TP 1

Ludovic Arnould

October 2019

Au cours du premier TP nous nous contenterons probablement d'utiliser la plateforme google colab pour faire les exercices.
https://colab.research.google.com

Au fur et à mesure du cours, les exercices se complexifieront. Un concept important de Python qui vous permettra de résoudre les exercices les plus difficiles est celui de **modularité**. Il s'agit de décomposer un problème difficile en différentes étapes et différentes tâches simples (en différents modules) puis de tout regrouper ensemble pour résoudre le problème. **La première étape de l'écriture d'un programme, c'est donc bien la conception : prendre une feuille et un stylo et se demander comment réaliser l'objectif, quelles fonctions vont intervenir, dans quel ordre, etc.**

Un conseil d'ordre général : prenez l'habitude de reproduire les scripts que vous lisez lorsque vous apprenez de la programmation. Modifiez les un peu pour voir ce qu'il se passe, puis essayez d'ecrire les scripts vous-mêmes sans regarder ceux du cours.

Exercice 1
Reproduire les scripts du cours

Exercice 2

1. Ecrire une fonction qui prend en entrée un entier et qui affiche 'Oui cet entier est pair' Ou 'Non, cet entier est impair'.
2. Ecrire une fonction qui prend deux nombres en entrée et renvoie leur moyenne.
3. Ecrire une fonction qui indique si un nombre (<1000) est premier. Envoyer un message d'alerte si le nombre est plus grand que 1000.
4. Vérifiez pour $n < 1000$ la conjecture de Goldbach qui postule que tout entier pair supérieur à 3 peut s'écrire comme somme de deux nombres premiers (éventuellement égaux). A cet égard il peut être utile de stocker tous les nombres premiers < 1000
5. Montrer que la conjecture suivante est fausse : Tout nombre impair est la somme d'une puissance de 2 et d'un nombre premier.

Exercice 3

1. Ecrire une fonction qui prend en entrée une liste et retourne le max (resp min, somme et moyenne) de la liste
2. Ecrire une fonction qui prend en entrée une liste et retourne une liste triée dans l'ordre croissant (resp décroissant)
3. Ecrire une fonction qui prend en entrée deux listes et retourne une liste triée par ordre croissant avec les éléments des deux listes.

Exercice 4

1. Ecrire une fonction qui convertit chaque élément d'une liste en str puis les concatène.
2. Ecrire une fonction qui ajoute "!!!" à la fin d'une string.
3. Ecrire une fonction qui prend en entrée une liste et concatène les éléments impairs. Exemple : [11, 2, 31, 28] renvoie 1131

Exercice suite géométrique

Ecrire une fonction qui prend en entrée une suite de nombres et renvoie True si la suite est en progression géométrique, False sinon.
Exemple : [1,2,4,8] renvoie True et [2, 4, 6] renvoie False

Exercice ROT13

Le ROT13 (rotate by 13 places) est un cas particulier du chiffre de César ; comme son nom l'indique, il s'agit d'un décalage de 13 caractères de chaque lettre du texte à chiffrer : A devient N, B devient O, C devient P, etc. Son principal aspect pratique est que le codage et le décodage se réalisent exactement de la même manière, puisque notre alphabet contient 26 lettres.

1. Définir une fonction nommée rot13 qui prend en argument une chaîne de caractères et retourne cette même chaîne codée en ROT13. On conviendra que : les caractères a, b, c, . . ., y, z seront transformés en n, o, p, . . ., l, m ; tous les autres caractères (espaces, ponctuation, chiffres, caractères accentués, . . .) ne seront pas modifiés.
   Indication Définir une chaîne de caractères alph = 'abcdefghijklmnopqrstuvwxyz' et utiliser le fait que si c est un caractère et s une chaîne de caractères, l'instruction c in s renvoie un booléen caractérisant la présence ou non de c dans s .
2. Utilisez votre fonction pour connaître la réponse à l'énigme suivante :
   Quelle est la différence entre un informaticien et une personne normale ?
   har crefbaar abeznyr crafr dh'ha xvyb-bpgrg rfg étny à 1000 bpgrgf, ha vasbezngvpvra rfg pbainvaph dh'haxvybzèger rfg étny à 1024 zègerf.

Exercice : Suite de Conway

Les premiers termes de la suite de Conway sont : 1, 11, 21, 1211, 111221, . . ., chaque terme étant obtenu en lisant à haute voix le terme précédent (c'est pourquoi Conway avait baptisé cette suite Look and say). Par exemple, le terme 1211 se lit un 1, un 2, deux 1 donc le terme suivant est 111221.

1. Rédiger en Python une fonction lookandsay(n) qui retourne le terme qui suit l'entier n dans cette énumération.
   Remarque. Pour rédiger cette fonction, il est plus facile de manipuler les entiers sous la forme de chaînes de caractères. Par exemple, lookandsay('1332') retournera la chaîne '112312' .

2. Afficher à l'aide de cette fonction les 20 premiers termes de la suite de Conway.

3. Il a été démontré que si on note u_n le nombre de chiffres du n^{eme} terme de cette suite alors le rapport u_{n+1}/u_n tend vers une constante, appelée constante de Conway. Calculez-en une valeur approchée.

4. Une autre propriété remarquable de cette constante est qu'elle ne dépend pas de la valeur initiale (à l'exception de 22 quiproduit une suite constante). Le vérifier expérimentalement en testant différentes valeurs initiales.

Exercice plus difficile (proposé en entretien d'embauche)
Une chaine de N caractères S est dite **imbriquée** si les trois conditions suivantes sont vérifiées :

- S est vide
- S est de la forme "(U)" or "[U]" or "U" où U est elle-même imbriquée
- S est de la forme "VW" où V et W sont imbriquées.

Par exemple, "{[()()]}" est imbriquée mais "([])()]" ne l'est pas.

Objectif : Ecrire une fonction qui prend en entrée une chaîne de caractères et renvoie 1 si la chaîne est imbriquée, 0 sinon.
On autorise seulement pour S les caractères suivants : "(", "{", "[", "]", "}" and/or ")".
Contraintes : Complexités temporelle et spatiale en O(len(S))
Indication : il faut d'abord trouver sous quelle condition une chaine est imbriquée. Commencez avec des exemples simples avant de généraliser. Il peut être utile d'utiliser les indices de la chaine.