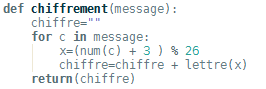
**Cryptage**

**Partie 1 (échauffement) : Cryptage de CESAR**

La fonction Python *chiffrement* ci-dessous prend en paramètre d'entrée une chaine de caractère *message* constituée uniquement de lettres majuscules sans aucun caractère autre (en particulier sans espaces) et lui applique la méthode de chiffrement dite de César.



*num()* est une fonction qui a une lettre majuscule renvoie sa position dans l'alphabet (par exemple A->0, B->1…). *lettre()* est une fonction qui à tout entier entre 0 et 25 renvoie la lettre correspondante dans l'alphabet en majuscule.

1- Quel traitement effectue cette fonction?

2- Modifier cette fonction en *cesar(message, decalage)* afin que le décalage soit un paramètre d'entrée.

3- Ecrire les fonctions *num()* et *lettre()*

Pour cela vous pourrez utiliser les fonctions :

* ord() qui prend en paramètre un caractère et renvoie son code ASCII/Unicode
* chr() qui prend en entrée un code ASCII/Unicode et renvoie le caractère correspondant.

exemples: ord('A')=65 chr(70)='F'

En Ascii, les majuscules correspondent à des nombres entre 65 (pour 'A') et 90 (pour 'Z'). Les minuscules correspondent à des nombres entre 97 (pour 'a') et 132 (pour 'z'), l'espace correspond à 32.

4- Ecrire une fonction *epure(texte)*, où texte est une chaine de caractères, qui renvoie cette chaîne de caractères en majuscule s'ils étaient en minuscules et épurée de tous les autres caractères.

5- Ecrire une fonction *dechiffrement(message,decalage)* qui permet de décoder un message crypté avec la méthode de César en connaissant le décalage.

6- Ecrire une fonction *cassechiffrement(message)* qui permet de décoder un message crypté avec la méthode de César même sans connaître le décalage.

Par exemple, décodons le message: "OHNLTOXSJNTGWFXFXEXIKHYWXFTMAXFTMBJNXLEXIENLVETLLXWNFHGWX"

*Petit aparté culturel:*

*Au départ, l'encodage des caractères différait d'un pays à l'autre (ou plutôt d'un alphabet à l'autre). Le premier standard fut l'ASCII (nécessitant 7 bits pour encoder tous les caractères nécessaires à un texte en alphabet anglosaxon). Les ordinateurs travaillant par octet, il restait donc 128 "codes" pour encoder les caractères manquant (La norme Latin-1 en Europe occidentale permettait d'encoder par exemple les lettres accentuées mais pas les caractères cyrilliques ou chinois…). Avec la mondialisation des échanges, il y a eu une volonté de créer un standard d'encodage permettant d'encoder tous les caractères du monde: l'Unicode. Deux enjeux se sont imposés. D'abord la nécessité d'une compatibilité avec toutes les données préexistantes. D'autre part, le besoin de ne pas trop augmenter la taille des données. En effet, si l'on veut réserver un nombre fixe de bits par caractères, il en faudrait 4 octets par caractère ce qui ferait exploser inutilement la taille des données pour un texte. La norme d'encodage Utf-8 est désormais la norme préférentielle pour la plupart des textes courants, parce que, d'une part, elle assure une parfaite compatibilité avec les textes encodés en « pur » ASCII (ce qui est le cas de nombreux codes sources de logiciels), ainsi qu'une compatibilité partielle avec les textes encodés à l'aide de ses variantes « étendues », telles que Latin-1 ; d'autre part, cette nouvelle norme est celle qui est la plus économe en ressources, tout au moins pour les textes écrits dans une langue occidentale. En effet, suivant cette norme, les caractères du jeu ASCII standard sont encodés sur un seul octet. Les autres seront encodés en général sur deux octets, parfois trois ou même quatre octets pour les caractères les plus rares.*

**Partie 2: Chiffrement par substitution**

La faiblesse de la méthode de cryptage de César réside dans son nombre de décalages possibles faible (25). Même pour un humain, il est facile de tester à la main toutes les possibilités, du fait de ce choix d'une permutation circulaire. Le nombre de possibilités devient plus intéressant si l'on choisit de remplacer une lettre par une autre lettre de l'alphabet sans respecter l'ordre des lettres.

par exemple, en utilisant la chaine alphabet="QZERTYUIOPASDFGHJKLMWXCVBN"

1- Ecrire une fonction *substitution(message)* qui renvoie une chaîne dans laquelle les lettres A du message ont été remplacées par Q, les B par Z, les C par E …

2- Ecrire une fonction *desubstitution(message)* pour décoder un message lorsqu'on connait l'alphabet de substitution.

On pourra utiliser la méthode pour chaine de caractère *index()* qui renvoie pour un caractère donné sa position dans la chaine.

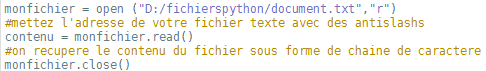
par exemple: alphabet.index('Z') = 1

Essayons de déchiffrer le message "FGFDQOLSTHKGYRTHIBLOJWTOSTLMLBDHQQWLLO"

Et quand on ne connait pas l'alphabet de substitution, quel est le nombre de possibilités à tester? Pour autant dès le moyen âge, on savait "casser" la méthode de chiffrement par substitution. En effet, dans chaque langue, certaines lettres ont une fréquence d'apparition caractéristique. Par exemple, en français, la lettre la plus fréquente est le 'E' (environ 16%). Si le caractère 'E' est remplacé toujours par la même lettre ('T' dans l'alphabet de substitution précédent), c'est cette lettre qui présentera l'occurrence la plus forte dans le message chiffré. On arrive ainsi à retrouver les lettres les plus fréquentes et déchiffrer le message. C'est la méthode dite d'analyse des fréquences, d'autant plus efficace que le texte est long.

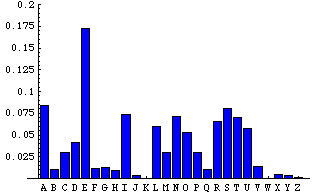
3- Ecrire une fonction *frequence(texte)* qui prend en entrée une chaine de caractère *texte* et renvoie une liste L contenant les fréquences d'apparition des différentes lettres de l'alphabet dans *texte*. L[0] correspondra à la fréquence d'apparition du caractère 'A', L[1] à celle de 'B'…

Vous pourrez la tester sur le fichier texte fourni.



Voici les fréquences observées dans un groupement de textes français

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| principales lettres | E | S | A | I | T | N | R | U |
| Fréquence observée |  |  |  |  |  |  |  |  |
| source Wikipedia | 14.7 | 7.9 | 7.6 | 7.5 | 7.2 | 7.1 | 6.6 | 6.3 |



( source wikipedia aussi!)

On peut se servir de cette méthode pour attaquer un code de César de manière plus systématique. Il suffit de rechercher quelle lettre est devenue la plus fréquente. Comme elle correspond à "E", on obtient le décalage utilisé.

4-Ecrire une fonction *decalage(texte)* qui renvoie le probable décalage utilisé si le texte a été codé par la méthode de César.

**Partie 3: Vigenere**

Inventé au 16ème siècle par un diplomate français en mission en Italie, il faut longtemps considéré comme inviolable. La méthode de chiffrement de Vigenère est une amélioration de la méthode de césar puisqu'on utilise plusieurs alphabets décalés pour chiffrer un message.

La table de Vigenère donne tous les alphabets décalés:



On choisit une clef de cryptage, par exemple : MATHS

Pour la première lettre du message on utilisera l'alphabet de substitution "M" (où A donne M), pour la seconde on utilisera l'alphabet de substitution "A" (où A donne A)…

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| message clair | R | D | V | V | E | N | D | R | E | D | I |
| clef de cryptage | M | A | T | H | S |  |  |  |  |  |  |
| message chiffré |  |  |  |  |  |  |  |  |  |  |  |

La lettre V est donc successivement codée par deux lettres différentes, ce qui rend le chiffrement plus résistant à l'analyse des fréquences.

Ecrire à partir des fonctions précédentes une fonction *vigenere(message,clef)*

Néanmoins sur un texte long, certains mots courants (les, une, des, pour, est …) finissent par "retomber" sur les mêmes morceaux de clefs. On observe donc des groupements de lettres récurrents.

Voici un message crypté par la méthode de Vigenère, cherchez des groupements de lettres récurrents:

LVQZWTFRGRYQHSWIPXKVQZWXDJGJWHMWNVGBWPUVTMHSWIPXKWGZKGWYTJRTARUUEGKPGLVLVIOVYKWWFVTBWVTVRXMJRFYKDIUYSFEIUDSKLINJHXKXKEILSYVIIISWWETHMVNVWXAKPVYKLIPVFKWYZJYKKSPJSFTVGKVHFIFRRLDIRRCLVIOFVWGVQJIMWRFVRMDIUFQUJIULRTFRGRYIGYTCILYSWMIKFITKSNKYPRRGWEWGSNJPGJXKGYXVVNFEPEITMTQLVEWWCDIGWVVFYLWXFRRLDIUKRUJIUCILDMGIENHEAJHXESTUSKGYUVXXFHGEXEWWQDFKWW

LVQZWTFRGRYQHSWIPXKVQZWXDJGJWHMWNVGBWPUVTMHSWIPXKWGZKGWYTJRTARUUEGKPGLVLVIOVYKWWFVTBWVTVRXMJRFYKDIUYSFEIUDSKLINJHXKXKEILSYVIIISWWETHMVNVWXAKPVYKLIPVFKWYZJYKKSPJSFTVGKVHFIFRRLDIRRCLVIOFVWGVQJIMWRFVRMDIUFQUJIULRTFRGRYIGYTCILYSWMIKFITKSNKYPRRGWEWGSNJPGJXKGYXVVNFEPEITMTQLVEWWCDIGWVVFYLWXFRRLDIUKRUJIUCILDMGIENHEAJHXESTUSKGYUVXXFHGEXEWWQDFKWW

Cela implique que la clef a été répétée un nombre entier de fois entre deux occurrences d'un même groupement de lettre. La taille de la clef est donc un multiple commun aux écarts entre deux mots récurrents.

Ici, l'écart entre les deux SWIP est de 30, l'écart entre les deux LVIO est de 108.

La clef comporte donc …. caractères.

Il suffit donc maintenant d'effectuer une analyse fréquentielle de toutes les lettres codées par la première lettre de la clé (puis de la seconde…).

On obtient

TNPSLUEPSEISDEEULMTSAAUITESBNLDOELRNOGNSERONUEEELRLPMONEE

RNORFSLOSUNLESRFEEETUSRGEUOREEERNEENUOEUALUAPSRTEEIAOUDSS

OEUOELSUERSEMDRPSSLITULNNXNEDPMODSSERURNUEVNOATDSSEYRSEO

IARISEERISDUEEEOHMSNRNEEESSTAAOSEOUALVTAPSENUMOATLRSDENM

SULSSCPLGNARUPNUOODEEPSUBUORNYRENMNUEEONOTREREUNNEADOTTB

NKRRBVGRANAFEVREZERFCBREEEZBFFQGGONCFEHAHEHNYAFFOFHRERYE

**Partie 4: le RSA**

***le principe***

Le cryptage RSA (Rivest Shamir Adelman) est une méthode de chiffrement assymétrique.

Bob souhaite faire passer à Alice un message "secret", un nombre qu'on notera M.

Alice se crée une clef publique de chiffrement qu'elle diffuse à tout le monde. Elle est composée de deux nombres : n (généralement très grand, en tout cas très supérieur à M) et e. Le couple (n;e) est appelé clef publique de chiffrement d'Alice. Cette clef permet de chiffrer le message mais pas de le déchiffrer.

Bob chiffre son message en calculant: C = Me modulo n puis l'envoie à Alice.

Alice reçoit le message, qu'elle déchiffre en utilisant sa clef privée de déchiffrement (n;d).

Pour cela elle calcule: M = Cd modulo n

**Comment sont déterminées les clefs publique (n;e) et privée (n;d)?**

Pour "choisir" le nombre n, Alice a pris deux nombres premiers (très grands) p et q qu'elle a multiplié: n = pq.

e et d ne sont pas choisis au hasard mais tels que e x d = 1 modulo phi(n)

phi(n) désigne la fonction d'Euler, donnant le nombre d'entiers inférieurs à n et premiers avec n. Le calcul de phi(n) est d'une complexité élevée (voir dm question 1) sauf dans le cas particulier où n est le produit de deux nombres premiers n = pq. On peut alors justifier que phi(n) = (p-1)(q-1).

On peut démontrer que (Me)d = M modulo n

La sécurité du système tient dans la difficulté à trouver d dans un temps raisonnable. Etant défini comme l'inverse de e modulo phi(n), son calcul nécessite de connaître phi(n).

***exemple d'utilisation***

Alice choisit p=7 et q=19 (premiers tous les deux) donc n= 7 x 19 = ………..

Elle en déduit que phi(n)= (p-1) x (q-1) = ……………………………………. = ………………..

Elle choisit e premier avec 108, par exemple 5 (il serait absurde de prendre 1 dont l'inverse est un peu trop simple à trouver puisque c'est 1). Reste à trouver d, inverse de 5 modulo 108. Pour cela on peut tester les 107 valeurs possibles (de 1 à 107).

C'est ce que fait l’algorithme A suivant de manière non optimale

|  |  |
| --- | --- |
| Variables | e,i,d entiers |
| 0 | e ←5 |
| 1 | i ←1 |
| 2 | Tant que ( i < 108 ) : |
| 3 | si ((i\*e)%108 =1) : |
| 4 | d ← i |
| 5 | finsi |
| 6 | i ← i+1 |
| 7 | Fin de tant que |
| 8 | **Sortie :** d |

Comment l'optimiser simplement?

On obtient que d=65 (car 65 x 5 = 325 = 3 x108 + 1).

La clef publique de chiffrement est donc (n=133; e=5) et la clef privée est donc (n=133; d=65).

Appliquons le au message M=6.

Bob chiffre: C = Me modulo n = 65 modulo 133 =…………..

Alice déchiffre : M = Cd modulo n = …..65 modulo 133 = ……..