Compte Rendu - Itération 2 - Fonctions et modules

Etapes

1 : Fonction `afficher_menu`

Code en Python :

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

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

    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
        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("")

    afficher_menu() # On appelle la fonction.

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

    match choix_utilisateurice:
        case 1:
            print("Mot de passe par défaut : mot2passe!")
        case 2:
            print("Mot de passe personnalisé : M0td3p4ss3#@")
        case 3:
            print("Phrase de passe par défaut : Voici1phrasemdppardefaut!")
        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
```

> *Question 1 :* Où doit être declarée la fonction ?

- La fonction doit être déclarée au tout début du code, afin d'afficher le menu lorsque le programme se lance.

> *Question 2 :* Comment tester que la fonction marche ?

- Pour tester que la fonction marche, on a juste à appeler la fonction dans la console `>>> afficher_menu()` . Si le menu s'affiche, alors la fonction marche sans problème.

2 : Fonction `choix_generateur`

Code en Python de la fonction :

```
def choix_generateur(choix_utilisateurice):
        match choix_utilisateurice:
            case 1:
                print("Mot de passe par défaut : mot2passe!")
            case 2:
                print("Mot de passe personnalisé : M0td3p4ss3#@")
            case 3:
                print("Phrase de passe par défaut : Voici1phrasemdppardefaut!")
            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

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

    choix_generateur(choix_utilisateurice)
```

Code en Python :

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

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

    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
        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("")

    afficher_menu() # On appelle la fonction.

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

    def choix_generateur(choix_utilisateurice): # On définit une fonction 'choix_generateur' ave
        match choix_utilisateurice:
            case 1:
                print("Mot de passe par défaut : mot2passe!")
            case 2:
                print("Mot de passe personnalisé : M0td3p4ss3#@")
            case 3:
                print("Phrase de passe par défaut : Voici1phrasemdppardefaut!")
            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

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

    choix_generateur(choix_utilisateurice)
```

> *Question 1 :* Pourquoi passer la variable `choix_utilisateurice` en paramètre au lieu d'une variable globale ?

- Cela permet à la fonction d'adapter son résultat en fonction du paramètre `choix_utilisateur` uniquement.

> *Question 2 :* Si je modifie `choix_utilisateurice` dans ma fonction, qu'en est-il de la variable `choix_utilisateurice` originale (celle à l'extérieure de la fonction) ?

- La variable sera modifiée uniquement à l'intérieur de la fonction et non à l'extérieur.

3 : Fonction `generer_mdp`

Code en Python de la fonction :

```
def generer_mdp():
    return "P@ssw@rd!"
```

Code en Python de `choix_generateur` après modification :

```
def choix_generateur(choix_utilisateurice): # On définit une fonction 'choix_generateur' avec c
        match choix_utilisateurice:
            case 1:
                print(generer_mdp())
            case 2:
                print("Mot de passe personnalisé : M0td3p4ss3#@")
            case 3:
                print("Phrase de passe par défaut : Voici1phrasemdppardefaut!")
            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

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

Code en Python :

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

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

    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
        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("")

    afficher_menu() # On appelle la fonction.

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

    def generer_mdp(): # On définit une fonction 'generer_mdp' sans paramètre.
        return "P@ssw@rd!"

    def choix_generateur(choix_utilisateurice): # On définit une fonction 'choix_generateur' ave
        match choix_utilisateurice:
            case 1:
                print(generer_mdp()) # On appelle la fonction 'generer_mdp' et on l'affiche.
            case 2:
                print("Mot de passe personnalisé : M0td3p4ss3#@")
            case 3:
                print("Phrase de passe par défaut : Voici1phrasemdppardefaut!")
            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

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

```
choix_generateur(choix_utilisateurice)
```

> *Question 1 :* Est-ce une bonne idée d'utiliser `P@ssw@rd!` comme vrai mot de passe pour un de vos comptes en ligne ? Pourquoi ?

- Non ce n'est pas une bonne idée. C'est un mot de passe qui peut être bruteforce assez facilement. Il suffit pour la personne malveillante de tester plusieurs combinaisons pour retrouver facilement le mot de passe. Un mot de passe plus long, et plus "aléatoire" serait plus efficace.

4 : Création d'un module `mdp`

Code en Python de `mdp.py` :

```
def generer_mdp(): # On définit une fonction 'generer_mdp' sans paramètre.
    return "P@ssw@rd!" # On renvoie "P@ssw@rd en string.
```

Ligne modifiée de `main.py` :

```
print(mdp.generer_mdp()) # On appelle la fonction 'generer_mdp' et on l'affiche.
```

Code en Python 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

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

    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
        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("")

    afficher_menu() # On appelle la fonction.

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

    def choix_generateur(choix_utilisateurice): # On définit une fonction 'choix_generateur' ave
        match choix_utilisateurice:
            case 1:
                print(mdp.generer_mdp()) # On appelle la fonction 'generer_mdp' et on l'affiche
            case 2:
                print("Mot de passe personnalisé : M0td3p4ss3#@")
            case 3:
                print("Phrase de passe par défaut : Voici1phrasemdppardefaut!")
            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

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

```
    choix_generateur(choix_utilisateurice)
```

> *Question 1 :* Quel est l'intérêt de créer des modules séparés pour certaines portions de code ?

- Cela permet d'avoir un code principal beaucoup plus lisible que si l'on avait une tonnes de fonction dans ce code. Grâce aux modules on peut diviser les fonctions en modules et les modifier une par une sans problème et sans se perdre dans notre code.

5 : Modification de `generer_mdp` pour générer un vrai mot de passe

Code en Python de `mdp.py` :

```
import secrets # On importe les modules secrets et string.
import string

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

def generer_mdp(): # On définit une fonction 'generer_mdp' avec le paramètre 'password'
    password = ""
    for i in range(14): # Pour chaque valeur jusqu'à un max de 14 (donc un mot de passe à 14 car
        password += secrets.choice(ALPHABET) # On fais un choix aléatoire dans l'alphabet pour 
    return password # On renvoie 'password'
```

Code en Python 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

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

    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
        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("")

    afficher_menu() # On appelle la fonction.

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

    def choix_generateur(choix_utilisateurice): # On définit une fonction 'choix_generateur' ave
        match choix_utilisateurice:
            case 1:
                print(mdp.generer_mdp()) # On appelle la fonction 'generer_mdp' et on l'affiche
            case 2:
                print("Mot de passe personnalisé : M0td3p4ss3#@")
            case 3:
                print("Phrase de passe par défaut : Voici1phrasemdppardefaut!")
            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

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

```
choix_generateur(choix_utilisateurice)
```

> *Question 1 :* Expliquer en quelques mots le rôle de la bibliothèque `random`

- Le rôle de la bibliothèque `random` est de générer un nombre flottant alétoire entre 0 et 1.

> *Question 2 :* Il y a un avertissement important sur la page, dans l'introduction. A la lecture de cet avertissement, expliquer en quelques mots pourquoi nous avons choisi la bibliothèque `secrets` plutôt que `random` pour générer un mot de passe **sécurisé**.

- Nous avons choisi la bibliothèque `secrets` à la place de `random` à des fins de sécurité. La bibliothèque `random` est considéré comme ne devant pas être utilisée pour la sécurité et qu'il est conseillé d'utiliser la bibliothèque `secrets`

6 : Réutiliser `generer_mdp` pour créer un mot de passe de taille pesonnalisable

Code en Python de `mdp.py` :

```python
import secrets # On importe les modules secrets et string.
import string

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'

def generer_mdp_avec_contrainte():

    return password
```

Code en Python 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("Phrase de passe par défaut : Voici1phrasemdppardefaut!")
            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)
```

> *Question 1 :* Expliquer pourquoi votre choix 1 fonctionne toujours alors que vous ne passez aucun paramètre à la fonction `generer_mdp`

- Le choix 1 fonctionne toujours car on défini la variable password directement dans la fonction, et quand on l'appelle dans le choix 1, elle ne requiert aucun paramètre.

Pour aller plus loin

Bonus facile - Ajout de docstrings

Code en Python de `mdp.py` avec Docstrings :

```python
import secrets # On importe les modules secrets et string.
import string

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'

def generer_mdp_avec_contrainte(password_length=14,special=1,digits=1):
    password = ""
    for _ in range(password_length):

    return password
```

Code en Python de `main.py` avec Docstrings :

```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("Phrase de passe par défaut : Voici1phrasemdppardefaut!")
            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)
```

On rajoute des docstrings pour décrire plus précisément le fonctionnement des fonctions du code.