Université de Picardie Jules Verne — Master 1 ISC: Intro. à la Prog.
TP 1 : Premiers pas en python

Exercice 1: Structures de contrôle

a) Ecrire un programme qui affiche les nombres pairs compris entre 0 et 16000 par ordre croissant.

b) Ecrire un programme qui affiche les nombres impairs compris entre 0 et 16000 par ordre décroissant.

c) Ecrire un programme qui demande un entier n à l'utilisateur et produit le triangle dont la base est de taille n (ici $n = 5$):

```
*
* *
* * *
* * * *
* * * * *
```

On testera au préalable l'instruction `9*'kk'`

d) Ecrire un programme qui trace le damier suivant:

```
**##**
**##**
##**##
##**##
**##**
**##**
```

on demandera à l'utilisateur le nombre de lignes n et la taille de chaque case c. Dans l'exemple précédent, $n = 3$ et $c = 2$.

e) Demander à l'utilisateur de rentrer deux chaines de caractères. Ecrire les instructions qui permettent de dire quelle est la plus grande (on utilisera l'instruction `len`).

Exercice 2: Entrées-Sorties

a) Ecrire les instructions qui permettent d'afficher les entiers de 1 à 10, leur carré et leur cube selon le format suivant:

```
1     1     1
2     4     8
3     9    27
4    16    64
5    25   125
6    36   216
7    49   343
```

```
8   64  512
9   81  729
10 100 1000
```

b) Reprendre l'exercice précédent mais au lieu d'afficher le résultat à l'écran, mettre le résultat dans le fichier 'exofich.txt'. Vérifier en ouvrant le fichier dans un premier temps puis en demandant à Python de lire ce fichier.

c) Ajouter au fichier précédent la suite de nombres construite comme précédemment mais cette fois-ci on prend le carré et le cube de 0.1, 0.2, $\cdots$, 1.

Exercice 3: Fonctions

a) Ecrire une fonction qui prend en arguments d'entrée deux nombres a et b et qui renvoie le rapport a/b. Essayer avec $a = 1$ et $b = 2$.

b) On indique qu'un nombre est dit parfait si il est la somme de ses diviseurs (lui même étant exclu). Par exemple, les diviseurs de 6 sont 3, 2 et 1 et $3 + 2 + 1 = 6$, donc 6 est parfait.

Ecrire une fonction qui prend en argument un entier n et qui affiche si ce nombre est parfait ou non. On pourra utiliser une liste pour stocker les diviseurs et utiliser la fonction `sum` pour additionner les éléments de la liste.

Ecrire ensuite un programme qui affiche tous les nombres parfaits compris entre 1 et 1000.

c) Ecrire une fonction qui calcule les N premiers termes de la suite de Fibonacci définie par:

$$u_{n+1} = u_n + u_{n-1}$$

avec l'initialisation $u_0 = 1$, $u_1 = 0$.

Sauvegarder ces nombres dans un fichier nommé `fibo.txt`.

d) On souhaite vérifier la loi de Benford sur quelques cas particuliers. Benford s'est aperçu que dans une série numérique provenant de la vie courante (longueur des fleuves, cours de la bourse ...), la fréquence des premiers chiffres non nuls des nombres impliqués n'étaient pas equi-répartis: ils commencent plus fréquemment par 1, puis par 2 etc.

- Ecrire une fonction `prem` qui prend un nombre en argument d'entrée (qui peut être plus petit que 1, ex: 0.04) et qui renvoie le premier chiffre non nul (dans l'exemple 4).
- Ecrire les instructions qui permettent d'observer que la suite de Fibonacci vérifie la loi de Benford: représenter les données en écrivant sur la ligne i autant d'étoiles que le nombre i est apparu dans la série. On pourra utiliser l'instruction `res.split()` qui transforme en liste la chaine de caractères `res`.