Informatique - TP 8

TD noté - 17 janvier

NB n°1 : vous enregistrerez votre travail dans votre répertoire réseau sous le titre `TD6-VotreNom.py`.
La première ligne de votre code devra faire apparaître un commentaire dans lequel sera écrit votre nom.
NB n°2 : parmi les questions qui suivent, celles pour lesquelles la réponse attendue n'est pas un code Python, seront traitées sous forme de commentaire dans votre code.
NB n°3 : on ne demande pas de vérifier la validité des arguments fournis aux fonctions demandées dans les exercices.

1. Créer une liste `impairs` constituée des entiers ***impairs*** de l'intervalle $[\![0;100]\!]$.
2. Écrire une fonction `crible(liste)` qui prend en paramètre une liste d'entiers et ***renvoie*** la liste constituée des éléments de `liste` dont l'écriture décimale contient le chiffre 7.
3. Afficher le nombre d'éléments de la liste `crible(impairs)`. Vérifier qu'il vaut 14.
4. On définit la suite $(D_n)_{n\in\mathbb{N}}$ par $\begin{cases} D_0 &= 1 \\ D_{n+1} &= \dfrac{2(2n+1)}{n+2} D_n \end{cases}$.
   Écrire une fonction `D(n)` qui prend en paramètre un entier n et renvoie la valeur de D_n.
5. On définit la suite $(C_n)_{n\in\mathbb{N}}$ par $\begin{cases} C_0 &= 1 \\ C_{n+1} &= \displaystyle\sum_{k=0}^{n} C_k C_{n-k} \end{cases}$.
   Par exemple, $C_1 = C_0^2 = 1$, $C_2 = C_0C_1 + C_1C_0 = 2$, etc...
   Écrire une fonction `C(n)` qui prend en paramètre un entier n et renvoie la valeur de C_n.
6. Soit p un entier strictement positif.
   On définit la suite $(E_n)_{n\in\mathbb{N}}$ par $\begin{cases} E_0 &= p \\ E_{n+1} &= \begin{cases} E_n/2 & \texttt{si } E_n \texttt{ est pair} \\ 3\times\frac{E_n+1}{2} & \texttt{si } E_n \texttt{ est impair} \end{cases} \end{cases}$.
   Écrire une fonction `descente(p)` qui prend en paramètre un entier p et renvoie le couple (E_n, n) où n est le plus petit entier vérifiant $E_n < 4$.
7. Afficher `descente(17)`. Vérifier que le résultat vaut $(3, 14)$.
8. Écrire une fonction `M(n)` qui prend pour argument un ***entier*** `n` et a pour effet d'***afficher*** les $2n+1$ lignes décrites par les exemples suivants :

```
>>> M(1)
* *
***
* *
```

```
>>> M(2)
*   *
** **
* * *
*   *
*   *
```

```
>>> M(3)
*     *
**   **
* * * *
*  *  *
*     *
*     *
*     *
```

9. Faire afficher `M(4)`.
10. Écrire une fonction `separe(liste)` qui prend en paramètre une liste de flottants, calcule le maximum M et le minimum m de cette liste, et ***renvoie*** le couple (a, b) où a est le nombre d'éléments de la liste ***strictement inférieurs à*** $\frac{m+M}{2}$ et b est le nombre d'éléments de la liste ***supérieurs ou égaux à*** $\frac{m+M}{2}$

11. Le module `random` permet de générer des nombres pseudo-aléatoires. Le code suivant génère par exemple une liste `liste` de 30 nombres flottants compris dans l'intervalle $[-30; 30[$:

```
import random as rd
liste=[-30+60*rd.random() for k in range(30)]
```

Tester la fonction `separe` sur une liste de 100 nombres flottants aléatoires pris dans l'intervalle $[1; 899[$.

12. La chaîne de caractères `s = 'Dtcxq"#"Xqwu"cxg|"vqwvg"oqp"guvkog"#'` a été obtenue en cryptant une autre chaine de caractères `chaine` de la façon suivante :

- pour chaque caractère de `chaine`, on a obtenu son code ***Unicode*** puis on lui a ajouté 2 ;
- on a ensuite transformé ce nombre en un nouveau caractère et concaténé tous les caractères ainsi obtenus.

Donner la chaine de caractères `chaine` originale - soit explicitement, soit sous la forme d'un code permettant de décrypter `s`.
Le message décodé `chaine` est un message personnel de ma part !

Informatique - TP 8

TD noté - 17 janvier

NB n°1 : vous enregistrerez votre travail dans votre répertoire réseau sous le titre `TD6-VotreNom.py`***.***
La première ligne de votre code devra faire apparaître un commentaire dans lequel sera écrit votre nom.
NB n°2 : parmi les questions qui suivent, celles pour lesquelles la réponse attendue n'est pas un code Python, seront traitées sous forme de commentaire dans votre code.
NB n°3 : on ne demande pas de vérifier la validité des arguments fournis aux fonctions demandées dans les exercices.

1. Créer une liste `pairs` constituée des entiers ***pairs*** de l'intervalle $[\![1; 101]\!]$.
2. Écrire une fonction `crible(liste)` qui prend en paramètre une liste d'entiers et ***renvoie*** la liste constituée des éléments de `liste` dont l'écriture décimale contient le chiffre 0.
3. Afficher le nombre d'éléments de la liste `crible(pairs)`. Vérifier qu'il vaut 10.
4. On définit la suite $(K_n)_{n\in\mathbb{N}}$ par $\left\{\begin{array}{lcl} K_0 & = & 1 \\ K_{n+1} & = & -(n+1)K_n + n^2 \end{array}\right.$.
   Écrire une fonction `K(n)` qui prend en paramètre un entier n et renvoie la valeur de K_n.
5. On définit la suite $(C_n)_{n\in\mathbb{N}}$ par $\left\{\begin{array}{lcl} C_0 & = & 1 \\ C_{n+1} & = & \displaystyle\sum_{k=0}^{n}(-1)^k C_k C_{n-k} \end{array}\right.$.
   Par exemple, $C_1 = C_0^2 = 1$, $C_2 = C_0C_1 - C_1C_0 = 0$, etc...
   Écrire une fonction `C(n)` qui prend en paramètre un entier n et renvoie la ***liste des valeurs*** $[C_0, C_1, ..., C_n]$.
   Remarque : notamment, la valeur de retour de `C(n)` doit comporter $n+1$ éléments...
6. Soit p un entier strictement positif.
   On définit la suite $(E_n)_{n\in\mathbb{N}}$ par $\left\{\begin{array}{lcl} E_0 & = & p \\ E_{n+1} & = & \left|\begin{array}{lll} E_n/2 & \texttt{si} & E_n \texttt{ est pair} \\ \frac{3E_n+1}{2} & \texttt{si} & E_n \texttt{ est impair} \end{array}\right. \end{array}\right.$.
   Écrire une fonction `syr(p)` qui prend en paramètre un entier p et renvoie le couple (E_n, n) où n est le plus petit entier vérifiant $E_n < 2$.
7. Afficher `syr(27)`. Vérifier que le résultat vaut $(1, 70)$.
8. Écrire une fonction `W(n)` qui prend pour argument un ***entier*** `n` et a pour effet d'***afficher*** les $2n+1$ lignes décrites par les exemples suivants :

```
>>> W(1)
* *
***
* *
```

```
>>> W(2)
*   *
*   *
* * *
** **
*   *
```

```
>>> W(3)
*     *
*     *
*     *
*  *  *
* * * *
**   **
*     *
```

9. Faire afficher `W(4)`.
10. Écrire une fonction `separe(liste)` qui prend en paramètre une liste d'entiers et ***renvoie*** le couple (a, b) où a est le nombre d'entiers ***pairs*** de la liste `liste` et b le nombre d'entiers impairs.

11. Le module `random` permet de générer des nombres pseudo-aléatoires. Le code suivant génère par exemple une liste `liste` de 30 nombres entiers compris dans l'intervalle $[\![-30; 29]\!]$:

```
import random as rd
liste=[rd.randrange(-30,30) for k in range(30)]
```

Tester la fonction `separe` sur une liste de 100 nombres aléatoires pris dans l'intervalle $[\![0; 1000]\!]$.

12. La chaîne de caractères `s = 'Eudyr#$#Yrxv#dyh}#wrxwh#prq#hvwlph#$'` a été obtenue en cryptant une autre chaine de caractères `chaine` de la façon suivante :

- pour chaque caractère de `chaine`, on a obtenu son code ***Unicode*** puis on lui a ajouté 3 ;
- on a ensuite transformé ce nombre en un nouveau caractère et concaténé tous les caractères ainsi obtenus.

Donner la chaine de caractères `chaine` originale - soit explicitement, soit sous la forme d'un code permettant de décrypter `s`.
Le message décodé `chaine` est un message personnel de ma part !