Programmation et cryptographie cours n°1
Fiche de Clubalkindi

Itaï

Septembre 2021

1 Introduction

Maintenant que vous connaissez de nombreuses methodes de chiffrements, vous avez sans doute du remarquer qu'il était très fastidieux d'appliquer ces methodes à la main. Contrairement au concours Alkindi ou vous n'avez le droit qu'à un papier un crayon et votre cerveau, dans la "vraie" vie tout le chiffrement se passe par ordinateur.

Nous allons voir dans ces cours des implémentations des principaux chiffrements en programmation, et peut-être aller encore plus loin !

ℹ Les codes en JavaScript pour ceux qui préféreraient ce language au Python sont disponibles en annexe à la fin du cours à la partie 5
Si vous n'avez aucune connaissance en programmation, je vous conseille personnellement de poursuivre en Python car sa syntaxe est beaucoup plus simple, et il est plus proche du pseudo-code que le JS(Javascript)

🦙 Au vu de la mise en page, vous allez sans doute avoir du mal à copier-coller les codes si vous voulez essayer, j'ai donc tout mis sur mon github que vous pouvez retrouver ici

2 Chiffrement par code de César et substitution mono-alphabétique

2.1 Code de César

Nous allons d'abord voir une implémentation en python (qui est très proche du pseudo-code).

Premièrement, prenons en entrée le message qu'il faut chiffrer, en python cela s'écrit :

```
message=input("Saisissez un message a chiffrer : ")
```

On demande également le décalage (sous la forme d'une lettre, un décalage de "E" signifie que le "A" va donner un "E")

```
decalage=input("Saisissez le decalage : ").upper()
# upper() met la lettre en majuscule (pour simplifier le code ensuite)
```

On va ensuite formater le message pour avoir un message qui comporte uniquement des majuscules sans accent et sans espaces. On va utiliser pour cela la librairie `unidecode`

```
import unidecode
message=unidecode.unidecode(message).replace(" ","").upper()
# unidecode.unidecode() va supprimer les accents
# replace(" ","") va remplacer les espaces par rien, donc les supprimer
# upper() va mettre le texte en majuscules
```

On va ensuite s'occuper de chiffrer le message. Pour cela, on crée une fonction `chiffre()` qui prend en paramètre une lettre et le décalage, et sort la lettre chiffrée.

```
def chiffre(lettre,decal):
    position_lettre=ord(lettre)-65
    # ord() donne la valeur ascii du charactere. On enleve donc 65 qui
    correspond au codage ascii de A.
    position_decal=ord(decal)-65
    position_finale=(position_lettre+position_decal)%26
    # on aditionne les deux, et si l'on depasse 26, on revient a 0.
    return chr(65+position_finale)
    # chr() donne un charactere en fonction de son chiffrement ascii.
```

On va ensuite boucler sur le texte initial et appliquer cette fonction à chacune des lettres.

```
message_chiffre=""
for i in message:
    message_chiffre+=chiffre(i,decalage)
print(message_chiffre)
```

On peut ensuite tester le programme pour vérifier que tout fonctionne bien :

```
>>> message=input("Saisissez un message a chiffrer : ")
Saisissez un message a chiffrer : Ceci est un message à chiffrer par ce
     programme génial
>>> decalage=input("Saisissez le decalage : ").upper()
Saisissez le decalage : k
>>> import unidecode
>>> message=unidecode.unidecode(message).replace(" ","").upper()
>>> message
'CECIESTUNMESSAGEACHIFFRERPARCEPROGRAMMEGENIAL'
>>> def chiffre(lettre,decal):
...     position_lettre=ord(lettre)-65
...     position_decal=ord(decal)-65
...     position_finale=(position_lettre+position_decal)%26
...     return chr(65+position_finale)
...
>>> message_chiffre=""
>>> for i in message:
...     message_chiffre+=chiffre(i,decalage)
...
>>> print(message_chiffre)
MOMSOCDEXWOCCKQOKMRSPPBOBZKBMOZBYQBKWWOQOXSKV
```

C'est parfait ! Sur un petit message comme celui-ci ce n'est pas tellement plus rapide avec un programme, mais celui-ci peut être utilisé plein de fois sur des messages beaucoup plus long et évite aussi de faire des erreurs. De plus, pour des chiffrements plus complexes, cela gagne un temps considérable.

2.2 Substitution mono-alphabétique

Pour la substitution monoalphabétique, c'est en gros le même principe. On va juste utiliser un dictionnaire pour associer à une lettre son chiffré. Un dictionnaire (en python) est un objet désordonné constitué de paires de clefs/objets.

```
dictionnaire=dict()
dictionnaire["A"]="B"
# ici la clef est "A" et l'objet associé "B".
dictionnaire["B"]="C"
# dictionnaire vaut donc {"A":"B","B":"C"} ou {"B":"C","A":"B"}
```

Pour récupérer l'objet associé à une clef, on va faire `objet=dictionnaire[clef]` On va donc boucler sur le message et pour chaque charactère retourner l'objet associé dans le dictionnaire. Le code ressemblera donc à ça :

```
message=input("Saisissez un message a chiffrer : ")
import unidecode
message=unidecode.unidecode(message).replace(" ","").upper()
# comme précedemment
substitution={"A":"M","B":"K"}
# à compléter en recopiant la substitution
message_final=""
for i in message:
    message_final+=substitution[i]
print(message_final)
```

On peut par exemple essayer avec comme dictionnaire `{'A': 'M', 'C': 'O', 'B': 'F', 'E': 'H', 'D': 'A', 'G': 'Z', 'F': 'Y', 'I': 'I', 'H': 'U', 'K': 'T', 'J': 'K', 'M': 'J', 'L': 'R', 'O': 'W', 'N': 'L', 'Q': 'X', 'P': 'P', 'S': 'N', 'R': 'C', 'U': 'V', 'T': 'S', 'W': 'D', 'V': 'B', 'Y': 'E', 'X': 'G', 'Z': 'Q'}`

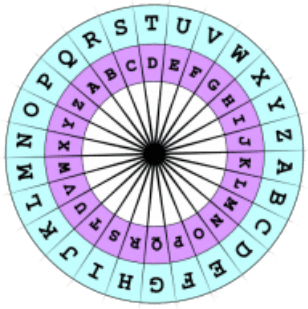
Cela donnerait :

```
>>> message=input("Saisissez un message a chiffrer : ")
Saisissez un message a chiffrer : message à chiffrer par substitution
    monoalphabétique
>>> import unidecode
>>> message=unidecode.unidecode(message).replace(" ","").upper()
>>> substitution={'A': 'M', 'C': 'O', 'B': 'F', 'E': 'H', 'D': 'A', 'G
    ': 'Z', 'F': 'Y', 'I': 'I', 'H': 'U', 'K': 'T', 'J': 'K', 'M': 'J',
     'L': 'R', 'O': 'W', 'N': 'L', 'Q': 'X', 'P': 'P', 'S': 'N', 'R': '
    C', 'U': 'V', 'T': 'S', 'W': 'D', 'V': 'B', 'Y': 'E', 'X': 'G', 'Z
    ': 'Q'}
>>> message_final=""
>>> for i in message:
...     message_final+=substitution[i]
...
>>> print(message_final)
JHNNMZHMOUIYYCHCPMCNVFNSISVSIWLJWLWMRPUMFHSIXVH
```

3 Et après le chiffrement ?

Le déchiffrement !

3.1 Déchiffrement du code de César



3.1.1 En connaissant la clef

Le déchiffrement en connaissant la clef pour le code de César est très similaire à son chiffrement. En effet, tout le monde sait (ou devrait savoir) que pour déchiffrer un César, il suffit de faire le décalage en sens inverse !
Grosso modo, on va donc juste devoir remplacer le + par un − !

Reprenons donc notre code du chiffrement ...

```
1  message=input("Saisissez un message a chiffrer : ")
2  decalage=input("Saisissez le decalage : ").upper()
3  import unidecode
4  message=unidecode.unidecode(message).replace(" ","").upper()
5  def chiffre(lettre,decal):
6       position_lettre=ord(lettre)-65
7       position_decal=ord(decal)-65
8       position_finale=(position_lettre+position_decal)%26
9       return chr(65+position_finale)
10 message_chiffre=""
11 for i in message:
12      message_chiffre+=chiffre(i,decalage)
13
14 print(message_chiffre)
```

On va faire quelques modifications d'ordre esthétiques (comme renommer des variables et des fonctions) ce qui ne va pas changer quoi que ce soit au code, puis on va modifier une toute petite chose pour déchiffrer :
Le + de la ligne 8 va être transformé en −. Et c'est tout !

Le code sera donc :

```
1  message_chiffre=input("Saisissez un message a déchiffrer : ")
2  clef=input("Saisissez la clef : ").upper()
3  import unidecode
4  message_chiffre=unidecode.unidecode(message_chiffre).replace(" ","").
       upper()
5  def dechiffre(lettre,clef):
6       position_lettre=ord(lettre)-65
7       position_clef=ord(clef)-65
8       position_finale=(position_lettre-position_clef)%26
9       return chr(65+position_finale)
10 message_dechiffre=""
11 for i in message_chiffre:
12      message_dechiffre+=dechiffre(i,clef)
13
14 print(message_dechiffre)
```

Comme d'habitude, on vérifie que ça marche bien (avec le message de tout à l'heure)

```
1  Saisissez un message a déchiffrer :
       MOMSOCDEXWOCCKQOKMRSPPBOBZKBMOZBYQBKWWOQOXSKV
2  Saisissez la clef : k
3  CECIESTUNMESSAGEACHIFFRERPARCEPROGRAMMEGENIAL
```

Génial !

3.1.2 Sans connaitre la clef

Imaginons maintenant que Alice et Bob aient réussi à récupérer un message d'un dangereux réseau de malfaiteurs planifiant la plus grande cyberattaque de tous les temps. On sait qu'ils communiquent par César, mais on ne connait pas le décalage. On veut donc déchiffrer leur message.

Alice et Bob cherchent des idées pour le déchiffrer, et Bob va dire à Alice :
— Et si on essayait toutes les clefs possibles ? Il n'y a que 26 lettres dans l'alphabet soit 26 clefs possibles. Avec un ordinateur, tout cela va si vite !
— Évidemment cela va marcher, mais n'y a-t-il pas moyen de ruser un peu mon cher Bob ?
— Je ne vois vraiment pas, tu n'aurais pas un indice pour moi ?
— Réflechis un peu à comment est construite la langue française ... Par exemple quelles lettres ai-je beaucoup utilisé dans ma phrase ?
— Mais oui ! Le E ! Il suffit de décaler la lettre la plus fréquente dessus !
— Bravo Bob !

Nous allons donc commencer par essayer de faire ce que disait Bob (même si, il a raison avec un ordinateur il est parfaitement possible de tout essayer sur un chiffrement aussi simple)
Puis nous nous interesserons dans un second temps à l'idée d'Alice ce qui va nous permettre d'aborder le déchiffrement par fréquences

L'idée de Bob

Ici on ne va pas faire compliqué, on va juste essayer toutes les lettres :p
On reprend le code précédent en bouclant sur toutes les clefs et puis on affiche tous les messages en sortie (oui oui c'est moche)
Cela va donc donner quelque chose de ce genre :

```
message_chiffre=input("Saisissez un message a déchiffrer : ")
alphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZ"

import unidecode
message_chiffre=unidecode.unidecode(message_chiffre).replace(" ","").
    upper()
def dechiffre(lettre,clef):
     position_lettre=ord(lettre)-65
     position_clef=ord(clef)-65
     position_finale=(position_lettre-position_clef)%26
     return chr(65+position_finale)
for lettre in alphabet:
    message_dechiffre=""
    for i in message_chiffre:
        message_dechiffre+=dechiffre(i,clef)
    print(message_dechiffre)
```

L'idée d'Alice

Pour faire le programme que nous suggère Alice, on va devoir compter les lettres les plus fréquentes pour repérer le E (celui qui a la fréquence la plus grande).

⚠️ Ceci ne fonctionnera pas toujours ! En effet, dans la règle générale le E est la lettre la plus fréquente en français (et de loin !) mais il peut y avoir des problèmes notamment avec des textes courts (si j'envoie `Coucou ! ça va` le E n'est naturellement pas la lettre la plus fréquente). Dans certaines œuvres comme *La Disparition* de Georges Perec, il y a aussi beaucoup moins de E (voir aucun dans le cas présent)

L'idée de Bob n'est donc pas si idiote en fin de compte, mais on va quand même coder l'idée d'Alice qui ne va donc marcher que dans des textes suffisamment long (au moins 100 à 200 charactères).
On va donc essayer de determiner la lettre la plus fréquente. Mais comment va-t-on faire ?

Il y a pour cela de nombreuses possibilités, mais ici on va en aborder une plutot simple, même si ce n'est pas la plus optimisée (en terme de mémoire et de temps d'éxecution).
Faisons une fonction `plus_frequente` qui va prendre en entrée un texte et ressortir la lettre la plus frequente.
On va tout d'abord boucler sur l'alphabet et compter pour chaque lettre son taux d'apparition à l'aide de la fonction `count` de la classe `string`. On prend ensuite le maximum de ces frequences et le tour est joué !
On peut le faire comme ceci par exemple (deux variantes):

```
def plus_frequente_1(texte):
    # on initialise une variable qui va contenir la fréquence maximale
    frequence_max=0
    alphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    for i in alphabet:
        # pour chaque lettre de l'alphabet, on compte sa fréquence
        if texte.count(i)>frequence_max:
            # si elle est plus grande que fréquence_max
            frequence_max=texte.count(i)
            # on définie fréquence_max sur cette frequence
            lettre=i
            # et lettre sur ce charactère
    # on retourne finalement la lettre la plus frequente
    return lettre
```

Ici c'est plus compliqué, on va des dictionnaires comme vu précédemment. (Si vous ne comprenez pas le code, c'est normal 😅, mais il faut que vous ayez bien compris le premier !) En gros, on va associer a chaque lettre sa fréquence d'apparition et on va prendre la fréquence la plus grande.

```
def plus_frequente_2(texte):
    alphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    dictionnaire={texte.count(i):i for i in alphabet}
    return dictionnaire[max(dictionnaire)]
```

On va finalement trouver le décalage entre la lettre la plus fréquente et le E ce qui va nous donner la clef !
Pour trouver ce décalage, on va simplement prendre la différence entre les deux

charactères.
On peut faire quelque chose comme ceci :

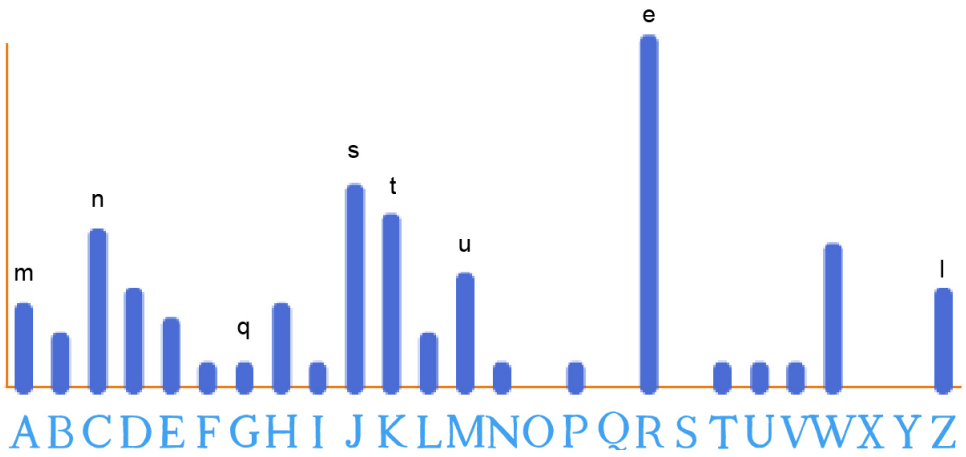
```
message_chiffre=input("Saisissez un message a déchiffrer : ")
import unidecode
message_chiffre=unidecode.unidecode(message_chiffre).replace(" ","").
    upper()
plus_frequente=plus_frequente_2(message_chiffre)
position_lettre=ord(plus_frequente)-65
position_e=4
position_finale=(position_lettre-position_e)%26
clef=chr(position_finale+65)
print(clef)
```

On essaie ?

```
Saisissez un message a déchiffrer :
    MOMSOCDEXWOCCKQOKMRSPPBOBZKBMOZBYQBKWWOQOXSKV
K
```

Parfait 🎉 !
Si vous avez bien suivi jusque là, vous êtes capable de faire un programme qui peut déchiffrer un code de César et afficher le message déchiffrer, je vous le laisse en exercice 😉

3.2 Le déchiffrement mono-alphabétique



Dans cette partie, on va se contenter de faire le déchiffrement en connaissant la clef, et cela pour trois principales raison :
- Le code et toutes les explications sont très longues et le cours fait déjà 9 pages donc ce sera pour une prochaine fois 😉
- Il faut trouver une bonne méthode et comme on l'a vu l'analyse fréquentielle n'est pas précise du tout (déjà pour le E, mais alors pour le W ...)
- Si on veut faire un programme abouti qui permet de déchiffrer presque à coup sur, il va falloir utiliser de nombreux outils comme la fréquence de bigrammes,

voire même un dictionnaire pour reconnaître les mots ce qui complique énormément le programme !

On va donc faire exactement comme le César, en faisant juste la permutation inverse notée σ^{-1} telle que si on note E l'alphabet et σ la permutation initiale :

$$\sigma^{-1}(\sigma(E)) = E$$

. Cette permutation est donc notée la permutation réciproque.
Avec un dictionnaire en python, il est très facile de trouver la permutation inverse. On a juste à inverser les clefs (`keys`) et les objets (`objects`). On va pour cela utilise juste la ligne suivante :

```
permutation_inverse={objet:clef for clef,objet in permutation.items()}
```

Le code va donc être le même que pour chiffrer en rajoutant cette petite ligne et en modifiant toujours quelques noms de variables pour faire joli 😄. Le code final va donc être :

```
message=input("Saisissez un message a déchiffrer : ")
import unidecode
message=unidecode.unidecode(message).replace(" ","").upper()
permutation={"A":"M","B":"K"}
permutation_inverse={objet:clef for clef,objet in permutation.items()}
# à compléter en recopiant la permutation
message_final=""
for i in message:
    message_final+=permutation_inverse[i]
print(message_final)
```

4 Conclusion

Merci beaucoup et bravo à vous d'avoir réussi à finir ce cours 👏 !
Vous avez découvert le chiffrement et déchiffrement à l'aide d'un ordinateur et d'un programme Python, et êtes maintenant capable d'utiliser Python pour faire de petits programmes simples.

Si vous trouvez des erreurs dans le cours ou avez une petite remarque, vous pouvez me contacter via le forum du site ou sur discord, mon pseudo est : itai#2042.

Merci encore, et à bientot pour de nouvelles aventures ! 👋

****

*****

***

*

5 Annexe : codes en Javascript

Chiffrement de César

```
function chiffre_message(message,decalage){
    message_final="";
    for(i of message){
        position_lettre=i.charCodeAt(0)-65;
        position_decalage=decalage.charCodeAt(0)-65;
        position_finale=(position_lettre+position_decalage)%26;
        message_final+=String.fromCharCode(position_finale+65);
    }
    return(message_final);
}
```

Chiffrement monoalphabétique

```
function chiffre_message(message,substitution){
    message_final="";
    for(i of message){
        message_final+=substitution[i];
    }
    return(message_final);
}
```

Déchiffrement de César connaissant la clef

```
function dechiffre_message(message,decalage){
    message_final="";
    for(i of message){
        position_lettre=i.charCodeAt(0)-65;
        position_decalage=decalage.charCodeAt(0)-65;
        position_finale=(position_lettre-position_decalage)%26;
        message_final+=String.fromCharCode(position_finale+65);
    }
    return(message_final);
}
```

Déchiffrement de César sans connaitre la clef : methode de Bob

```
function dechiffre_message(message){
    alphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    for(j of alphabet){
        message_final="";
        for(i of message){
            position_lettre=i.charCodeAt(0)-65;
            position_decalage=decalage.charCodeAt(0)-65;
            position_finale=(position_lettre-position_decalage)%26;
            message_final+=String.fromCharCode(position_finale+65);
        }
        console.log(message_final);
    }
}
```

Déchiffrement de César sans connaitre la clef : methode d'Alice

```
function plus_frequente(message){
    frequence_max=0;
    alphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    for(i of alphabet){
        if((message.split(i).length - 1)>frequence_max){
            frequence_max=(message.split(i).length - 1);
            lettre=i;
        }
    }
    return(lettre);
}
function dechiffre_message(message){
    message_final="";
    decalage=plus_frequente(message);
    for(i of message){
        position_lettre=i.charCodeAt(0)-65;
        position_decalage=decalage.charCodeAt(0)-65;
        position_finale=(position_lettre-position_decalage)%26;
        message_final+=String.fromCharCode(position_finale+65);
    }
    return(message_final);
}
```

Déchiffrement monoalphabétique

```
function chiffre_message(message,substitution){
    subsitution_reciproque=Object.fromEntries(Object.entries(
    substitution).map(([ key, val ]) => [ val,key ]));
    message_final="";
    for(i of message){
        message_final+=substitution_reciproque[i];
    }
    return(message_final);
}
```