Iteration 4 - Lites et dictionnaires

Etapes

1 : Etude documentaire

> *Question 1 :* Qu'est ce que l'Electronic Frontier Foundation en quelques mots ?

- L'EFF (Electronic Frontier Foundation) est une organisation qui défend la liberté des citoyens dans le monde digital. Elle a été fondée en 1990.

> *Question 2 :* Expliquez en quelques lignes le principe de la génération de passphrases selon le document

- Ils choisissent au hasard 5 chiffres qui désigne un mot aléatoire. Ils répètent cette étape 4 fois de plus pour un total de 6 mots. Ils assemblent ces mots pour créer une passphrase, qui peut être retenu en prenant chacun de ses mots et en faisant une phrase mémotechnique pour s'en souvenir.

> *Question 3 :* Dans quels cas l'EFF recommande d'utiliser les passphrases plutôt que des mots de passes aléatoires classiques ?

- L'EFF recommande d'utiliser les passphrases pour des informations encryptés comme des disques dans un pc ou un téléphone. Ils peuvent aussi être utilisées pour des clé encryptés (clé PGP, SSH).

2 : Créer la fonction `generer_passphrase`

Code de `generer_passphrase()` :

```
    """
    Docstring de la fonction generer_passphrase

    On renvoie 'passphrase-generee-par-defaut'
    """

    def generer_passphrase():

        return "passphrase-generee-par-defaut"
```

Code de `main.py` :

```python
import mdp # On importe le module mdp

choix_utilisateurice=-1 # On affecte une valeur non utilisé à la variable pour pouvoir lancer la

"""
Docstring de la fonction afficher_menu().

Ici on définit la fonction afficher_menu() sans paramètre.

Elle permet, quand elle est appellée, d'afficher le menu du générateur de mots de passe.
"""

def afficher_menu(): # On définit une fonction 'afficher_menu' sans paramètre.
    print("================================================")
    print("HAUTOT Nolan | CIEL - Générateur de mots de passe")
    print("================================================")
    print("") # Cette ligne permet de créer un espace entre deux lignes, ce qui rend le menu plu
    print("Quel type de mot de passe souhaitez-vous créer ?")
    print("")
    print("1 - Mot de passe avec configuration par défaut.")
    print("2 - Mot de passe avec configuration personnalisée.")
    print("3 - Phrase de passe avec configuration par défaut.")
    print("4 - Phrase de passe avec configuration personnalisée.")
    print("")
    print("0 - Quitter le menu du générateur.")
    print("")

"""
Docstring de la fonction choix_generateur(choix_utilisateur).

On définit une fonction choix_générateur() avec comme paramètre choix_utilisateur.

On détecte le choix de celui-ci, et on renvoie plusieurs cas avec plusieurs messages personnalis
on génère un mot de passe complètement aléatoire.
"""

def choix_generateur(choix_utilisateurice): # On définit une fonction 'choix_generateur' avec co
    match choix_utilisateurice:
        case 1:
            print(mdp.generer_mdp()) # On appelle la fonction 'generer_mdp' et on l'affiche.
        case 2:
            password_length = int(input("Veuillez entrer une longueur de mot de passe : ")) # On
            print(mdp.generer_mdp(password_length)) # On renvoie son mot de passe en fonction de
```

```
        case 3:
            print(mdp.generer_passphrase())
        case 4:
            print("Phrase de passe personnalisée : V0ic1unephr4s3mdpPERSO#!$")
        case 0:
            print("Vous quittez le programme. Au revoir :)")
        case _:
            print(f"Erreur : l'option {choix_utilisateurice} n'existe pas ! Veuillez entrer une

    if choix_utilisateurice!=0:
        input("Appuyez sur 'Entrée' pour reboucler") # Ici dès que l'utilisateur appuie sur 'Ent

while choix_utilisateurice!=0: # Tant que 'choix_utilisateurice' est différent de 0.

    afficher_menu() # On appelle la fonction.

    choix_utilisateurice=int(input("Saisissez votre choix : ")) # On demande le choix, qui sera

    choix_generateur(choix_utilisateurice)
```

3 : Générer une phrase de passe à partir d'une liste fixe

Code de `generer_passphrase()` :

```
"""
Docstring de la fonction generer_passphrase

On créer une liste 'words' avec plusieurs chaînes de caractères.

On créer une variable 'passphrase' à laquelle on affecte la liste words, séparé par des tirets.

On renvoie passphrase
"""

def generer_passphrase():
    words = ["passphrase", "generee", "par", "defaut"] # On affecte une liste de chaînes de cara
    passphrase = "-".join(words) # On ajoute la liste words, séparée par des tirets à la variabl
    return passphrase # On renvoie 'passphrase'
```

> *Question 1 :* Est-ce qu'une liste contient les éléments dans un ordre fixe ?

- Oui la liste contient les éléments dans un ordre fixe (0, 1, 2, 3). On dit que c'est une structure de données ordonnée.

4 : Générer une phrase de passe aléatoire

Code de `generer_passphrase()` :

```
"""
Docstring de la fonction generer_passphrase

On créer une liste 'words' qui prend la liste entière de 'eff_words.py'

On créer une liste vide 'passphrase_words'

On répète ensuite 6 fois :

    On prend notre liste et on lui ajoute un mot aléatoire de la liste 'words'

On sépare chaque chaîne de caractère de la liste 'passphrase_words' par un tiret

On renvoie passphrase_words
"""

def generer_passphrase():
    words = get_word_list()
    passphrase_words = []
    for _ in range(6):
        passphrase_words.append(secrets.choice(words))
    passphrase_words = "-".join(passphrase_words)
    return passphrase_words
```

5 : Rendre paramétrable la longueur de la passphrase

Code de `generer_passphrase()` :

```
"""
Docstring de la fonction generer_passphrase

Cette fonction va nous permettre de générer une phrase de passe avec un nombre de mots aléatoire

On l'initialise avec un paramètre 'nb_words' (qui aura pour valeur par défaut 6)

On créer une liste 'words' qui prend la liste entière de 'eff_words.py'

On créer une liste vide 'passphrase_words'

On répète ensuite 6 fois :

    On prend notre liste et on lui ajoute un mot aléatoire de la liste 'words'

On sépare chaque chaîne de caractère de la liste 'passphrase_words' par un tiret

On renvoie passphrase_words
"""

def generer_passphrase(nb_words=6):
    words = get_word_list()
    passphrase_words = []
    for _ in range(nb_words):
        passphrase_words.append(secrets.choice(words))
    passphrase_words = "-".join(passphrase_words)
    return passphrase_words
```

Code de `main.py` avec cas n°4 modifié :

```python
import mdp # On importe le module mdp

choix_utilisateurice=-1 # On affecte une valeur non utilisé à la variable pour pouvoir lancer la

"""
Docstring de la fonction afficher_menu().

Ici on définit la fonction afficher_menu() sans paramètre.

Elle permet, quand elle est appellée, d'afficher le menu du générateur de mots de passe.
"""

def afficher_menu(): # On définit une fonction 'afficher_menu' sans paramètre.
    print("================================================")
    print("HAUTOT Nolan | CIEL - Générateur de mots de passe")
    print("================================================")
    print("") # Cette ligne permet de créer un espace entre deux lignes, ce qui rend le menu plu
    print("Quel type de mot de passe souhaitez-vous créer ?")
    print("")
    print("1 - Mot de passe avec configuration par défaut.")
    print("2 - Mot de passe avec configuration personnalisée.")
    print("3 - Phrase de passe avec configuration par défaut.")
    print("4 - Phrase de passe avec configuration personnalisée.")
    print("")
    print("0 - Quitter le menu du générateur.")
    print("")

"""
Docstring de la fonction choix_generateur(choix_utilisateur).

On définit une fonction choix_générateur() avec comme paramètre choix_utilisateur.

On détecte le choix de celui-ci, et on renvoie plusieurs cas avec plusieurs messages personnalis
on génère un mot de passe complètement aléatoire.
"""

def choix_generateur(choix_utilisateurice): # On définit une fonction 'choix_generateur' avec co
    match choix_utilisateurice:
        case 1:
            print(mdp.generer_mdp()) # On appelle la fonction 'generer_mdp' et on l'affiche.
        case 2:
            password_length = int(input("Veuillez entrer une longueur de mot de passe : ")) # On
            print(mdp.generer_mdp(password_length)) # On renvoie son mot de passe en fonction de
```

```
        case 3:
            print(mdp.generer_passphrase())
        case 4:
            nb_words = int(input("Veuillez entrer un nombre de mots à générer : "))
            print(mdp.generer_passphrase(nb_words))
        case 0:
            print("Vous quittez le programme. Au revoir :)")
        case _:
            print(f"Erreur : l'option {choix_utilisateurice} n'existe pas ! Veuillez entrer une

    if choix_utilisateurice!=0:
        input("Appuyez sur 'Entrée' pour reboucler") # Ici dès que l'utilisateur appuie sur 'Ent

while choix_utilisateurice!=0: # Tant que 'choix_utilisateurice' est différent de 0.

    afficher_menu() # On appelle la fonction.

    choix_utilisateurice=int(input("Saisissez votre choix : ")) # On demande le choix, qui sera

    choix_generateur(choix_utilisateurice)
```

6 : Utiliser un dictionnaire et un jet de 5 dés

Code de `mdp.py` :

```python
import secrets # On importe les modules secrets et string.
import string
from eff_words import get_word_list, get_word_dict

LETTRES = string.ascii_letters # Définit toutes les lettres autorisées
CHIFFRES = string.digits # Définit tous les chiffres autorisés
SPECIAL = string.punctuation # Définit les caractères spéciaux autorisés

ALPHABET = LETTRES + CHIFFRES + SPECIAL # Ensemble complet des caractères autorisés
"""
Docstring de la fonction generer_mdp.

Cette fonction va nous permettre de générer un mot de passe 'password' aléatoire avec une certai

On l'initialise avec deux paramètres, password et password_length (qui aura pour valeur par défa

Ensuite on fait une boucle pour générer un mot de passe avec le nombre de caractère que l'on veu

On renvoie un mot de passe avec des caractères aléatoires.
"""

def generer_mdp(password_length=14): # On définit une fonction 'generer_mdp' avec les paramètres
    password = ""
    for _ in range(password_length): # Pour chaque valeur jusqu'à un max de password_length.
        password += secrets.choice(ALPHABET) # On fais un choix aléatoire dans l'alphabet pour
    return password # On renvoie 'password'

"""
Docstring de la fonction generer_passphrase

Cette fonction va nous permettre de générer une phrase de passe avec un nombre de mots aléatoire

On l'initialise avec un paramètre 'nb_words' (qui aura pour valeur par défaut 6)

On créer une liste 'words' qui prend la liste entière de 'eff_words.py'

On créer une liste vide 'passphrase_words'

On répète ensuite 6 fois :

    On prend notre liste et on lui ajoute un mot aléatoire de la liste 'words'

On sépare chaque chaîne de caractère de la liste 'passphrase_words' par un tiret
```

```
On renvoie passphrase_words
"""

def generer_passphrase(nb_words=6):
    words = get_word_list()
    passphrase_words = []
    for _ in range(nb_words):
        passphrase_words.append(secrets.choice(words))
    passphrase_words = "-".join(passphrase_words)
    return passphrase_words

"""
Docstring de la fonction generer_dice_passphrase

On initialise la variable 'words' à laquelle on affecte le dictionnaire 'get_word_dict()'

On initialise une liste passphrase_words vide

On boucle nb_words de fois :

    On appelle la fonction tirer_les_des() qu'on affecte à la variable des

    On ajoute dans la lite passphrase_words les mots correspondant à la valeur 'XXXXX' du mot

On renvoie la passephrase en une chaîne de caractère entière
"""

def generer_dice_passphrase(nb_words=6):
    words = get_word_dict()
    passphrase_words = []
    for _ in range(nb_words):
        des = tirer_les_des()
        passphrase_words.append(words[des])
    return "".join(passphrase_words)

"""
Docstring de la fonction tirer_les_des()

On initialise une liste dice vide

On boucle 5 fois la liste à laquelle on ajoute un chiffre entre 1 et 5
```

```
    On renvoie la liste de chiffre sous forme 'XXXXX'
    """

    def tirer_les_des():
        dice = []
        for _ in range(5):
            dice.append(str(secrets.randbelow(6) + 1))
        return "".join(dice)
```

Code de `main.py` :

```python
import mdp # On importe le module mdp

choix_utilisateurice=-1 # On affecte une valeur non utilisé à la variable pour pouvoir lancer la

"""
Docstring de la fonction afficher_menu().

Ici on définit la fonction afficher_menu() sans paramètre.

Elle permet, quand elle est appellée, d'afficher le menu du générateur de mots de passe.
"""

def afficher_menu(): # On définit une fonction 'afficher_menu' sans paramètre.
    print("================================================")
    print("HAUTOT Nolan | CIEL - Générateur de mots de passe")
    print("================================================")
    print("") # Cette ligne permet de créer un espace entre deux lignes, ce qui rend le menu plu
    print("Quel type de mot de passe souhaitez-vous créer ?")
    print("")
    print("1 - Mot de passe avec configuration par défaut.")
    print("2 - Mot de passe avec configuration personnalisée.")
    print("3 - Phrase de passe avec configuration par défaut.")
    print("4 - Phrase de passe avec configuration personnalisée.")
    print("5 - Phrase de passe avec configuration EFF")
    print("")
    print("0 - Quitter le menu du générateur.")
    print("")

"""
Docstring de la fonction choix_generateur(choix_utilisateur).

On définit une fonction choix_générateur() avec comme paramètre choix_utilisateur.

On détecte le choix de celui-ci, et on renvoie plusieurs cas avec plusieurs messages personnalis
on génère un mot de passe complètement aléatoire.
"""

def choix_generateur(choix_utilisateurice): # On définit une fonction 'choix_generateur' avec c
    match choix_utilisateurice:
        case 1:
            print(mdp.generer_mdp()) # On appelle la fonction 'generer_mdp' et on l'affiche.
        case 2:
            password_length = int(input("Veuillez entrer une longueur de mot de passe : ")) # O
```

```
            print(mdp.generer_mdp(password_length)) # On renvoie son mot de passe en fonction de
        case 3:
            print(mdp.generer_passphrase())
        case 4:
            nb_words = int(input("Veuillez entrer un nombre de mots à générer : "))
            print(mdp.generer_passphrase(nb_words))
        case 5:
            print(mdp.generer_dice_passphrase())
        case 0:
            print("Vous quittez le programme. Au revoir :)")
        case _:
            print(f"Erreur : l'option {choix_utilisateurice} n'existe pas ! Veuillez entrer une

    if choix_utilisateurice!=0:
        input("Appuyez sur 'Entrée' pour reboucler") # Ici dès que l'utilisateur appuie sur 'Ent

while choix_utilisateurice!=0: # Tant que 'choix_utilisateurice' est différent de 0.

    afficher_menu() # On appelle la fonction.

    choix_utilisateurice=int(input("Saisissez votre choix : ")) # On demande le choix, qui sera

    choix_generateur(choix_utilisateurice)
```