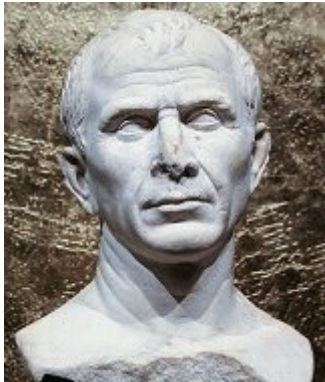


TD : Code César

1. Chiffrement de César



En **cryptographie**, le chiffrement par décalage, aussi connu comme le chiffre de César ou le code de César, est une méthode de **chiffrement** très simple utilisée par Jules César dans ses correspondances secrètes (ce qui explique le nom « chiffre de César »).

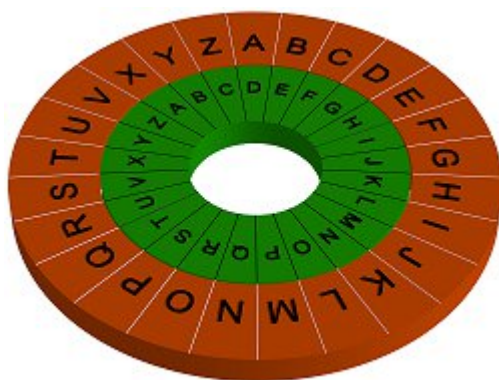
Le texte chiffré s'obtient en remplaçant chaque lettre du texte clair original par une lettre à distance fixe, toujours du même côté, dans l'ordre de l'alphabet. Pour les dernières lettres (dans le cas d'un décalage à droite), on reprend au début. Par exemple avec un décalage de 3 vers la droite, A est remplacé par D, B devient E, et ainsi jusqu'à W qui devient Z, puis X devient A etc. Il s'agit d'une permutation circulaire de l'alphabet. La longueur du décalage, 3 dans l'exemple évoqué, constitue **la clé du chiffrement**

qu'il suffit de transmettre au destinataire — s'il sait déjà qu'il s'agit d'un chiffrement de César — pour que celui-ci puisse déchiffrer le message. Dans le cas de l'alphabet latin, le chiffre de César n'a que 26 clés possibles.



Source : https://fr.wikipedia.org/wiki/Chiffrement_par_d%C3%A9calage

2. Activité : des chiffres et des lettres



Nous adopterons la convention suivante, en vert c'est la partie du message à laquelle tout le monde a accès (ou qui pourrait être intercepté), c'est donc le message crypté. Alors qu'en rouge c'est la partie du message confidentiel, c'est le message en clair.

Pour prendre en compte aussi les dernières lettres de l'alphabet, il est plus judicieux de représenter l'alphabet sur un anneau. Ce décalage est un décalage circulaire sur les lettres de l'alphabet. Pour déchiffrer le message de César, il suffit de décaler les lettres dans l'autre sens, D se déchiffre en A, E en B,...

Il est plus facile de manipuler des nombres que des lettres, aussi nous passons à une formulation mathématique. Nous associons à chacune des 26 lettres de A à Z un nombre de 0 à 25.

Source: Arnaud Bodin & François Recher, université de Lille

Le code ASCII de la lettre 'A' a pour valeur 65, celui de la lettre 'Z' 90. Quel que soit le décalage appliqué, il faut rester dans l'intervalle [65 ; 90].

Méthode :

- I. $([65 ; 90] - 65) \rightarrow [0 ; 25]$ on soustrait 65 pour être dans un intervalle $[0 ; 25]$
- II. $([0 ; 25] + \text{clef}) \rightarrow [\text{clef} ; 25 + \text{clef}]$ on décale selon la clef
- III. $[\text{clef} ; 25 + \text{clef}] \bmod 26 \rightarrow [0 ; 25]$ le modulo permet de rester dans l'intervalle $[0 ; 25]$
- IV. $([0 ; 25] + 65) \rightarrow [65 ; 90]$ on ajoute 65 pour revenir dans un intervalle $['A' ; 'Z']$

On appelle **modulo** d'un nombre x par N , l'opérateur qui renvoie le reste de la division entière de x par N . Exemple : $11 \bmod 5 = 1$ (en Python, le modulo est notée `%`).

1. Aller sur EduPython et créer le TD-Caesar.
2. Écrire une fonction qui utilise la méthode de César pour crypter le message : "ATTAQUEZ DE-MAIN".

Le prototype de la fonction est : **caesar(text : str, key : int) -> str**

text représente le texte à chiffrer, key représente la clef de chiffrement. La fonction doit renvoyer le texte chiffré.

NB : les caractères (lettres ou signes de ponctuations) qui ne figurent pas dans l'alphabet $[A..Z]$ restent inchangés.

3. Créer une variable « message_code » qui prend pour valeur le message "ATTAQUEZ DE-MAIN" codé par votre fonction. Afficher la valeur de la variable message_code et vérifiez que vous obtenez : "DWWDTXHC GHPDLQ"
4. Tester la fonction « caesar(message_code,-key) » avec pour paramètres le message codé précédemment et l'opposé de la clef utilisée pour coder le message. Que remarquez-vous ?
5. Créer un docstring pour la fonction du programme caesar.py.

Une fonction est basée sur le modèle de la **boite noire** ; les sorties n'étant fonction que des entrées. Cette boite noire doit être à la fois **cohérente** (des entrées supposées fournir un résultat juste doivent donner un résultat juste, des entrées supposées donner un résultat faux doivent donner un résultat faux) et **stable** (le programme doit terminer, ne pas « crasher »). La vérification de ces deux critères se fait à travers des **tests unitaires**.

6. Tests de cohérence, écrire une fonction tests_unitaires() dans laquelle il y aura une série de tests unitaires qui vérifie les conditions suivantes :
 - pour key = 0 et text = 'A' : sortie = 'A'
 - pour key = 1 et text = 'Z' : sortie = 'A'
 - pour key = 2 et text = 'Y' : sortie = 'A'
 - pour key = 26 et text = 'A' : sortie = 'A'
 - pour key = 26*2 et text = 'A' : sortie = 'A'
 - pour key = -26 et text = 'A' : sortie = 'A'
 - pour key = 1 et text = " " : sortie = " "

NB : utiliser l'instruction **assert**.

8. Appeler la fonction « tests_unitaires() » dans le programme principal.

9. Tests de stabilité

En génie logiciel, le **Secure by design**, implique que le programme doit être conçu dès le départ pour être sécurisé. Les pratiques malveillantes sont considérées comme allant de soi et on prend soin de minimiser l'impact en anticipant les vulnérabilités de sécurité.

Soure : https://en.wikipedia.org/wiki/Secure_by_design

Pour éviter que le programme « crash » sur une erreur parce que le type de la valeur d'entrée (TypeError), une façon de gérer cette erreur est d'intercepter le message d'erreur et de le traiter correctement.

10. Écrire une deuxième série de tests unitaires dans la fonction « tests_unitaires() » qui vérifie les conditions suivantes :

- pour key = 'a' et text = 'Z' : sortie = 'Z'
- pour key = 1 et text = 123456 : sortie = 12345

3. Activité : décryptage

Décrypter consiste à retrouver le texte original à partir d'un message chiffré sans posséder la clé de (dé)chiffrement.

Le challenge consiste à décrypter le message ci-dessous.

AV WRZJ JFLMVEK TV IVMV VKIREXV VK GVEVKIREK
U'LEV WVDDV ZETFEELV, VK HLV A'RZDV, VK HLZ D'RZDV,
VK HLZ E'VJK, TYRHLV WFZJ, EZ KFLK R WRZK CR DVDV
EZ KFLK R WRZK LEV RLKIV, VK D'RZDV VK DV TFDGIVEU.

Le chiffrement de César a été utilisé mais avec une clef différente de 3.

Écrire un programme afin de donner le nom de l'auteur du message crypté.

NB : La clef de chiffrement étant inconnue, il faut faire une boucle qui les teste toutes (technique de l'attaque par force brute).