CRYPTOGRAPHIE 2 : DÉCODAGE D'UN MESSAGE

1 Le logiciel Pyscripter

Nous allons voir quelques commandes du langage de programmation Python et les utiliser pour crypter un message.

Pour commencer, ouvrir **Pyscripter** dans Poste de Travail / Ressources / Portable Python / Pyscripter-Portable. Si un message d'erreur s'affiche, cliquez sur "abort" puis attendez l'ouverture du logiciel.

Dans la zone de saisie taper :

```
print("Hello world")
```

Ce petit bout de code qu'on tape dans la zone de saisie est appelé un **script**. Un script est donc une succession de commandes dont l'on va demander à l'ordinateur de les exécuter.
Cliquer sur l'icône play ou appuyer sur les touches [CTRL]+[F9]. Le résultat s'affiche dans la fenêtre "Python Interpreter".

1) Que fait la commande `print` ?

Remarque : Dans le menu : Run / Python Engine cochez la case : **Internal**.

Rang d'une lettre et réciproquement

Effacer le script précédent et taper le script suivant :

```
def rang(lettre):
        return ord(lettre) - ord("A");

print("A : ", rang("A"))
print("B : ", rang("B"))
print("C : ", rang("C"))
print("Z : ", rang("Z"))
```

2) Exécuter le script.
3) Écrire un script qui affiche le rang des lettres "R" et "Y". *Recopier votre script sur votre feuille*

```
def lettre(x):
        return chr(x + ord("A") )

print("0 : ", lettre(0))
print("1 : ", lettre(1))
print("2 : ", lettre(2))
```

4) Exécuter le script.
5) Modifier le script précédent pour qu'il affiche la lettre de rang 12.*Recopier votre script sur votre feuille.*

Fonction affine

```
a = 3
b = 1
def f(x):
        return (a*x + b) % 26

print("f(1) = ", f(1) )
print("f(2) = ", f(2) )
```

6) Exécuter le script.
7) Calculer $f(9)$, $f(12)$. Comparer les résultats avec le tableau réalisé lors de la première séance de cryptographie. *La commande* `%26` *permet de calculer le reste de la division euclidienne par 26.*

Le cryptage affine

On rappelle que pour coder une lettre L, on procède en trois étapes :

- on calcul son rang x ;
- on calcul y l'image de x par la fonction f ;
- on trouve C la lettre de rang y.

Compléter le script suivant pour que la fonction crypte réalise bien un cryptage affine associé à la fonction f définie précédemment.

```
def crypte(L):
        x = rang(     )
        y =
        C = lettre(   )
        return C

print("A -> ", crypte("A"))
print("B -> ", crypte("B"))
print("C -> ", crypte("C"))
```

Exécuter le script précédent.

```
a = 3
b = 1

code = ""
for L in "CRYPTOGRAPHIE":
        code += crypte(L)
print(code)
```

8) Tester le script précédent et expliquer ce qu'il fait.
L'instruction `for L in "CRYPTOGRAPHIE"` signifie :
Pour toutes les lettres L du mot "CRYPTOGRAPHIE" faire :

9) Comparer le résultat avec le résultat obtenu lors de la première activité.

10) Modifier les paramètres a et b pour que f soit définie par $f(x) = 7x + 3$.
Relancer le script pour coder le mot "CRYPTOGRAPHIE" et comparer avec votre réponse à l'exercice 2 de l'activité précédente.

Exercice 1. On fixe $b = 1$.

1) Avec $a = 0$, coder le mot **"CANDIDE"** en vous aidant du script précédent.
Que remarque-t-on ? Est-ce que le cryptage affine avec $a = 0$ correspond à un cryptage, au sens où l'on peut décrypter le message aussi ?

2) Avec $a = 1$, justifier que le cryptage affine est possible.

3) Avec $a = 2$, coder le mot **"CANDIDE"**.
Est-ce qu'on peut décrypter le code sans faire de choix ? En déduire que pour $a = 2$ aussi le cryptage n'est pas valide.

4) Pour a allant de 3 à 25, en codant le mot **"CANDIDE"** dites si le cryptage est valide ou pas.

5) Conjecturer une liste des valeurs que a peut prendre pour que le cryptage affine soit valide.

On admettra que le cryptage affine de fonction $f(x) = ax + b$ est possible si et seulement si a est premier avec 26. En particulier, la liste des valeurs possibles pour a que vous avez conjecturé est la bonne.

6) Justifier qu'il y a $12 \times 26 = 312$ façons différentes de coder un message à l'aide du cryptage affine $f(x) = ax + b$.

Remarque. On peut montrer qu'il y a $26 \times 25 \times 24 \times ... \times 3 \times 2 \times 1 \simeq 4 \times 10^{27}$ façons différentes de coder un mot en permutant les lettres de l'alphabet.

Décryptage Dans cette section, on allons admettre que la fonction définie dans le script suivant permet de décrypter un message codé à l'aide de la méthode de chiffrement affine de fonction $f(x) = ax + b$.

```
def decrypte(C):
        if not( 0 <= rang(C) and rang(C) < 26 ): return C
        for c in range(26):
                if (c*a) % 26 == 1:
                        break
        y = rang(C)
        x = ( c * y - c * b ) % 26
        L = lettre(x)
        return L
```

Exécuter le script précédent.

```
a = 3
b = 1
print( decrypte(crypte("A")) )
```

Exécuter le script précédent et faites le même test avec la lettre "B".

2 Analyse fréquentielle

Le premier traité exposant une procédure pour décrypter un texte codé a été écrit par Al Kindi au neuvième siècle après J.C. Sa théorie repose sur le fait que dans un texte, les lettres ont des fréquences différentes. Par exemple, en français, la fréquence de la lettre E est, selon le texte, presque toujours supérieure aux fréquences des autres lettres. Selon sa théorie, il y a donc de fortes chances pour que, dans un texte codé, la lettre qui apparaît le plus fréquemment représente un E. Les lettres les moins fréquentes représentent probablement un W, un K ou un X...
Le tableau ci-dessous exprime, en pourcentage, les fréquences moyennes , des lettres utilisées dans les textes écrits en français.

Lettre	A	B	C	D	E	F	G	H	I	J	K	L	M
fréquence en %	9,42	1,02	2,64	3,39	15,87	0,95	1,04	0,77	8,41	0,89	0,00	5,34	3,24

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
7,15	5,14	2,86	1,06	6,46	7,90	7,26	6,24	2,15	0,00	0,30	0,24	0,32

On se propose de décoder le code suivant qui a été codé avec une méthode de chiffrement affine :

```
code = "QBON LV CPEEBFV, EVN LKVCBEPVIN QPNVOS STXUTXIN EB CVIPSV, VS EVN
BISPNBON JVOSVOS STXUTXIN. UVBO POSVIITFV DXBSIV Q'VOSIV VXM. ETXPN
BAAPIJV DXV YBXE VNS XO BISPNBO; YBXE YIVSVOQ VSIV EV NVXE LKVCBEPVI
YBIJP VXM; LKBIEVN QVLEBIV DX'BX JTPON XO YBIJP ETXPN VS YPVIIV VNS XO
BISPNBO; YPVIIV NTXSPVOS DXV STXN EVN DXBSIV NTOS QVN LKVCBEPVIN.
QVSVIJPOVI EV OTJGIV QV LKVCBEPVIN."
```

Exécuter le script précédent et le suivant :

```
def crypte(L):
        x = rang(L)
        if 0<= x and x < 26:
                y = f(x)
                C = lettre(y)
        else:
                C = L
        return C

def effectif(code, lettre):
        N = 0
        for L in code:
                if L == lettre:
                        N = N + 1
        return N
```

Maintenant, pour déterminer le nombre d'apparition de la lettre "A", il suffit d'exécuter le script suivant :

```
print("A : ", effectif(code, "A"))
```

1) Déterminer le nombre d'apparitions de la lettre A dans le code ?
2) Déterminer le nombre d'apparitions des autres lettres de l'alphabet dans le code :

Lettre	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Effectif																										

3) Déterminer les 5 lettres les plus fréquentes du code.
4) Justifier que, sans doute, la lettre E correspond, une fois crypté, à la lettre V.
5) En déduire que $f(4) = 21$ et donc $4a + b \equiv 21$ modulo 26.
6) Justifier que, sans doute, la lettre A correspond une fois crypté à la lettre I, N, B ou S.
7) Distinguons 4 cas :
   a) Si $A \mapsto I$:
      i. Justifier qu'alors on a la seconde condition $f(0) = 8$.
      ii. En déduire que a et b sont solutions du système

$$\begin{cases} 4a + b & \equiv 21 \quad \text{modulo } 26 \\ b & \equiv 8 \quad \text{modulo } 26 \end{cases}$$

      iii. En déduire que $4a \equiv 13$ modulo 26.
      iv. * Justifier qu'il n'y a pas de solution.
   b) Si $A \mapsto N$:
      i. Déduire qu'on a le système suivant :

$$\begin{cases} 4a + b & \equiv 21 \quad \text{modulo } 26 \\ b & \equiv 13 \quad \text{modulo } 26 \end{cases}$$

      Le résoudre.
      ii. En déduire que la fonction affine f ne correspond pas à un chiffrement affine.
   c) Si $A \mapsto B$:
      i. Déduire qu'on a le système suivant :

$$\begin{cases} 4a + b & \equiv 21 \quad \text{modulo } 26 \\ b & \equiv 1 \quad \text{modulo } 26 \end{cases}$$

      Le résoudre.
      ii. Compléter le script suivant :

```
a =
b =

message = ""
for L in code:
        message += decrypte(    )

print(          )
```

   * Résoudre l'énigme.

```
def rang(lettre):
        return ord(lettre) - ord("A");

def lettre(x):
        return chr(x + ord("A") )

a = 3
b = 1
def f(x):
        return (a*x + b) % 26

def crypte(L):
        x = rang(L)
        if 0<= x and x < 26:
                y = f(x)
                C = lettre(y)
        else:
                C = L
        return C

def decrypte(C):
        if not( 0 <= rang(C) and rang(C) < 26 ): return C
        for c in range(26):
                if (c*a) % 26 == 1:
                        break
        y = rang(C)
        x = ( c * y - c * b ) % 26
        L = lettre(x)
        return L

code = "QBON LV CPEEBFV, EVN LKVCBEPVIN QPNVOS STXUTXIN EB CVIPSV, VS EVN
BISPNBON JVOSVOS STXUTXIN. UVBO POSVIITFV DXBSIV Q'VOSIV VXM. ETXPN
BAAPIJV DXV YBXE VNS XO BISPNBO; YBXE YIVSVOQ VSIV EV NVXE LKVCBEPVI
YBIJP VXM; LKBIEVN QVLEBIV DX'BX JTPON XO YBIJP ETXPN VS YPVIIV VNS XO
BISPNBO; YPVIIV NTXSPVOS DXV STXN EVN DXBSIV NTOS QVN LKVCBEPVIN.
QVSVIJPOVI EV OTJGIV QV LKVCBEPVIN."
```