Cyber-Sécurité
TD5 - 2023

Les Class , Code64 et Analyse Fichiers

TD individuel
Rédiger un document Jupiter-NoteBook « .ipynb »
Déposer les exercices à rendre sur DVO à la date indiquée
uniquement un « .zip » intégrant le notebook et les fichiers associés

« La connaissance s'acquiert par l'expérience,
tout le reste n'est que de l'information.»
Albert Einstein

- **Modalité de réalisation du TD**

- **Documents Python nécessaires ou liens pour la réalisation du TD :**
  - Document Python officiel :https://docs.python.org/fr/3/
  - Document Python Fonction officiel : https://docs.python.org/3/library/functions.html
  - Index des bibliothèques officielles Python : https://docs.python.org/3/py-modindex.html
  - Communauté officielle Python : https://www.python.org/community/

- **Documents Jupiter-Lab nécessaires ou liens pour la réalisation du TD :**
  - **Ensemble des exercices de ce TD devra être réaliser avec Jupiter-lab**
  - Installation de Jupyter-Lab : https://jupyter.org/install
  - Documentation de Jupiter-lab : https://docs.jupyter.org/en/latest/
  - Utilisation de Jupiterlab pour le champ **Markdown** :
    - https://jupyterlab.readthedocs.io/en/stable/user/file_formats.html

- **Modalité de remise du TD :**
  - **Format de Jupiter-Lab**
    - Votre .zip contiendra obligatoirement :
      - L'ensemble des programmes Python dans jupyter-lab au format « ipynb »
        - Votre ipynb devra contenir obligatoirement:
          - Votre nom et prénom
          - N° du TD
          - N°de l'exercice
          - Question et réponse à l'exercice
          - Code Python, une fonction par cellule avec commentaires
          - ...
- **Liens nécessaires pour la réalisation du TD :**
  - **Test programme online : https://pythontutor.com/render.html#mode=edit**

1.1 ...

2 Exercices – Encodage en Code Base 64

- Le terme Base64 ou **Base64 de MIME** est tiré de la norme MIME (Multipurpose Internet Mail Extensions) , qui est largement utilisée pour HTTP , XML et QrCode, et a été développée à l'origine pour coder les pièces jointes aux e-mails à transmettre.
- Base64 est très utilisé pour la représentation des données binaires, ce qui les rend plus fiables pour stocker dans data brut dans des bases de données. Base64 est essentiellement utilisé pour représenter des données dans un ASCII . format de chaîne de caractère
- L'encodage Base64 est le processus de conversion des données binaires en un jeu de caractères limité de 64 caractères, ces caractères sont A-Z, a-z, 0-9, +, et /(64 caractères).
- Les données encodées en Base64 seront plus grandes que les données d'origine. Cela est dû au moyen par lequel les données binaires 8 bits sont codées dans un format qui peut être représenté en 6 bits.
    - https://fr.wikipedia.org/wiki/Base64

- Table Base64
- Le document 4648 prévoit une alternative pour un encodage compatible avec les noms de fichiers et les URI. En effet les caractères 62 (+) et 63 (/) peuvent poser problème avec certains systèmes de fichiers et dans les URI. La solution retenue consiste à remplacer ces caractères respectivement par un moins (-) et un souligné (_). Le caractère de complément reste le « = », mais peut être ignoré.

Index	Binary	Char	Index	Binary	Char	Index	Binary	Char	Index	Binary	Char
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/

2.1 Exo2_1 :Étape 1 : Écrire une fonction Python qui traduit un texte en ascii en liste binaire d'octet :

- **En Cybersécurité, certaines librairies ne pourront pas être utilisée, donc pour tous les exercices suivants aucune librairie Python n'est autorisée.**
- Étape 1 : Modifier le code ci-dessous pour permettre à un texte d'encoder en valeurs décimales du code ASCII, (c'est-à-dire a:97, b:98, etc.). table ASCII
    - .L'instruction python ord() permet d'obtenir le code ASCII

```
def enAscii(lettre):
    return
```

2.2 Exo2_2 : Étape 2 : Écrire une fonction Python qui convertie les codes Ascii en liste de code binaire : aucune librairie Python n'est autorisée.

- Étape 2 : Modifier le code ci-dessous pour permettre de convertir le résultat de l'étape 1 en binaires
- Les valeurs décimales obtenues sont converties en leurs équivalents binaires (97 : 01100001).

```
def listeBinaire(texte):
    listBinaire=[]
    ….
    return listBinaire
```

2.3 Exo2_3 : Étape 3 : Écrire une fonction Python qui concatène toutes la liste binaire est un string binaire : aucune librairie Python n'est autorisée.

- Étape 3 : Modifier le code ci-dessous pour permettre de convertir le résultat de l'étape 2 en string binaires
- Toute la liste binaire est concaténé, obtenant un grand ensemble de nombres binaires (liste en python).
- Les valeurs décimales obtenues sont converties en leurs équivalents binaires (97 : 01100001).

```
def stringBinaire(listBinaire):
    ...
    return textbinaire
```

- Résultat Attendu

```
mot= «helloo»
BinaireList =>  ['01101000', '01100101', '01101100', '01101100', '01101111', '01101111']
string binaire  0110100001100101011011000110110001101111011011111 longueur= 48 bits
```

2.4 Exo2_4 : Étape 4: Écrire la fonction Python def decoupeString(string, nb=6): qui découpe par 6 bits le stringBinaire :

- aucune librairie Python n'est autorisée.
- **Étape 4** : Compléter le code ci-dessous pour permettre :
- Toute la liste binaire est concaténé, maintenant il faut faire obtenant un grand ensemble de nombres binaires (liste en python).
  - Le grand ensemble de nombres binaires est divisé en sections égales, chaque section ne contenant que 6 bits.
  - Les ensembles égaux de 6 bits sont convertis en leurs équivalents décimaux.
    - Via l'instruction int(bin,2)

```
def decoupeString(string, nb=6):
    ...
    return listValeur
```

- Résultat Attendu

```
print("Coder =>",mot, "lg=",len(mot))
print(binaire," lg=",len(binaire))
decoupeString(binaire,6)

Coder => helloo lg= 6
```

```
011010000110010101101100011011000110111101101111  lg= 48
len= 48
011010          valeur du code base64 = 26
000110          valeur du code base64 = 6
010101          valeur du code base64 = 21
101100          valeur du code base64 = 44
011011          valeur du code base64 = 27
000110          valeur du code base64 = 6
111101          valeur du code base64 = 61
101111          valeur du code base64 = 47
nb Bits traités= 48
```

liste de string 6 bits en integer : [26, 6, 21, 44, 27, 6, 61, 47]

2.5 Exo2_5 : Étape 5: Écrire une fonction Python qui transforme le code int 6bits de la liste de string en code Base64 en vous servant de la table ci-dessous :

- En vous servant des dictionnaires Python et du tableau de conversion ci-dessous
  - tableau généré automatique par une fonction
- Ensuite, les équivalents décimaux sont convertis en leurs valeurs Base64 (c'est-à-dire 4 => E). avec l'alphabet Base64
- Attention le '=' est particulier et sera traité que dans une prochaine étape

  - .

```
table_base64 = {
   'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8, 'J': 9,
   'K': 10,'L': 11,'M': 12,'N': 13,'O': 14,'P': 15,'Q': 16,'R': 17,'S': 18,'T': 19,
   'U': 20,'V': 21,'W': 22,'X': 23,'Y': 24,'Z': 25,'a': 26,'b': 27,'c': 28,'d': 29,
   'e': 30,'f': 31,'g': 32,'h': 33,'i': 34,'j': 35,'k': 36,'l': 37,'m': 38,'n': 39,
   'o': 40,'p': 41,'q': 42,'r': 43,'s': 44,'t': 45,'u': 46,'v': 47,'w': 48,'x': 49,
   'y': 50,'z': 51,'0': 52,'1': 53,'2': 54,'3': 55,'4': 56,'5': 57,'6': 58,'7': 59,
   '8': 60,'9': 61,'+': 62,'/': 63,'=': 0,
}
print(table_base64)
```

- La fonction cherche_key_Base64(1,table_base64) devra permettre de trouver le codeBase64
  - Exemple d'exécution :
    - print(cherche_key_Base64(0,table_base64))
    - print(cherche_key_Base64(32))
    - print(cherche_key_Base64(61))
- Résultat Attendu

```
A
g
9
```

2.6 Exo2_6 : Étape 6: Écrire une fonction Python qui complète avec le caractère '=' : aucune librairie Python n'est autorisée.

- Étape 6 :
  - Si nous prenons une chaine de caractères qui n'a pas un nombre de bits multiple de 24 (l'encodage se faisant par paquet de 4 x 6 bits = 24 bits), par exemple la chaîne « Salut » :

```
 S    a    l    u    t
01010011 01100001 01101100 01110101 01110100
```

```
⇔  010100 110110 000101 101100 011101 010111 010000 ?????? (nombre de bits multiple de 24)
⇔  U    2    F    s    d    X    Q    =
```

- La procédure d'encodage est la suivante: Le dernier groupe "0100" est complété par deux "0" pour avoir une taille de 6 bits. Le symbole correspondant dans la table est la lettre "Q". Puis, pour obtenir un nombre de bits multiple de 24, il faut également ajouter un groupe de 6 bits "de remplissage", qui est symbolisé par "=". tableau généré automatique par une fonction
  - **Nous obtenons pour « *Salut* » ⇒ « *U2FsdXQ*= ».**

```
   B     o     n     j     o     u     r
  01000010 01101111 01101110 01101010 01101111 01110101 01110010
⇔  010000 100110 111101 101110 011010 100110 111101 110101 011100 100000 ?????? ???????
⇔  Q    m    9    u    a    m    9    1    c    g    =    =
                                        (g vaut 100000)
```

  - **Nous obtenons pour « Bonjour » ⇒ « Qm9uam91cg== ».**

2.7 Exo2_7 : Écrire une fonction Python EncodeBase64(texte) qui englobe l'ensemble des fonctions pour encoder un texte en Code Base64.

2.8 .

3 Exercices – Encodage en Code Base 64

3.1 .Exo3_1 : Écrire les fonctions en suivants le même principe que le décodage en Base64

- Comme le décodage Base64 est l'opposé de l'encodage Base64. En d'autres termes, il s'effectue en inversant les étapes décrites dans les exercices précédents.
- Ainsi, les étapes du décodage Base64 peuvent être décrites comme suit :
  - Chaque caractère de la chaîne est remplacé par sa valeur décimale Base64.
  - Les valeurs décimales obtenues sont converties en leurs équivalents binaires.
  - Les deux premiers bits des nombres binaires sont tronqués à partir de chacun des nombres binaires obtenus, et les ensembles de 6 bits sont combinés, formant une grande chaîne de chiffres binaires.
  - La grande chaîne de chiffres binaires obtenue à l'étape précédente est divisée en groupes de 8 bits.
  - Les nombres binaires 8 bits sont convertis en leurs équivalents décimaux.
  - Enfin, les valeurs décimales obtenues sont converties en leur équivalent ASCII.
  - Attention de ne pas oublier de traiter le caractères '='
  - Écrire la fonction DecodeBase64(codeBase64)) qui englobe toutes les fonctions et décode un code base64

3.2 .Exo 3_2 : Tester ce code qui vous permettra de valider vos fonctions python pour les exercices suivants ou vous ne pourrez pas utiliser la librairie base64 :

```
import base64
def codage_Lib_Base64(m):
```

```
    mot_bytes = m.encode("ascii")
    base64_bytes = base64.b64encode(mot_bytes)
    xxx64 = base64_bytes.decode("ascii")
    return xxx64

print("Vous devez obtenir ce résultat")
print(" En Base64=> ",codage_Lib_Base64("hello"))
print(" En Base64=> ",codage_Lib_Base64("helloZ"))
print(" En Base64=> ",codage_Lib_Base64("helloZW"))
print(" En Base64=> ",codage_Lib_Base64("helloZWO"))
print(" En Base64=> ",codage_Lib_Base64("helloo"))
```

3.3 .

4 Exercices – Fréquence d'apparition des lettres

4.1 Exo4_1 :

-

Fréquence d'apparition des lettres			
Lettre	Fréquence	Lettre	Fréquence
A	8.40 %	N	7.10 %
B	1.00 %	O	5.20 %
C	3.00 %	P	3.00 %
D	4.10 %	Q	0.90 %
E	9.00 %	R	6.50 %
F	1.10 %	S	8.00 %
G	1.20 %	T	7.00 %
H	0.90 %	U	5.70 %
I	7.30 %	V	1.30 %
J	0.30 %	W	0.05 %
K	0.05 %	X	0.40 %
L	6.00 %	Y	0.30 %
M	2.90 %	Z	0.10 %

-