 3 : (**Calcul de la moyenne**) Ecrire un programme qui demande des valeurs entières à l'utilisateur jusqu'à ce qu'une valeur négative soit saisie et affiche la moyenne des valeurs positives données. Par exemple, si on rentre successivement les valeurs 12, 14, -3, le programme renvoie la valeur 13 car $(12 + 14)/2 = 13$.

Exercice 4 : Créez une procédure qui demande d'abord un entier (nombre de lignes) et affiche une figure similaire à celle de droite.

```
Combien de lignes ? 6
-----**-----
----****----
---******---
--********--
-**********-
**********
```

Exercice 5 : (**Permutation de lettres**) *Il pîaart que puor la lctreue, l'orrde des lrttees à l'iétunreir des mots n'a acnuue itnpocmare. La sulee chose qui cptmoe est que la pemièrre et la dneèirre lttree seonit à leur pclae.* [1] Écrire un programme permettant de tester cette théorie. Il devra prendre en entrée une chaîne de caractères et mélanger aléatoirement les lettres à l'intérieur des mots. On supposera que la chaîne de caractères ne comporte pas de signe de ponctuation. On écrira d'abord une fonction `permutemot` qui prend en entrée un mot et permute les lettres à l'interieur du mot en gardant les première et dernière lettres inchangées, puis une deuxième fonction `permute` qui prend en entrée une phrase, et applique la première fonction sur chacun des mots, puis renvoie la phrase contenant les mots permutés. Exemple : `permute("Je vais avoir une bonne note")` peut renvoyer `'Je vias avior une bnone ntoe'`.

Indications : `lesmots=phrase.split(" ")` renvoie la liste des mots de la phrase. On utilisera aussi le module `random` qui contient la fonction `random.randint(a:b)` qui renvoie un nombre aléatoire compris dans l'intervalle $[a, b]$.

Exercice 6 : Ecrivez une fonction `isSeq` qui prend en entrée une chaîne de caractères et retourne un booléen qui dit si cette chaîne ne contient que les caractères `A T G C` ou `N`. La lettre `N` signifie que le séquençage n'a pas permis de connaître précisément la base à cet endroit. Par exemple `isSeq('ATGCCCCNTCN')` renvoie `True` alors que `isSeq('ATGVCCCNTCN')` renvoie `False` à cause de la présence de la lettre `V`.

Exercice 7 : Ecrivez une fonction `compatibles` qui prend en entrée deux séquences (supposées correctes au sens de l'exercice précédent) et retourne un booléen qui dit si elles pourraient être égales. Cela signifie qu'elles ont même longueur et que pour chaque position de lettre, soit les deux lettres sont égales soit l'une au moins est égale à `N`. Par exemple, `AATTAGC` est compatible non seulement avec elle-même mais aussi avec `ANTTANC`. `compatibles('AATTAGC','ANTTANC')` renvoie donc `True`. En revanche, `compatibles('ANTTAGC','AAGTANC')` renvoie `False`.

1. Il paraît que pour la lecture, l'ordre des lettres à l'intérieur des mots n'a aucune importance. La seule chose qui compte est que la première et la dernière lettre soient à leur place

PARTIE 2

Exercice 8 : (**Base de données bibliographiques**) Nous considérons ici une liste de dictionnaires qui représentent une liste d'articles scientifiques. Chaque élément de la liste est un dictionnaire du type :

```
{"titre"    : "PNA-Based MicroRNA Detection Methodologies",
 "auteur"   : ["Enrico Cadoni", "Alex Manicardi", "Annemieke Madder"],
 "date"     : 2020,
 "journal"  : "Molecules",
 "pages"    : "769",
 "volume"   : 25,
 "numero"   : 6}
```

1. Ecrire la fonction `listeTitres` qui prend en entrée la liste des dictionnaires et retourne la liste des titres des articles publiés.
2. Ecrire la fonction `listeTitresAnnee` qui prend en entrée la liste des dictionnaires ainsi qu'un entier représentant une année, et retourne la liste des titres des articles publiés pendant l'année passée en paramètre.
3. Ecrire une fonction qui prend en argument la liste des dictionnaires et un auteur sous forme de chaîne de caractères et qui renvoie les titres des articles auxquels il a participé.
4. On suppose maintenant que la fonction répondant à la question précédente est écrite. Comment peut-on faire pour avoir la liste des titres des articles auxquels ont participés deux auteurs donnés, par exemple les auteurs "Enrico Cadoni" et "Alex Manicardi". On supposera que chacun des titres est unique dans la liste des publications. Expliquez votre solution et donnez le code de la fonction python correspondante (la fonction prend 3 arguments : les deux noms d'auteurs sous forme de chaînes de caractères et la liste de dictionnaires, et elle renvoie la liste des titres).
5. Ecrire une fonction qui prend en argument la liste des dictionnaires et un mot quelconque et qui renvoie la sous liste des dictionnaires qui contiennent ce mot soit dans le titre, soit dans l'un des noms des auteurs, soit dans le nom du journal.

Exercice 9 : (**Jeu des allumettes**) On considère le jeu suivant à deux joueurs : 21 allumettes sont disposées sur une seule rangée. Chaque joueur peut enlever 1, 2 ou 3 allumettes à la fois mais est obligé d'en retirer au moins une. Celui qui perd, c'est celui qui enlève la dernière allumette.

La stratégie pour gagner est la suivante. Pour gagner, il suffit de laisser une allumette. Donc au coup d'avant, il faut en laisser 5 : effet s'il reste 5 allumettes, quelque soit le choix de l'adversaire (enlever 1,2 ou 3 allumettes), on pourra toujours le laisser perdre en ne laissant qu'une allumette. De manière similaire, il faut en laisser 9, 13, 17, ou 21. Du coup, c'est celui qui commence qui perd.

Le but de cet exercice et de programmer ce jeu en python. L'ordinateur joue le deuxième joueur : il gagne toutes les parties. A chaque tour, le programme devra demander au joueur combien d'allumettes il souhaite enlever, vérifiera que ce nombre est compatibles avec les contraintes, puis jouera ensuite en fonction de la réponse donnée.

Exemple de partie :

```
>>> allumettes()
|||||||||||||||||||| (il reste 21 allumettes)
Votre tour : combien d'allumettes enlevez-vous ? 3
||||||||||||||||||
    Je joue  1
|||||||||||||||| (il reste 17 allumettes)
Votre tour : combien d'allumettes enlevez-vous ? 3
|||||||||||||
    Je joue  1
|||||||||||| (il reste 13 allumettes)
Votre tour : combien d'allumettes enlevez-vous ? 2
||||||||||
    Je joue  2
||||||||| (il reste 9 allumettes)
Votre tour : combien d'allumettes enlevez-vous ? 1
||||||||
    Je joue  3
||||| (il reste 5 allumettes)
Votre tour : combien d'allumettes enlevez-vous ? 3
||
    Je joue  1
| (il reste 1 allumette)
Votre tour : combien d'allumettes enlevez-vous ? 1
vous avez perdu !
```