Chiffrement en Python

Emmanuel MORAND
(http://www.emmanuelmorand.net)

11 janvier 2008

Ce document a pour objectif de permettre la découverte du langage de programmation Python et de sa très grande efficacité. Les différentes leçons consistent en l'élaboration de programmes de chiffrement (algorithme ROT13, algorithme de Vigenère...) et s'enchaînent de manière progressive.

Installation de l'environnement de travail IDLE de Python

Mots-Clés. *print, lower, upper*

La première chose à faire est de télécharger (gratuitement) la dernière version de Python sur le site http ://python.org, de l'installer puis de lancer l'environnement de travail intégré IDLE. Une fenêtre nommée Python Shell apparait alors à l'écran, c'est la fenêtre de l'interpréteur dont le rôle est d'exécuter les commandes Python entrées par l'utilisateur. Chaque commande Python doit être tapée à la suite du prompt >>> et s'exécute après la frappe de la touche Entrée.

Premières commandes Python

Python peut effectuer des calculs :

```
>>> 3+5*2
13
```

Python peut afficher du texte :

```
>>> print 'Bonjour !'
Bonjour !
```

Python permet de définir des variables et de les utiliser dans des calculs :

```
>>> a=1
>>> b=2
>>> a+b
3
>>> c='Monty'
>>> d='Python'
>>> c+d
'MontyPython'
```

Python est un langage de haut niveau, il contient donc des méthodes évoluées :

```
>>> e='The Meaning of Life'
>>> e.lower()
'the meaning of life'
>>> e.upper()
'THE MEANING OF LIFE'
```

Leçon 0

L'objectif de cette leçon est de créer un premier programme Python.

Mots-Clés. *print, raw_input*

Il est possible d'exécuter d'un bloc plusieurs commandes en les regroupant dans un fichier programme. Commençons par créer un fichier programme à l'aide de la commande du Menu Fichier nommée New Window, une nouvelle fenêtre sans prompt apparait, taper à l'intérieur le texte ci-dessous :

```
# entree du prenom de l'utilisateur
prenom=raw_input('Quel est votre prenom ?\n')
# affichage du message de salutation
print 'Bonjour '+prenom+' !'
# fermeture du programme
raw_input('Fin du programme')
```

Ensuite vous devez sauvegarder le programme sous le nom prog0.py par exemple puis l'exécuter avec le Menu Run et la commande Run Module. Une autre façon d'exécuter le programme est de double-cliquer sur le fichier prog0.py (à condition que l'extension .py soit associée au logiciel python.exe), les entrées et sorties s'effectuant alors au moyen de la console Windows, remarquons à ce propos le rôle de la dernière instruction du programme qui est d'éviter la fermeture prématurée de cette dernière.

Une bonne habitude est de commenter ses programmes (un commentaire en Python s'écrit au moyen du signe dièse) et de donner des noms explicites aux variables.

Le programme ci-dessus fait intervenir la commande raw_input qui permet de faire entrer un texte à l'utilisateur et la commande print qui permet elle d'afficher du texte, notons à ce sujet l'utilisation de la commande \n qui permet d'effectuer un saut de ligne.

Il est prudent dans un premier temps d'éviter de taper dans les programmes des caractères accentués ceux-ci n'étant pas reconnus sans définition préalable de l'alphabet correspondant.

Leçon 1

L'objectif de cette leçon est de construire un programme qui pour l'entrée d'une quelconque des 26 lettres de l'alphabet $a, b, c, \ldots, x, y, z$ renvoie son rang sous la forme $0, 1, 2, \ldots, 23, 24, 25$.

Mots-Clés. *ord, chr, int, str*

Exemples

Voici quelques fonctions Python pouvant être utiles pour ce programme :

- La fonction ord qui renvoie le code numérique d'un caractère (code ascii) :
  ```
  >>> ord('g')
  103
  ```
- La fonction chr qui renvoie le caractère associé à un code ascii :
  ```
  >>> chr(115)
  's'
  ```
- La fonction int qui permet de convertir une chaine de caractères en un nombre entier :
  ```
  >>> int('225')
  225
  ```
- La fonction str qui permet de convertir un entier en chaine de caractères :
  ```
  >>> str(118)
  '118'
  ```

Exercice

Réaliser un programme de codage nommé prog1.py permettant de renvoyer le rang sous la forme $0, 1, 2, \ldots, 23, 24, 25$ d'une quelconque des 26 lettres de l'alphabet $a, b, c, \ldots, x, y, z$, puis réaliser le programme de décodage nommé prog1bis.py permettant de réaliser l'opération réciproque.

Indications : après avoir repéré les codes ascii correspondant aux 26 lettres de l'alphabet on pourra au moyen d'une opération simple se ramener à un nombre compris entre 0 et 25.

Leçon 2

L'objectif de cette leçon est de créer un programme permettant de détecter si un caractère fait partie ou pas de l'alphabet minuscule.

Mots-Clés. *len, if, else*

Exemple

Testez la commande suivante qui permet de déterminer si la phrase 'BRING OUT YOUR DEAD' comporte plus ou moins de 17 caractères :

```
>>> if (len('BRING OUT YOUR DEAD')<17):
    print 'La phrase \'BRING OUT YOUR DEAD\' comporte moins de 17 caractères'
else:
    print 'La phrase \'BRING OUT YOUR DEAD\' comporte plus de 17 caractères'
```

Cette commande utilise :

- La fonction len qui renvoie la longueur d'une chaine de caractères :
  ```
  >>> len('anticonstitutionnellement')
  25
  ```
- L'instruction if qui permet d'exécuter une instruction si une condition donnée est vérifiée, à noter que l'instruction doit être indentée !
- L'instruction else qui permet d'exécuter une instruction alternative si la condition de l'instruction if n'est pas vérifiée, à noter que l'instruction alternative doit être indentée !
- La commande \' qui est nécessaire pour écrire un guillemet dans une chaine de caractères, elle permet d'éviter la confusion avec les guillemets délimiteurs.

Exercice

Réaliser un programme nommé prog2.py qui permet de détecter si un caractère entré par l'utilisateur fait partie ou pas de l'alphabet minuscule.

Indications : opérateurs de comparaison en Python

x==y	x est égal à y
x!=y	x est différent de y
x<y	x est strictement inférieur à y
x>y	x est strictement supérieur à y
x<=y	x est inférieur ou égal à y
x>=y	x est supérieur ou égal à y

Leçon 3

L'objectif de cette leçon est de réaliser un programme permettant de décaler une lettre de treize crans dans l'alphabet.

Théorie

Parmi les algorithmes de chiffrement basiques, on peut tout d'abord évoquer l'algorithme ROT13 qui consiste à décaler chaque lettre du message de treize crans dans l'alphabet :

$$\begin{array}{ccc} a & \mapsto & n \\ b & \mapsto & o \\ & \vdots & \\ y & \mapsto & l \\ z & \mapsto & m \end{array}$$

Mathématiquement, si les lettres de l'alphabet sont numérotées de 0 à 25, l'algorithme consiste simplement à ajouter 13 à leur rang et à considérer le résultat modulo 26 pour s'assurer qu'il reste compris entre 0 et 25. En langage Python, cette opération s'écrit de manière simple :

```
>>> (7+13)%26
20
>>> (21+13)%26
8
```

Exercice

Réaliser un programme de chiffrement nommé prog3.py qui pour un caractère entré par l'utilisateur retourne le caractère chiffré selon l'algorithme ROT13. Pourquoi n'est-il pas nécessaire de réaliser le programme de déchiffrement ?

Indications : le programme se déroulera en plusieurs étapes, à savoir l'entrée du caractère par l'utilisateur, son codage numérique, l'application de l'algorithme ROT13, la reconversion en caractère et enfin l'affichage du caractère chiffré. Dans le cas où le caractère entré par l'utilisateur n'appartient pas à l'alphabet, celui ci sera chiffré par le caractère espace ' '.

Leçon 4

L'objectif de cette leçon est de réaliser un programme qui pour un texte donné retourne la liste de ses caractères chiffrés selon l'algorithme ROT13.

Mots-Clés. *for, range*

Le chiffrement caractère par caractère est bien évidemment fastidieux, il est indispensable de pouvoir chiffrer directement l'ensemble des caractères d'un texte.

Exemple

Tester le programme suivant que l'on nommera enumeration.py qui permet d'énumérer les caractères d'un texte :

```
# entree du texte
texte=raw_input('Entrez un texte :\n')
# boucle d'affichage des caracteres du texte
for i in range (0,len(texte)):
    print texte[i]
# fermeture du programme
raw_input('Fin du programme')
```

Ce programme utilise :

- La fonction range(a,b) qui renvoie la liste des entiers i tels que $a \leq i < b$:
  ```
  >>> range(0,5)
  [0, 1, 2, 3, 4]
  ```
- L'instruction for qui permet de réaliser une boucle, à noter que l'instruction à itérer (ici print texte[i]) doit être indentée !

Exercice

Réaliser un programme de chiffrement nommé prog4.py qui pour un texte entré par l'utilisateur retourne la liste de ses caractères chiffrés selon l'algorithme ROT13.

Leçon 5

L'objectif de cette leçon est de réaliser l'algorithme de chiffrement ROT13.

Le programme de la leçon précédente retourne une liste de caractères chiffrés, ceux-ci doivent encore être rassemblés pour obtenir le message chiffré.

Exemple

Tester et analyser le programme suivant que l'on nommera inversion.py qui permet de réécrire un texte en inversant l'ordre des caractères :

```
# entree du texte
texte=raw_input('Entrez un texte :\n')
# initialisation du texte inverse
texteinverse=''
# renversement du texte
for i in range (0,len(texte)):
    texteinverse=texte[i]+texteinverse
# affichage du texte inverse
print texteinverse
# fermeture du programme
raw_input('Fin du programme')
```

Exercice

Réaliser un programme de chiffrement nommé prog5.py permettant de chiffrer un texte selon l'algorithme ROT13, on appelera par convention texteclair le texte à chiffrer et textechiffre le texte chiffré.

Leçon 6

L'objectif de cette leçon est de modifier le programme de la leçon précédente de manière à permettre le chiffrement direct d'un document texte.

Mots-Clés. *open, close, read, write*

Il est fastidieux d'avoir à taper le message en clair dans la fenêtre de l'interpréteur Python, il serait bien plus agréable que le programme de chiffrement puisse fonctionner directement sur un fichier texte (afin de chiffrer des messages électroniques par exemple). En fait, le langage Python permet d'effectuer cette tache de manière simple.

Exemple

Tester le programme suivant que l'on nommera copie.py qui permet de lire le contenu d'un fichier texte nommé texte1.txt et de le copier dans un nouveau fichier texte nommé texte2.txt (un fichier texte nommé texte1.txt doit bien sûr être créé préalablement dans le repertoire du programme) :

```
# ouverture en lecture du fichier texte1.txt
fichier=open('texte1.txt','r')
# lecture du fichier
texte=fichier.read()
# fermeture du fichier texte1.txt
fichier.close()
# ouverture en ecriture d'un fichier texte2.txt
fichier=open('texte2.txt','w')
# ecriture dans le fichier
fichier.write(texte)
# fermeture du fichier texte2.txt
fichier.close()
```

Ce programme utilise :

- La fonction open qui permet l'ouverture d'un fichier en lecture (r pour read) ou en écriture (w pour write).
- La fonction close qui permet la fermeture d'un fichier.
- Les fonctions read et write qui permettent respectivement la lecture et l'écriture dans un fichier.

Exercice

Réaliser un programme de chiffrement nommé prog6.py permettant de chiffrer un fichier texte nommé messageclair.txt en un fichier texte nommé messagechiffre.txt selon l'algorithme ROT13 (un fichier texte nommé messageclair.txt doit bien sûr être créé préalablement dans le repertoire du programme).

Réaliser le programme de déchiffrement associé prog6bis.py permettant de déchiffrer un message nommé messagechiffre.txt en un fichier texte nommé messageclair.txt.

Indications : le programme prog6.py sera mené en trois étapes, à savoir la lecture de texteclair dans messageclair.txt, le chiffrement de texteclair en textechiffre puis l'écriture de textechiffre dans messagechiffre.txt.

Leçon 7

L'objectif de cette leçon est de réaliser l'algorithme de chiffrement de Vigenère.

Théorie

Les algorithmes de chiffrement à décalage fixe comme le ROT13 sont relativement faciles à casser, en effet, pour un alphabet de 26 lettres, il n'existe que 26 décalages possibles (dont un trivial) et il suffit de tous les tester pour déchiffrer le message considéré. L'algorithme de Vigenère consiste à introduire une clef qui permet de rendre le décalage variable selon la position du caractère dans le message. L'exemple ci-dessous effectue le chiffrement du message en clair "bring out your dead" par l'algorithme de Vigenère avec la clef "grail" :

b	r	i	n	g		o	u	t		y	o	u	r		d	e	a	d	
1	17	8	13	6		14	20	19		24	14	20	17		3	4	0	3	
g	r	a	i	l															
6	17	0	8	11	6	17	0	8	11	6	17	0	8	11	6	17	0	8	11
7	8	8	21	17		5	20	1		4	5	20	25		9	21	0	11	
h	i	i	v	r		f	u	b		e	f	u	z		j	v	a	l	

Le message chiffré "hiivr fub efuz jval" s'obtient en utilisant successivement les décalages donnés par la clef.

Exercice

Réaliser un programme de chiffrement nommé prog7.py et le programme de déchiffrement associé prog7bis.py fonctionnant selon l'algorithme de Vigenère.

Question préliminaire : on souhaite chiffrer un message par l'algorithme de Vigenère avec une clef de longueur 7, quel caractère de la clef donne le décalage à appliquer à la 22^{eme} lettre du message ?

Indications : on pourra se baser sur le programme de la leçon précédente, il faudra ajouter une demande de clef de chiffrement et modifier l'opération mathématique.

Leçon 8

L'objectif de cette leçon est de modifier les programmes de la leçon précédente afin de permettre le chiffrement des caractères accentués, des caractères majuscules, des signes de ponctuation, des chiffres...

Exercice

Réaliser à partir des programmes de la leçon précédente des programmes nommés prog8.py et prog8bis.py permettant le chiffrement selon l'algorithme de Vigenère des caractères accentués, des caractères majuscules, des signes de ponctuation, des chiffres...

Indication : utiliser l'ensembles des codes ascii de 0 à 255 et réaliser le décalage modulo 256.

Leçon 9

L'objectif de cette leçon est de créer un exécutable Windows (.exe) à partir d'un programme Python (.py) afin de pouvoir l'exécuter sur un ordinateur ne possédant pas d'interpréteur Python.

Mots-Clés. *from, import*

Il est nécessaire pour cette leçon d'installer le programme py2exe téléchargeable gratuitement sur le site http ://www.py2exe.org.

Exemple

Considérons un programme Python prog.py que l'on souhaite convertir en exécutable Windows prog.exe, nous devons tout d'abord créer un fichier progsetup.py qui va contenir les informations nécessaires à la conversion :

```
from distutils.core import setup
import py2exe
setup(zipfile=None,console=["prog.py"])
```

L'option zipfile=None impose d'inclure la bibliothèque dans le programme, son omission entraine la création d'un fichier library.zip supplémentaire qui sera nécessaire à l'éxecution du programme. L'option console=["prog.py"] impose l'utilisation de la console Windows pour toutes les entrées clavier du programme.

Ensuite, il faut placer les fichiers prog.py et progsetup.py dans le répertoire d'installation de Python (C :\Program Files\Python25).

A partir de maintenant, la création de l'exécutable Windows va se faire en ligne de commande DOS, il est donc nécessaire d'ouvrir une fenêtre DOS (Menu démarrer > Tous les programmes > Accessoires > Invite de commandes).

On commence par se placer dans le répertoire d'installation de Python au moyen de la commande Change Directory :

```
cd C:\Program Files\Python25
```

Puis on exécute la conversion :

```
python.exe progsetup.py py2exe
```

La conversion génère dans le répertoire d'installation de Python deux nouveaux répertoires nommés dist et build que l'on pourra effacer après avoir récupéré l'exécutable prog.exe et la bibliothèque dynamique python25.dll dans le répertoire dist.

Le programme prog.exe muni de python25.dll peut maintenant fonctionner sous Windows sans interpréteur Python.

Exercice

Convertir les programmes de la leçon précédente en exécutables Windows.

Solutions des exercices

Leçon 1

prog1.py

```
# demande de la lettre
lettre=raw_input('Entrez une lettre\n')
# affichage du rang de la lettre
print 'le rang de la lettre '+lettre+' est',ord(lettre)-97
# fermeture du programme
raw_input('Fin du programme')
```

prog1bis.py

```
# demande du rang de la lettre
rang=raw_input('Entrez le rang de la lettre\n')
# affichage de la lettre
print 'la lettre de rang '+rang+' est',chr(int(rang)+97)
# fermeture du programme
raw_input('Fin du programme')
```

Leçon 2

prog2.py

```
# demande d'un caractere
caractere=raw_input('Entrez un caractere\n')
# test d'appartenance a l'alphabet minuscule
if 97<=ord(caractere)<=122:
    print 'le caractere '+caractere+' appartient a l\'alphabet minuscule'
else:
    print 'le caractere '+caractere+' n\'appartient pas a l\'alphabet minuscule'
# fermeture du programme
raw_input('Fin du programme')
```

Leçon 3

prog3.py

```
# demande d'un caractere
caractere=raw_input('Entrez un caractere\n')
# codage numerique du caractere
code=ord(caractere)-97
# algorithme ROT13
codechiffre=(code+13)%26
# decodage
caracterechiffre=chr(codechiffre+97)
# affichage du caractere chiffre
if 97<=ord(caractere)<=122:
    print 'le caractere chiffre par l\'algorithme ROT13 est :\n',caracterechiffre
else:
    print 'le caractere chiffre par l\'algorithme ROT13 est :\n',' '
# fermeture du programme
raw_input('Fin du programme')
```

Leçon 4

prog4.py

```
# demande d'un texte
texte=raw_input('Entrez un texte\n')
# boucle de chiffrement des caracteres du texte
print 'la liste des caracteres chiffres selon l\'algorithme ROT13 est :'
for i in range(0,len(texte)):
    caractere=texte[i]
    code=ord(caractere)-97 # codage numerique du caractere
    codechiffre=(code+13)%26 # algorithme ROT13
    caracterechiffre=chr(codechiffre+97) # decodage
    if 97<=ord(caractere)<=122: # affichage du caractere chiffre
        print caracterechiffre
    else:
        print ' '
# fermeture du programme
raw_input('Fin du programme')
```

Leçon 5

prog5.py

```
# demande du texte a chiffrer
texteclair=raw_input('Entrez un texte\n')
# initialisation du texte chiffre
textechiffre=''
# boucle de chiffrement des caracteres du texte
for i in range(0,len(texteclair)):
    caractere=texteclair[i]
    code=ord(caractere)-97 # codage numerique du caractere
    codechiffre=(code+13)%26 # algorithme ROT13
    caracterechiffre=chr(codechiffre+97) # decodage
    if 97<=ord(caractere)<=122: # affichage du caractere chiffre
        textechiffre=textechiffre+caracterechiffre
    else:
        textechiffre=textechiffre+' '
# affichage du texte chiffre
print 'le texte chiffre selon l\'algorithme ROT13 est :'
print textechiffre
# fermeture du programme
raw_input('Fin du programme')
```

Leçon 6

prog6.py

```
# ouverture en lecture du fichier messageclair.txt
fichier=open('messageclair.txt','r')
# lecture du fichier
texteclair=fichier.read()
# fermeture du fichier messageclair.txt
fichier.close()
# initialisation du texte chiffre
textechiffre=''
# boucle de chiffrement des caracteres du texte
for i in range(0,len(texteclair)):
    caractere=texteclair[i]
    code=ord(caractere)-97 # codage numerique du caractere
    codechiffre=(code+13)%26 # algorithme ROT13
    caracterechiffre=chr(codechiffre+97) # decodage
    if 97<=ord(caractere)<=122: # affichage du caractere chiffre
        textechiffre=textechiffre+caracterechiffre
    else:
        textechiffre=textechiffre+' '
# ouverture en ecriture du fichier messagechiffre.txt
fichier=open('messagechiffre.txt','w')
# ecriture dans le fichier
fichier.write(textechiffre)
# fermeture du fichier messagechiffre.txt
fichier.close()
```

prog6bis.py

```
# ouverture en lecture du fichier messagechiffre.txt
fichier=open('messagechiffre.txt','r')
# lecture du fichier
textechiffre=fichier.read()
# fermeture du fichier messagechiffre.txt
fichier.close()
# initialisation du texte clair
texteclair=''
# boucle de dechiffrement des caracteres du texte
for i in range(0,len(textechiffre)):
    caractere=textechiffre[i]
    code=ord(caractere)-97 # codage numerique du caractere
    codeclair=(code+13)%26 # algorithme ROT13
    caractereclair=chr(codeclair+97) # decodage
    if 97<=ord(caractere)<=122: # affichage du caractere clair
        texteclair=texteclair+caractereclair
    else:
        texteclair=texteclair+' '
# ouverture en ecriture du fichier messageclair.txt
fichier=open('messageclair.txt','w')
# ecriture dans le fichier
fichier.write(texteclair)
# fermeture du fichier messageclair.txt
fichier.close()
```

Leçon 7

Question préliminaire

```
>>> 22%7
1
```

on utilise la première lettre de la clef pour obtenir le décalage à appliquer à la 22^{eme} lettre du message.

prog7.py

```
# demande de la clef de chiffrement
clef=raw_input('Entrez la clef de chiffrement\n')
# lecture du fichier messageclair.txt
fichier=open('messageclair.txt','r')
texteclair=fichier.read()
fichier.close()
# initialisation du texte chiffre
textechiffre=''
# chiffrement selon l'algorithme de Vigenere
for i in range (0,len(texteclair)):
    if 97<=ord(texteclair[i])<=122:
        caractere=texteclair[i]
        code=ord(caractere)-97
        decalage=ord(clef[i%len(clef)])-97
        codechiffre=(code+decalage)%26
        caracterechiffre=chr(codechiffre+97)
        textechiffre=textechiffre+caracterechiffre
    else:
        textechiffre=textechiffre+' '
# ecriture dans le fichier messagechiffre.txt
fichier=open('messagechiffre.txt','w')
fichier.write(textechiffre)
fichier.close()
```

prog7bis.py

```
# demande de la clef de dechiffrement
clef=raw_input('Entrez la clef de dechiffrement\n')
# lecture du fichier messagechiffre.txt
fichier=open('messagechiffre.txt','r')
textechiffre=fichier.read()
fichier.close()
# initialisation du texte clair
texteclair=''
# dechiffrement selon l'algorithme de Vigenere
for i in range (0,len(textechiffre)):
    if 97<=ord(textechiffre[i])<=122:
        caractere=textechiffre[i]
        code=ord(caractere)-97
        decalage=ord(clef[i%len(clef)])-97
        codeclair=(code-decalage)%26
        caractereclair=chr(codeclair+97)
        texteclair=texteclair+caractereclair
    else:
        texteclair=texteclair+' '
# ecriture dans le fichier messageclair.txt
fichier=open('messageclair.txt','w')
fichier.write(texteclair)
fichier.close()
```

Leçon 8

prog8.py

```
# demande de la clef de chiffrement
clef=raw_input('Entrez la clef de chiffrement\n')
# lecture du fichier messageclair.txt
fichier=open('messageclair.txt','r')
texteclair=fichier.read()
fichier.close()
# initialisation du texte chiffre
textechiffre=''
# chiffrement selon l'algorithme de Vigenere
for i in range (0,len(texteclair)):
    caractere=texteclair[i]
    code=ord(caractere)
    decalage=ord(clef[i%len(clef)])
    codechiffre=(code+decalage)%256
    caracterechiffre=chr(codechiffre)
    textechiffre=textechiffre+caracterechiffre
# ecriture dans le fichier messagechiffre.txt
fichier=open('messagechiffre.txt','w')
fichier.write(textechiffre)
fichier.close()
```

prog8bis.py

```
# demande de la clef de dechiffrement
clef=raw_input('Entrez la clef de dechiffrement\n')
# lecture du fichier messagechiffre.txt
fichier=open('messagechiffre.txt','r')
textechiffre=fichier.read()
fichier.close()
# initialisation du texte clair
texteclair=''
# dechiffrement selon l'algorithme de Vigenere
for i in range (0,len(textechiffre)):
    caractere=textechiffre[i]
    code=ord(caractere)
    decalage=ord(clef[i%len(clef)])
    codeclair=(code-decalage)%256
    caractereclair=chr(codeclair)
    texteclair=texteclair+caractereclair
# ecriture dans le fichier messageclair.txt
fichier=open('messageclair.txt','w')
fichier.write(texteclair)
fichier.close()
```

Leçon 9

prog8setup.py

```
from distutils.core import setup
import py2exe
setup(zipfile=None,console=["prog8.py"])
```

prog8bissetup.py

```
from distutils.core import setup
import py2exe
setup(zipfile=None,console=["prog8bis.py"])
```

Index