M1 BIO-INFO
X1BI030

Université de Nantes
2021/2022

Feuille d'exercices n° 2
Programmation Python 1

Exercice 2.1

1. Écrire un programme demandant son prénom à l'utilisateur et affichant le nombre de lettres le composant ;
2. Écrire un programme prenant en paramètre un millésime et indiquant s'il s'agit d'une année bissextile. Reprendre le problème en écrivant une fonction `is_bissextile()` ;
3. Écrire le programme affichant la moyenne des valeurs numériques passées en paramètres (ou 0 si aucune valeur numérique n'est donnée) ;
4. Écrire le programme prenant une liste d'entiers en entrée et créant une liste composée des valeurs négatives de la première liste ;
5. Écrire un programme demandant son prénom et son année de naissance à l'utilisateur et affichant le message « *<Prénom>*, vous aurez cent ans en *<année>*. », où *année* est l'année de leurs cent ans.

Exercice 2.2

Implémenter un jeu de chifoumi contre l'ordinateur : le programme fait un choix au hasard parmi « pierre », « feuille », « ciseau » et demande à l'utilisateur de faire son propre choix, puis il détermine le gagnant.

Exercice 2.3

Écrire la fonction `factorielle()` calculant la factorielle $n!$ de n : $\begin{cases} 0! = 1 \\ n! = n(n-1)! \end{cases}$

Exercice 2.4

Écrire le programme le plus concis possible affichant la sortie ci-dessous :

```
1
22
333
4444
55555
666666
7777777
88888888
999999999
```

Exercice 2.5

Écrire le programme du jeu « Plus grand / plus petit » pour deviner un nombre compris en 1 et 100. On affichera le nombre de coups nécessaire.

Exercice 2.6

Écrire un programme demandant une chaîne sans accents ni ponctuation et indiquant si c'est un palindrome.

Exercice 2.7

Écrire le programme demandant à l'utilisateur une borne maximale n et affichant la somme des nombres divisibles par 3 compris entre 3 et n compris.

Exercice 2.8

Écrire un programme prenant en entrée un fichier de la forme ci-dessous à gauche et affichant sur la sortie standard le résultat ci-dessous à droite :

pomme	5
poire	3
banane	2
pomme	4
prune	2
banane	1

banane	3
poire	3
pomme	9
prune	2

Exercice 2.9

Écrire un programme lisant le contenu d'un fichier et affichant le nombre d'occurrences de chaque mot du texte. On acceptera de lire le contenu sur l'entrée standard ou de prendre le nom du fichier en paramètre indifféremment.

Exercice 2.10 (D'après Rosalind)

Écrire le programme prenant en paramètres deux chaînes d'ADN s_1 et s_2 de même taille et retournant la distance de Hamming entre elles, i.e., le nombre de fois où $s_1[i] \neq s_2[i]$.
Exemple :

$$\text{Hamming}(\text{"GAGCCTACTAACGGGAT"}, \text{"CATCGTAATGACGGCCT"}) = 7$$

On vérifiera que les deux chaînes sont de même taille et l'on affichera un message d'erreur dans le cas contraire. On s'assurera aussi que l'utilisateur a bien passé deux chaînes en paramètres.

Exercice 2.11

1. Écrire le programme Python demandant une chaîne de bases à l'utilisateur et affichant la chaîne complémentaire.
   Aide : on pourra utiliser un dictionnaire ;
2. Modifier le programme pour lire la chaîne sur la ligne de commande. On affichera un message d'aide si l'utilisateur appelle le programme sans paramètre.

Exercice 2.12 (D'après « Programming for Biologists »)

Le fichier `birds.py` référence le nombre d'oiseaux bagués sur différents sites. Il est composé d'une liste de paires, chaque paire étant elle-même composée de l'identifiant d'un site de baguage et du nombre d'oiseaux bagués sur le site.

1. Combien de sites y a t-il ?
2. combien d'oiseaux ont été bagués sur le septième site ?
3. Combien d'oiseaux ont été bagués sur le dernier site ?
4. Combien d'oiseaux ont été bagués au total ?
5. Combien d'oiseaux ont été bagués au total sur les sites dont le code commence par `'C'` ?

Exercice 2.13

Implémenter le jeu « Bulls and Cows » : votre programme choisit un nombre de 4 chiffres entre 0 et 9 tous différents et vous devez trouver ce nombre en un minimum d'essais. À chaque essai, le programme indique le nombre de chiffres bien placés (les *bulls*) et le nombre de chiffres mal placés mais présents (les *cows*).

Exercice 2.14

Écrire un programme Python testant si un mot passé en paramètre est un hétérogramme. Les espaces ne seront pas considérés et les majuscules seront assimilées aux minuscules correspondantes ; de même, les lettres accentuées « ÀÁÂÃÄÅàáâãäåÒÓÔÕÖØòóôõöøÈÉÊËèéêëÇçÌÍÎÏìíîïÙÚÛÜùúûüÿÑñ » seront assimilées à leur forme non accentuée.
Exemples d'appels :

```
% heterogramme.py "Lampez un fort whisky!"
C'est un hétérogramme
% heterogramme.py cryptogamiques
C'est un hétérogramme
% heterogramme.py heterogramme
Ce n'est pas un hétérogramme
```