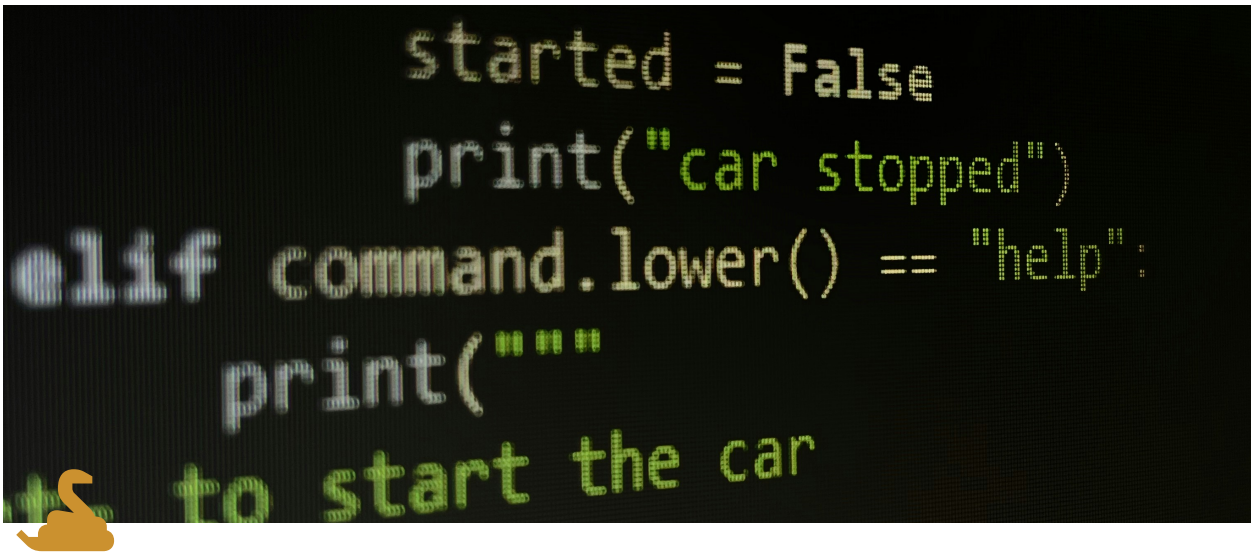


Python Projects

Ici je vais présenter et détailler quelques script python que j'ai pu rédiger.

Donut Animation :

Ce script Python utilise la bibliothèque `Pygame` pour créer une animation graphique représentant un "donut" tournant, utilisant des caractères ASCII pour le rendu visuel. Voici une explication détaillée de ce que fait le script :

Le script commence par initialiser `Pygame` et configurer la fenêtre d'affichage. Ensuite, il définit divers paramètres pour contrôler la taille de la fenêtre, la grille de caractères, la séparation entre les lettres, les couleurs, et les angles de rotation du donut. L'animation est affichée dans une boucle principale, et l'utilisateur peut la mettre en pause ou fermer la fenêtre avec certaines touches du clavier.

Explications des Parties Principales du Script

1. **Initialisation et Configuration :**
   - `pygame.init()` : Initialise tous les modules `Pygame` nécessaires.

- Les variables `WIDTH` et `HEIGHT` définissent la taille de la fenêtre, et `screen` configure cette fenêtre.

2. **Variables Importantes :**
   - `x_separator` et `y_separator` contrôlent l'espacement horizontal et vertical entre les caractères du rendu.
   - `rows` et `columns` calculent le nombre de lignes et de colonnes pour la grille de caractères, basée sur la taille de la fenêtre.
   - `chars` est une chaîne de caractères utilisée pour représenter visuellement le donut.
3. **Couleurs et Conversion :**
   - `hue` : Une valeur de teinte qui change graduellement pour créer une animation colorée.
   - `hsv2rgb()` : Une fonction qui convertit des valeurs de couleur HSV (teinte, saturation, luminosité) en RGB pour les utiliser avec `Pygame`.
4. **Affichage des Lettres :**
   - `text_display(letter, x_start, y_start)` : Cette fonction affiche une lettre sur l'écran aux coordonnées spécifiées avec la couleur définie par la teinte `hue`.
5. **Animation :**
   - Une série de calculs mathématiques, utilisant les fonctions sinus et cosinus, génère les coordonnées x et y pour chaque point du donut en rotation. Les caractères sont placés sur la grille en fonction de ces calculs pour créer une illusion de rotation 3D.
   - Les angles `A` et `B` sont progressivement augmentés pour faire tourner le donut.
6. **Contrôle de l'Animation :**
   - La boucle principale (`while run`) gère les événements utilisateur. L'animation peut être mise en pause avec la barre d'espace ou arrêtée en fermant la fenêtre ou en appuyant sur Échap.

- Lorsqu'elle n'est pas en pause, la boucle met à jour l'affichage avec le donut animé.

Conclusion

Le script génère une animation d'un donut tournant, rendu en caractères ASCII avec des couleurs changeantes pour un effet visuel dynamique. C'est une démonstration amusante de la manipulation des graphiques et des animations avec Pygame, en utilisant des mathématiques pour simuler un effet 3D.

Menu Chiffrement/Déchiffrement et Brut force :

Ce script Python est une application interactive qui propose plusieurs fonctionnalités, notamment le chiffrement et le déchiffrement de texte avec le chiffrement de César, ainsi que des attaques de force brute pour trouver des codes PIN et des mots dans un dictionnaire. Voici une explication détaillée de chaque section du script :

Le script est structuré autour d'un menu principal qui permet à l'utilisateur de choisir entre différentes options : chiffrer/déchiffrer un message, réaliser une attaque de force brute pour un code PIN ou un mot de passe, ou quitter l'application. Deux sous-menus gèrent les options de chiffrement et de force brute.

Explication des Fonctions

1. **Chiffrement de César :**

- `chiffrement_cesar(texte, decalage)` :
  - Cette fonction chiffre un texte en utilisant le chiffrement de César, qui consiste à décaler chaque caractère d'un nombre spécifié de positions dans l'alphabet.
  - `alpha` est une chaîne qui contient les lettres de l'alphabet en minuscules, majuscules, et quelques caractères spéciaux (`' .'`), pour gérer plus de caractères que le chiffrement traditionnel.
  - Pour chaque caractère dans `texte`, s'il est présent dans `alpha`, il est décalé de `decalage` positions. Sinon, le caractère est ajouté au texte chiffré sans modification.
- `dechiffrement_cesar(texte, decalage)` :
  - Cette fonction utilise `chiffrement_cesar` avec un décalage négatif pour déchiffrer un message.

---

1. **Attaques de Force Brute :**
   - `brute_force1(pin)` :
     - Cette fonction réalise une attaque de force brute sur un code PIN à 4 chiffres. Elle essaie toutes les combinaisons de `0000` à `9999`, en les affichant et en vérifiant si elles correspondent au PIN donné.
     - Si le code PIN est trouvé, il est retourné. Sinon, la fonction retourne `None`.
   - `brute_force2(mot_a_trouver, nom_fichier)` :
     - Cette fonction réalise une attaque de force brute à l'aide d'un fichier de dictionnaire.
     - Elle essaie chaque mot dans le fichier pour voir s'il correspond au mot recherché.
     - Si le fichier spécifié n'existe pas, une exception `FileNotFoundError` est capturée, et un message d'erreur est affiché.

---

1. **Menu Principal et Sous-Menus :**
   - `menu()` :

    - Affiche le menu principal et gère les choix de l'utilisateur (chiffrement, force brute, ou quitter).
    - Redirige vers `sous_menu1()` pour le chiffrement/déchiffrement ou `sousmenu2()` pour les attaques de force brute.
- `sous_menu1()` :
    - Propose de chiffrer ou de déchiffrer un texte.
    - Vérifie que le texte ne contient que des caractères alphabétiques et demande un décalage pour le chiffrement.
    - Affiche le texte chiffré ou déchiffré.
- `sousmenu2()` :
    - Propose deux méthodes d'attaque de force brute : exhaustive (toutes les combinaisons possibles pour un code PIN) ou par dictionnaire (recherche de mots dans un fichier).
    - Vérifie que le code PIN est un nombre à 4 chiffres et effectue l'attaque de force brute appropriée.

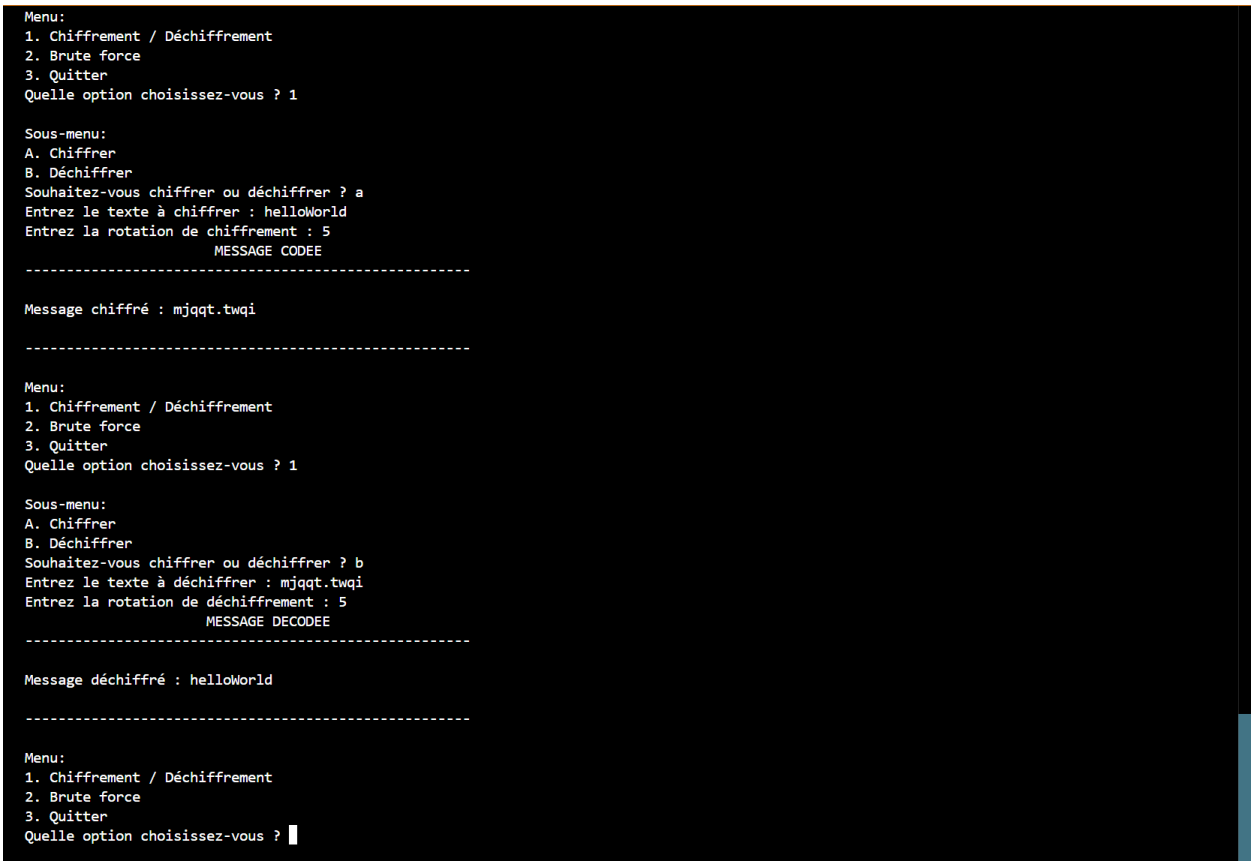
Points Importants

- **Chiffrement de César** : Une méthode de chiffrement par substitution simple où chaque lettre du texte est remplacée par une lettre située plus loin dans l'alphabet. Le décalage peut être positif (pour chiffrer) ou négatif (pour déchiffrer).
- **Force Brute** : Une méthode de recherche qui essaie toutes les combinaisons possibles ou compare les mots d'un fichier de dictionnaire pour trouver une correspondance.
- **Gestion des Erreurs** : La fonction `brute_force2()` gère les erreurs lorsqu'un fichier n'est pas trouvé, ce qui empêche le script de planter.

Conclusion

Ce script est un outil simple mais complet pour chiffrer et déchiffrer des messages ou tenter de casser des codes par force brute. Il introduit des concepts

de cryptographie basique et des méthodes de sécurité informatique. L'utilisation des menus rend le programme interactif et facile à utiliser.



Chiffrement/Déchiffrement de fichiers :

Ce script Python utilise la bibliothèque `cryptography` pour chiffrer et déchiffrer un fichier avec un algorithme de chiffrement symétrique, `Fernet`.

Le script permet de sécuriser des fichiers en les chiffrant avec une clé générée automatiquement, puis de les déchiffrer en utilisant cette même clé. Le chiffrement et le déchiffrement sont réalisés grâce à la classe `Fernet` de la bibliothèque `cryptography`.

Explication des Fonctions

1. `generer_cle()`
    - **Description** : Génère une clé de chiffrement symétrique de manière aléatoire.

- **Retourne** : Une clé binaire qui sera utilisée pour chiffrer et déchiffrer les données.
- **Utilisation** : Cette clé est essentielle pour les opérations de sécurité, et elle doit être protégée pour éviter que les données ne soient compromises.

2. `chiffrer_fichier(nom_fichier, cle)`
   - **Description** : Chiffre un fichier en utilisant la clé fournie.
   - **Étapes** :
     - Le fichier spécifié par `nom_fichier` est lu en mode binaire.
     - Un objet `Fernet` est créé avec la clé de chiffrement.
     - Les données sont chiffrées avec la méthode `encrypt()`.
     - Les données chiffrées sont écrites dans un nouveau fichier nommé avec l'extension `.encrypted`.
   - **Affichage** : Un message de confirmation indique que le fichier a été chiffré avec succès.
3. `dechiffrer_fichier(nom_fichier_chiffre, cle)`
   - **Description** : Déchiffre un fichier en utilisant la clé fournie.
   - **Étapes** :
     - Le fichier chiffré est lu en mode binaire.
     - Un objet `Fernet` est créé avec la clé de chiffrement.
     - Les données sont déchiffrées avec la méthode `decrypt()`.
     - Les données déchiffrées sont écrites dans un fichier nommé `fichier_dechiffre.txt`.
   - **Affichage** : Un message de confirmation indique que le fichier a été déchiffré avec succès.

Partie Principale du Script

- Le script exécute les opérations de chiffrement et de déchiffrement de manière automatique lorsqu'il est exécuté directement (`if __name__ ==`

`"__main__":`).

- **Étapes Suivies :**
  1. Une clé est générée avec `generer_cle()`.
  2. Un fichier nommé `mon_fichier.txt` est chiffré en utilisant cette clé, et le fichier chiffré est nommé `mon_fichier.txt.encrypted`.
  3. Le fichier chiffré est ensuite déchiffré avec la même clé, et les données déchiffrées sont écrites dans `fichier_dechiffre.txt`.

Points Importants

- **Algorithme de Chiffrement :** `Fernet` est un algorithme de chiffrement symétrique qui garantit la confidentialité des données en utilisant une clé unique pour chiffrer et déchiffrer.
- **Sécurité de la Clé :** La clé générée doit être stockée de manière sécurisée, car quiconque possède cette clé peut déchiffrer les données.
- **Format des Données :** Les fichiers sont manipulés en mode binaire (`'rb'` et `'wb'`) pour préserver leur intégrité, ce qui est crucial pour le chiffrement.

Conclusion

Ce script est un exemple simple et efficace de la manière de protéger des fichiers avec un chiffrement fort en Python. Il montre l'importance de la sécurité des données et de la gestion des clés dans le chiffrement symétrique.

Code Barre conversion :

Ce script Python utilise la bibliothèque `python-barcode` pour générer des codes-barres de type EAN-13. Il permet à l'utilisateur d'entrer un numéro de 13 chiffres, puis crée et enregistre un code-barres correspondant.

Le script est une application interactive qui demande à l'utilisateur de saisir un code de 13 chiffres, puis génère un code-barres de type EAN-13 à partir de ce code et l'enregistre sous forme d'image. L'utilisateur peut entrer plusieurs codes successifs ou taper `'q'` pour quitter le programme.

Explication du Fonctionnement

1. **Importation de la Bibliothèque :**
   - `from barcode import EAN13`: Importation de la classe `EAN13` de la bibliothèque `python-barcode`, qui est utilisée pour créer des codes-barres EAN-13.
2. **Boucle Principale :**
   - Le script utilise une boucle `while True` pour continuer à demander un code de 13 chiffres jusqu'à ce que l'utilisateur décide de quitter en entrant `'q'`.
   - **Saisie de l'Utilisateur :**
     - L'utilisateur est invité à entrer un code de 13 chiffres ou à taper `'q'` pour quitter.
     - Si `'q'` est entré (en minuscules ou majuscules), la boucle se termine, et le script affiche "Programme terminé".
   - **Vérification du Format :**
     - Si l'entrée n'est pas un code de 13 chiffres ou contient des caractères non numériques, le script affiche un message d'erreur et demande de réessayer.
     - La saisie est validée pour s'assurer qu'elle contient exactement 13 chiffres.
3. **Génération et Enregistrement du Code-Barres :**
   - Une fois que l'utilisateur a entré un code valide, un code-barres EAN-13 est créé avec `EAN13(number)`.
   - Le code-barres est ensuite enregistré en tant qu'image avec le nom `new_code_barre`. Le fichier est sauvegardé dans le répertoire courant, et l'extension du fichier est déterminée automatiquement par `python-barcode` (généralement `.svg` ou `.png`).
   - Le script affiche un message de confirmation indiquant que le code-barres a été généré et sauvegardé.

Points Importants

- **EAN-13** : Le code-barres EAN-13 est un standard utilisé couramment pour identifier des produits, particulièrement dans le commerce de détail. Il est composé de 13 chiffres, et sa validité est essentielle pour qu'il soit lisible par les scanners de codes-barres.
- **Vérification des Entrées :** Le script s'assure que l'entrée de l'utilisateur est conforme (13 chiffres exactement) pour éviter des erreurs lors de la génération du code-barres.
- **Gestion de la Sortie :** Le fichier du code-barres est nommé de manière fixe (`new_code_barre`), ce qui signifie qu'il écrasera tout fichier précédent portant le même nom. Si l'utilisateur souhaite générer plusieurs codes-barres sans écraser les précédents, une modification du script serait nécessaire pour nommer les fichiers de manière unique.

Conclusion

Ce script est un exemple simple d'utilisation de la bibliothèque `python-barcode` pour générer et enregistrer des codes-barres EAN-13. Il est interactif et accessible, tout en mettant l'accent sur la validation des entrées de l'utilisateur pour garantir que les codes-barres générés soient corrects.