**Documentation Complète pour le Langage TWORK**

**1. Introduction**

**1.1. Présentation**

TWORK est un langage de codage personnalisé conçu pour transformer les lettres et espaces en une structure unique et cryptée. Ce langage innovant a été pensé pour offrir une nouvelle perspective sur la manipulation des données textuelles. Il est particulièrement adapté aux applications artistiques, cryptographiques, ou simplement pour expérimenter de nouvelles formes de représentation textuelle.

**2. Fonctionnalités Clés**

1. **Transformation des lettres :**  
   Chaque lettre de l'alphabet est transformée en une combinaison composée de :
   * Une nouvelle lettre basée sur un décalage dynamique (par défaut, 7 positions).
   * Une valeur décimale obtenue par la division par 3 de la position de la lettre dans l'alphabet.
2. **Traitement des espaces et caractères spéciaux :**
   * Les espaces sont représentés par le symbole **0.5**.
   * Les caractères spéciaux, chiffres ou autres éléments restent inchangés.
3. **Clé secrète :**  
   Une clé secrète introduit un facteur dynamique, permettant de personnaliser le décalage pour chaque caractère.
4. **Cryptage supplémentaire :**  
   Le texte codé en TWORK peut être crypté à l’aide d’un algorithme de cryptage moderne comme **Fernet**.
5. **Décryptage :**  
   TWORK intègre un processus inverse pour retourner au texte original, à condition de disposer de la clé secrète et des paramètres exacts utilisés pour le codage.
6. **Sauvegarde et gestion des fichiers :**  
   Le système permet de sauvegarder les différents états du texte (codé, crypté, décodé ou décrypté).

**3. Structure des Données**

**3.1. Algorithme de Transformation**

Pour une lettre donnée (L) :

* **Étape 1 : Décalage**  
  Identifiez la position de (L) dans l’alphabet (1 pour A, 26 pour Z). Ajoutez le décalage (par défaut, 7 positions). Si le décalage dépasse (Z), recommencez à partir de (A).
* **Étape 2 : Calcul de la valeur numérique**  
  Prenez la position de la lettre résultante et divisez-la par 3. Conservez deux décimales.
* **Exemple de conversion :**
  + (L = A)  
    Position initiale = 1.  
    Décalage de 7 → Nouvelle lettre = (H).  
    Position de (H = 8), donc (8 \div 3 = 2,66).  
    Résultat = **H2,66**.

**3.2. Espaces et Limites**

* Les espaces sont codés sous la forme **0.5**.
* Les limites de l’alphabet sont circulaires (après (Z), le comptage reprend à (A)).

**4. Sécurité**

**4.1. Clé Secrète**

* Une clé secrète est utilisée pour modifier dynamiquement le décalage pour chaque caractère.
* Par exemple, si la clé est **"CODE"**, les décalages seront respectivement de 3, 15, 4 et 5 pour chaque lettre, en boucle.

**4.2. Cryptage Fernet**

* Le cryptage Fernet ajoute une couche supplémentaire de sécurité en transformant le texte TWORK en une chaîne cryptée non lisible.
* Il garantit la confidentialité des données lors de leur échange.

**4.3. Hachage**

* Chaque message codé peut être accompagné d’une signature hachée (SHA-256), permettant de vérifier l’intégrité du texte.

**5. Interface Utilisateur**

**5.1. Interface Graphique**

L’interface est conçue avec **Tkinter** et comprend les éléments suivants :

* **Zone d’entrée :** Permet de saisir le texte original.
* **Zone pour la clé secrète :** Permet de personnaliser le décalage.
* **Boutons d’action :**
  + Convertir et Crypter.
  + Décrypter et Décoder.
  + Sauvegarder les résultats.
* **Zones d’affichage :**
  + Texte codé en TWORK.
  + Texte crypté.
  + Texte original après décryptage.
  + Signature Hachée.

**5.2. Sauvegarde**

Les options de sauvegarde permettent d’exporter :

* Le texte codé en langage TWORK.
* Le texte crypté.
* Le texte original décrypté.

**6. Cas d’Utilisation**

1. **Applications éducatives** :
   * Enseignement des concepts de décalage, cryptage et transformation des données.
2. **Projets artistiques** :
   * Création de textes ou d’œuvres utilisant une représentation unique.
3. **Sécurisation des messages** :
   * Envoi de données textuelles confidentielles grâce au cryptage intégré.

**7. Développement et Maintenance**

**7.1. Technologies Utilisées**

* **Langage :** Python.
* **Bibliothèques :**
  + Tkinter : Interface utilisateur.
  + cryptography : Cryptage et décryptage.
  + hashlib : Génération de signatures hachées.

**7.2. Améliorations Futures**

* Support de nouveaux caractères ou alphabets.
* Intégration avec des bases de données pour enregistrer automatiquement les textes.
* Ajout de thèmes personnalisables pour l’interface.

**8. Manuel d'Utilisation**

**8.1. Installation**

1. Installez Python sur votre système.
2. Installez la bibliothèque cryptography :
3. pip install cryptography

**8.2. Lancement**

1. Téléchargez le fichier twork.py.
2. Exécutez le fichier avec la commande :
3. python twork.py

**8.3. Utilisation**

* **Codage :** Saisissez un texte, entrez une clé secrète, puis cliquez sur "Convertir et Crypter".
* **Décryptage :** Copiez le texte crypté, entrez la même clé secrète, et cliquez sur "Décrypter et Décoder".
* **Sauvegarde :** Sélectionnez une option dans le menu déroulant et cliquez sur "Sauvegarder".

**9. Exemple Concret**

**9.1. Texte Original**

Bonjour le monde

**9.2. Texte Codé en TWORK**

i2,66 p2,33 s2,66 0.5 s2,33 o1,33 p3,0 0.5 n2,0 v2,33 a0,33

**9.3. Texte Crypté**

gAAAAABkxdG8K2lQHzEvOpV8Z7rxKm1......

**9.4. Texte Décrypté**

Bonjour le monde

**10. Auteurs et Licence**

* **Auteur :** Vladimir Maxime
* **Contact :** [Vcoronaldo@gmail.com]
* **Licence :** Open Source, sous la licence MIT.