TP 1 Introduction

Ilyas Bambrik

Table des matières

TP1

1. Fonction prédéfinies

Testez les fonctions suivantes depuis votre IDLE shell directement

```
Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit
32
Type "copyright", "credits" or "license()" for more information.
>>> char="C"
>>> print char
C
>>> # conversion binaire
>>> bin(ord(char))
'0b1000011'
>>> # conversion hexadecimale
>>> hex(ord(char))
'0x43'
>>> # code ascii
>>> ord(char)
67
>>> # convertor code ascii en caractere
>>> chr(67)
'C'
>>>
```

```python
def convertirEnCoedASCII(char):
    return ord(char)
def convertirRepresentationBinaire(char):
    return bin(ord(char))[2:]
def convertirRepresentationBinaireAvecPading(char,taille):
    representation_binaire=convertirRepresentationBinaire(char)
    return (taille-len(representation_binaire))*"0"+representation_binaire

def convertirRepresentationHexaDecimale(char):
    return hex(ord(char))[2:]

def convertirBinaireVersEntier(valBinaire):
    return int("0b"+valBinaire,0)

def convertirHexaVersEntier(valhexa):
    return int("0x"+valhexa,0)

def convertirEntierEnCaractere(CodeAscii):
    return chr(CodeAscii)

print convertirEnCoedASCII('C')
print convertirRepresentationBinaire('C')
print convertirRepresentationBinaireAvecPading('C',8)
print convertirRepresentationHexaDecimale('C')
print convertirBinaireVersEntier(convertirRepresentationBinaire('C'))
print convertirHexaVersEntier(convertirRepresentationHexaDecimale('C'))
print convertirEntierEnCaractere(convertirEnCoedASCII('C'))

```

2. Division en bloques

Lisez et testez le programme suivant :

```
1 def diviserEnbloque(text,t): # texte est le texte claire et t la taille du bloque
2      bloques=[] # liste de bloques
3      n=len(text) # taille du texte
4      for i in range(0,n,t): # for i= 0 ; i < n ; i+=t
5           bloques.append(text[i:min(i+t,n)]) # ajoute le bloque de i -> min(i+t,
  n)
6      return bloques # retourne la liste des bloques
7 for bloque in diviserEnbloque(raw_input(),8):
8      print bloque
```

Chiffrement par permutation

Lisez et testez le programme suivant :

```
import random # librerie de generation de chiffre aleatoire
def GenerateurDePermutation(nombreD_Elements):
     ListedePermutation=range(0,nombreD_Elements) # creer un tableau des positions des caracteres

     for i in range(nombreD_Elements-1):
          permutation_i=random.randint(0,nombreD_Elements-1)# generer un nombre aleatoire entre [i+1,nombreD_Elements-1]
          a=ListedePermutation[ ListedePermutation[i]]
          ListedePermutation[ ListedePermutation[i]]=ListedePermutation[ permutation_i] # le caractere a la position ListedePermutation[ListedePermutation[i]] sera copie' a la position ListedePermutation[ ListedePermutation[ permutation_i] ]
          ListedePermutation[permutation_i]=a # (le caractere a la position ListedePermutation[ListedePermutation[i]] sera copie' a la position ListedePermutation[ ListedePermutation[ permutation_i]] )

     return ListedePermutation # retourner la liste des permutation



TexteClaire= raw_input()
# TexteClaire="inscrire la permutation dans le HashMap (le caractere a la position i sera copie' a la position permutation_i)"
TailleTexteClaire=len(TexteClaire)
MaPermutation= GenerateurDePermutation(TailleTexteClaire)

print "Tableau de permutation = ",MaPermutation
ChiffrementParPermutation=[0]*TailleTexteClaire

for p in MaPermutation:
     ChiffrementParPermutation[p]=TexteClaire[MaPermutation[p]]

print "#"*20
print "#chiffrement"
TexteChiffre=''.join(ChiffrementParPermutation)
print TexteChiffre
print "#"*20
print "#dechiffrement"
TexteDe_chiffre=[0]*TailleTexteClaire
for p in MaPermutation:
     TexteDe_chiffre[MaPermutation[p]]=TexteChiffre[p]
print "".join(TexteDe_chiffre)
```

3. Exercice 1

On définie une fonction de chiffrement / déchiffrement comme suite :

E(X)=(X^K) = Y

D(Y)=(Y^K) = X

^ : *XOR* binaire

K : clé de chiffrement

Taille de bloques (X et Y) 8bits

Taille de la clé 8bits

Question 1

Écrivez le programme permettant de chiffrer un texte de plusieurs caractère avec cette algorithme.

On souhaite améliorer cette algorithme lors on définie une nouvelle fonction permettant de chiffrer avec une clé de 64bits :

E(X)=(X^K) = Y

D(Y)=(Y^K) = X

Taille de bloques (X et Y) 64bits

Taille de la clé 64bits

Question 2

Écrivez le programme permettant de chiffrer avec une clé de 64 bits. Afin de générer une clé aléatoire de taille 64bits K , importez le package random et faite appel à la methode random.randint(0,2**64)

Indice :

X doit être de taille 64 bits (concaténation de 8 caractères)