Cryptographie classique

[1]
```
from IPython.display import HTML
import urllib.request
HTML(url="https://deptinfo-ensip.univ-poitiers.fr/FILES/NB/files/nb2.cs
```

Chaînes de caractères

- Les chaînes de caractères python sont des séquences non mutables (On peut utiliser la notation `[.]`, et on ne peut pas modifier un caractère d'une chaîne. La documentation officielle est ici : Documentation du type `str`
- Le module `string` peut être utile : Documentation du module `string`
- Le notebook suivant Notebook sur les chaînes contient la plupart des informations dont vous aurez besoin (mais peut être pas toutes).

Nettoyage des chaînes

Écrivez une fonction qui prend un texte, le met en majuscules, remplace toutes les lettres accentuées par leur équivalent non accentué, et retire toutes les ponctuations et espacements.

Cette fonction devra être appliquée à tout texte, avant de chiffrer. Ou bien, lorsque vous aurez un cryptogramme, vous pourrez l'utiliser pour retirer les espaces parasites et les retours chariot.

La fonction doit renvoyer la chaîne nettoyée. La chaîne résultat ne contiendra donc que certains des 26 caractères suivants : `ABCDEF...WXYZ`.

Vous pourriez avoir besoin des éléments suivants :

- méthode `str.upper`
- méthode `str.translate`
- méthode `str.join`
- chaîne prédéfinie `string.ascii_uppercase`
- conversion d'une chaîne en liste avec `list(s)`

Testez votre fonction avec des exemples complexes (des chaines contenant des majuscules, des minuscules, des lettres accentuées, les espace etc...)

```
[2]    # Votre travail ici
```

Chiffre de César

Le chiffre de César consiste à décaler les lettres de l'alphabet. La valeur du décalage constitue la clé. Par exemple, si on décale de 4, A devient E, B devient F, V devient Z, W devient A (l'alphabet est cyclique) etc...

Faites une fonction qui prend en paramètres une chaîne et un décalage, et renvoie la chaîne chiffrée. Faites en sorte qu'un décalage positif ou négatif fonctionne.

Votre fonction devrait se comporter ainsi :

```
cesar("Il ne respecte pas absolument l'orthographe, c'est-à-dire l'art
d'écrire correctement les mots", 12)
UXZQDQEBQOFQBMEMNEAXGYQZFXADFTASDMBTQOQEFMPUDQXMDFPQODUDQOADDQOFQYQZFXQEYA
FE
```

Vous pourriez avoir besoin des éléments suivants :

- chaîne prédéfinie `string.ascii_uppercase`
- fonction `ord()`
- fonction `chr()`
- opérateur modulo `%`
- méthode `str.join`

```
[3]    # Votre travail ici
```

Voici un texte chiffré avec le chiffre de césar, mais vous ne connaissez pas le décalage. Trouvez une (ou deux) stratégies pour le déchiffrer :

```
VUWVZ ZLKLL UMPUK LJLZH YKLZS LAAYL ZHJPJ LYVUL AZHJV YYLZW
VUKHU JLHCL JZLZH TPZZB YZLZH MMHPY LZKVT LZAPX BLZPS FLTWS
VFHPA WVBYS LZJOV ZLZAV BAHMH PAZLJ YLALZ BULLZ WLJLK LJOPM
MYLXB PLUYL UKHPA SLZLU ZPUPU ALSSP NPISL SLZSL AAYLZ LAHUA
KPZWV ZLLZK LTHUP LYLHU LWVBC VPYQH THPZM VYTLY BUTVA LAXBP
JVUZP ZAHPA QLSLK PZWVB YJLBE XBPCV BKYVU ASLZK LJOPM MYLYH
JOHUN LYSLY HUNKL ZSLAA YLZKH UZSHS WOHIL ALULJ YPCHU ASHXB
HAYPL TLWVB YSHWY LTPLY LJLZA HKPYL SLKWV BYSHL AHPUZ PKLZB
PAL
```

```
[4]    # Votre code ici
```

Chiffre de Vigénère

Le chiffre de Vigénère est une des plus anciennes substitutions polyalphabétiques utilisant un mot clé. On commence par choisir un mot-clé, par exemple `PYTHON`. Puis, on superpose le

message à chiffrer (par exemple : "Basé sur le carré de Thithème") et le mot-clé :

```
BASESURLECARREDETRITHEME
PYTHONPYTHONPYTHONPYTHON
```

Enfin on chiffre chaque lettre de la première ligne en utilisant la lettre de la clé située juste en dessous, qui donne un décalage : A ne décale pas, B décale de 1, C décale de 2... Z décale de 25. Ce qui donne :

```
BASESURLECARREDETRITHEME
PYTHONPYTHONPYTHONPYTHON
------------------------
QYLLGHGJXJOEGCWLHEXRALAR
```

Écrivez une fonction qui prend en paramètres un message chiffré, et un mot-clé, puis **déchiffre** en utilisant l'algorithme de Vigénère.

[5]
```
# Votre code ici
```

Utilisez votre programme pour déchiffrer le message suivant (heuseusement, vous avez intercepté la clé, il s'agit du mot `SNEFFELS`).

```
EBRTS GWWJR PJAED WKYYS JXEWK
CVNYY YWXBV YJPZM HRIYU EDKSF
SNLRP MKRQJ SXPFJ RZZJP PKHEI
RNICW KCELJ WOMDV ZWJEF NWEWT
IIWSK RGTSH PUWYP JIYQS MKXNY
VPADQ IHTYG JAGYS JWZJL RHJRE
NMDRU ZNJLA KNMYF PZWAY PJKJP
LVHRJ YENZW QISHV P
```

Analyse fréquentielle

L'analyse fréquentielle est une méthode de cryptanalyse qui consiste à utiliser le fait que les lettres n'ont pas toutes la même fréquence d'apparition (en français, par exemple, le `E` a une fréquence d'apparition de 0.172, et le `K` a une fréquence d'apparition de 0.005).

Dans le cas d'un chiffrement monoalphabétique, il est parfois laborieux, mais toujours assez facile de déchiffrer un texte. C'est particulièrement vrai pour le chiffre de César, puisque déchiffrer une des lettres revient à connaître le décalage et donc à déchiffrer toutes les autres.

Écrivez une fonction qui prend en paramètre un texte, le nettoie puis renvoie la liste des fréquences dans l'ordre alphabétique.

Par exemple, pour l'entrée 'AABBCDEEEE', votre fonction devrait renvoyer :

```
[0.2, 0.2, 0.1, 0.1, 0.4, 0, 0, 0, ...]
```

[6]
```
# Votre travail ici
```

Utilisez cette fonction pour déterminer le décalage utilisé dans un chiffrement de césar et déchiffrer automatiquement un message. Vous devez écrire une fonction (`autocesar`) qui s'utilise ainsi :

```
s="""OJCUT XFNIJ STZAJ FZQJA JQNSF ZKJZF UWJXF ATNWF
     ZLRJS YJQFH MFQJZ WRFNX WNJSS JUFWZ Y"""
d = autocesar(s)
print(d)
r = cesar(s, -d)
print(r)
```

Affichera :

```
5
JEXPOSAIDENOUVEAULEVELINAUFEUAPRESAVOIRAUGMENTELACHALEURMAISRIENNEPARUT
```

[7]
```
# Votre travail ici
```

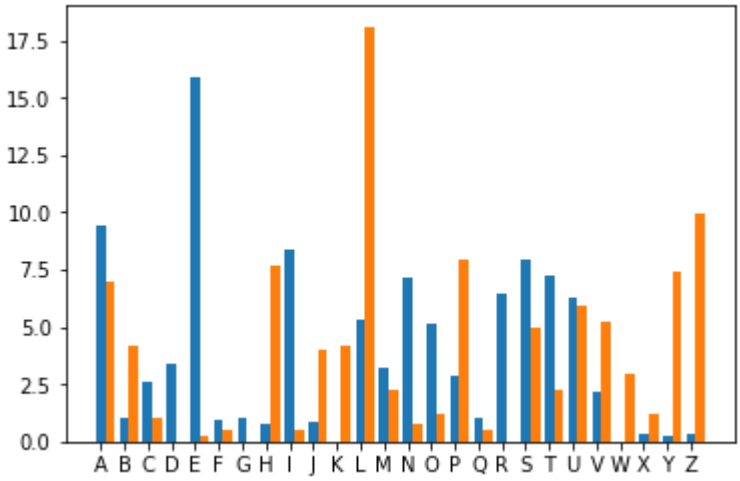
Utilisez votre fonction pour déchiffrer le message suivant :

```
GUCMN YHYTJ IOLWY LNUCH KOYFY
MJYWC GYHKO YHIOM UPIHM MIOMF
YMSYO RUJJU LNCYH NUFUW UNYAI
LCYFU JFOMM CGJFY XYFUW LSJNI
ALUJB CY
```

L'analyse fréquentielle est un outil de cryptanalyse puissant pour déchiffrer des substitutions monoalphabétiques. Mais elle est *a priori* inopérente dans le cas de substitutions polyalphabétiques, qui ont pour effet de «lisser» l'histogramme des fréquences des lettres.

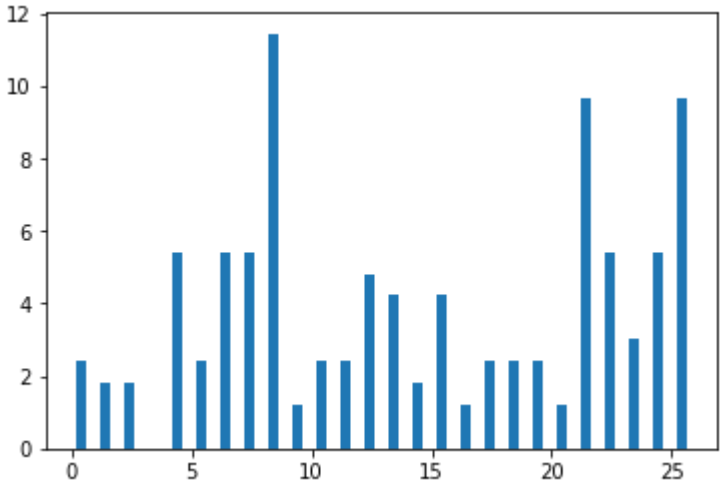
Au passage, voici un moyen de tracer des histogrammes. La figure ci-dessous trace l'histogramme d'un texte chiffré avec le chiffre de César en vis à vis de l'histogramme des fréquences des lettres en français.

[15]
```
import matplotlib.pyplot as plt
import string
franc = [9.42, 1.02, 2.64, 3.39, 15.87, 0.95, 1.04, 0.77, 8.41, 0.89, 0
crypt = [6.95, 4.22, 0.99, 0.0, 0.25, 0.5, 0.0, 7.69, 0.5, 3.97, 4.22, 1
plt.bar(range(26), franc, width=0.4, tick_label=list(string.ascii_upper
plt.bar([_+0.4 for _ in range(26)], crypt, width=0.4)
plt.show();
```



Sur la figure précédente, on devine assez facilement le décalage de César employé. En revanche, avec une substitution polyalphabétique, on pourrait obtenir ceci :

[9]
```
franc = [9.42, 1.02, 2.64, 3.39, 15.87, 0.95, 1.04, 0.77, 8.41, 0.89, 0
crypt = [2.41, 1.81, 1.81, 0.0, 5.42, 2.41, 5.42, 5.42, 11.45, 1.2, 2.4
# plt.bar(range(26), franc, width=0.4, tick_label=list(string.ascii_upp
plt.bar([_+0.4 for _ in range(26)], crypt, width=0.4)
plt.show();
```



Indice de coïncidence

La première étape pour déchiffrer la méthode de vigénère sans connaître la clé est de deviner sa longueur. Pour cela, on peut utiliser l'indice de coïncidence, qui mesure la probabilité que deux lettres tirées au hasard dans un texte soient identiques. L'expression de l'indice de coïncidence est :

$$IC = \sum_{i=A}^{i=Z} \frac{n_i(n_i - 1)}{n(n-1)}$$

avec n_A le nombre de A, n_B le nombre de B... et n le nombre total de lettres.

Sa valeur habituelle pour un texte français est déduit des fréquences des lettres. Il vaut environ 0.074 pour un texte assez long. Notons que si un texte particulier est chiffré par un chiffrement monoalphabétique, cela ne change pas la valeur de l'indice de coïncidence (on le voit dans le calcul, la somme est commutative...)

Pour un texte formé de lettres aléatoires, l'indice de coïncidence vaudra environ 0.0385.

Une substitution polyalphabétique a tendance à faire descendre l'indice de coïncidence et le rapproche de celui d'un texte aléatoire.

Enfin, le chiffre de Vigénère est formé de sous-textes (autant de sous textes qu'il y a de lettres dans la clé), qui sont tous des substitutions monoalphabétiques différentes.

Si on reprend l'exemple précédent :

```
BASESURLECARREDETRITHEME
PYTHONPYTHONPYTHONPYTHON
------------------------
```

Écrivez une fonction qui renvoie l'indice de coïncidence d'un texte.

```
[10]   # Votre travail ici
```

Il est assez facile d'extraire une sous-chaîne d'une chaîne de caractère en utilisant les slices. Si `s` est une chaîne, `s[2::5]` est la sous-chaîne obtenue en prenant une lettre sur 5, à partir de la troisième.

Utilisez l'indice de coïncidence et les sous-chaînes pour deviner la longueur de la clé utilisée pour chiffrer le texte suivant (avec la méthode de Vigénère) :

```
MHMGU LRVAN THAUI IGOZA XNRAY EGSEC VDMDZ LXETZ
MPVDU EIEMI JKUSV MXEDW ZTMGG SJSHI VRXED WZAXW
DICXQ RRAFD VDMPV FPUCW FOLLO EHMQV FMEVF MPRZW
LPRJQ XVEBL VKGPG GEOEK CUZDE OZLZE LXTEL JQSUM
YAKAZ LYWGR VVQVF KXETG ZSMGG SESGR ZWLQL SPIIW
FOLLN AJMZP VLUTD GFALL QMGKQ TCSUG LAXLV HMRKA
DAZLQ NLFOL ZFPIC HAUIE MRHMQ RLFQH VMDEV LPEDA
QLYWG RVVGD ZFQR
```

Puis, une fois connue la longueur de la clé, trouvez chacune des lettres de cette clé, en utilisant les sous-textes, et la fonction `autocesar` déjà réalisée.

Vous prendrez bien un peu de transpositions ?

Si une substitution change un caractère en un autre, une transposition conserve les caractères, mais modifie leur ordre dans le texte.

La transposition suivante est basée sur des propriétés arithmétiques. Pour un message de longueur n, on choisit un entier p premier avec n. Puis on recopie le message en prenant une lettre sur p (et en cyclant). Par exemple si n vaut 26 et p vaut 7, le message :

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

sera chiffré en :

```
AHOVCJQXELSZGNUBIPWDKRYFMT
```

Observez l'exemple attentivement, en particulier ce qui se passe quand on arrive à la fin du message et qu'il faut repartir au début.

Un tel message peut être déchiffré avec la même procédure, mais en changeant p par son **inverse modulo n**, c'est à dire le nombre q inférieur à n tel que le reste de la division de p x q par n soit égale à 1. Ici la valeur qui convient est 15 (7 x 15 = 105, et 105 = 26 * 4 + 1)

Écrivez une fonction qui prend en paramètres un message et un pas (p ou q... c'est pareil) et transpose le message. Votre fonction peut donc aussi bien chiffrer que déchiffer (attention à la valeur de n, c'est la longueur du message **après** nettoyage...)

[11]
```
# Votre travail ici
```

Le message suivant a été intercepté :

```
CATDO SRELO VAIQL ECMCN ERMNA DUDSU SPUEH EELCH
NCREB RELTU DAMEP RUEEO EAALA ETBEN ITV
```

Votre service de renseignement vous a indiqué que la clé utilisée pour chiffrer (le pas) était 17. Calculez la clé de déchiffremenet et utilisez votre fonction pour déchiffrer le message.

[12]
```
# Votre travail ici
```

Composition de méthodes

Le message suivant a été intercepté :

```
IIERO ZVLQK KJIYF AZDNB VEIQK KLMVE GZENF ACQGT
KDCNG GVSFN RRSHE ZLPFX BCTNQ GREYL TROET ALSCG
RZRYX VWENE XVUQX KVDYL OZEVN OCCVX OUCKL SZAEX
KFOWX OIORJ ZGCFL GKTAN BDNVF XLERX KGESG GCETQ
KERAT UGEQE MVONK YRERX GVOFG RLEEN UTDNB YLTBX
KDIRX RJARH ZFEGL XICNK XUVRL GAMGX GEBNG KHNFG
RZQHG IVMAB OOEYG JIUGX JIDRX YJSBL YWEYE XVSGL
KIERH VTANX YKRIX XDRFL XKTHX SZUGN DKHFB YELVT
KKGAK KIEDX REAQT XJISL KVPVI YHSRW YEIRU TTEEW
```

Vous savez qu'il s'agit d'une méthode de chiffrement assez complexe, puisqu'elle consiste à appliquer une transposition, du genre de celles que nous avons déjà vues, suivie d'un chiffrement de Vigénère.

Utilisez vous outils pour déchiffrer ce message. Vous avez un seul indice : il est question de la *mer*.

```
[13]    # Votre travail ici
```

Pour finir....

Vous avez déchiffré plusieurs messages. Pour être sûr qu'ils soient correctement identifiés, voici le début de chaque message :

- Le premier : `VUWVZ ZLKLL UMPUK`
- Le deuxième : `EBRTS GWWJR PJAED`
- Le troisième : `GUCMN YHYTJ IOLWY`
- Le quatrième : `MHMGU LRVAN THAUI`
- Le cinquième : `CATDO SRELO VAIQL`
- Le dernier : `IIERO ZVLQK KJIYF`

Tous ces messages sont extraits de textes (souvent des romans) dans lesquels la cryptographie joue souvent un rôle (parfois c'est juste parce que le texte nous plaît...). Et vous avez déchiffré tous ces messages. Pour les plus difficiles, vous avez utilisé le fait que le message était suffisament long par rapport à la clé (utilisée dans Vigénère).

Voici un dernier message intercepté. Il a été chiffré en utilisant l'algorithme de Vigénère.

`DYBVSCXDSERWIZETEXCSPYFFWAXFGSEAHQW`

Malheureusement, l'analyse fréquentielle ne donnera rien... Il a été chiffré en utilisant comme clé les noms de famille de 6 auteurs. Les 5 premiers sont les auteurs, dans l'ordre, des 5 premiers textes mentionnés plus haut (le dernier ne fait pas partie du jeu...). Le sixième est inconnu. Quel est le nom de ce sixième auteur ?

```
[14]    # Votre travail ICI
```