Correction détaillée de l'exercice - Gestion de fichiers (avec erreurs)

Voici le modèle de correction de l'exercice sur la gestion des fichiers. Des erreurs ont été introduites dans le code pour tester la capacité des étudiants à identifier et corriger ces erreurs.

1. Problème 1 : Lire un fichier et afficher son contenu

Code de l'étudiant avec erreurs :

```python
def lire_fichier(nom_fichier):
    f = open(nom_fichier, 'r')
    for ligne in f:
        print(ligne)  # Affichage du contenu du fichier
    # Oublie de fermer le fichier !
```

Erreur :

- Le fichier n'est pas fermé après avoir été ouvert. Cela peut entraîner des fuites de ressources.

Questions pour l'étudiant :

- Pourquoi est-il important de fermer un fichier après l'avoir ouvert ?
- Comment pourrais-tu corriger cette erreur ?

Correction :

```python
def lire_fichier(nom_fichier):
```

```
    try:
        with open(nom_fichier, 'r') as f:
            for ligne in f:
                print(ligne.strip())  # Affichage du contenu du fichier sans sauts de ligne
    except FileNotFoundError:
        print(f"Erreur : Le fichier {nom_fichier} n'existe pas.")
```

2. Problème 2 : Compter les mots dans un fichier

Code de l'étudiant avec erreurs :

```python
def compter_mots(nom_fichier):
    try:
        with open(nom_fichier, 'r') as f:
            contenu = f.read()
        mots = contenu.split()
        return len(mots)
    except IOError:
        print(f"Erreur : Impossible de lire le fichier {nom_fichier}.")
        return -1  # Retourne -1 en cas d'erreur
```

Erreur :

- Le `IOError` n'est pas le bon type d'exception à capturer pour une erreur de lecture de fichier.

Questions pour l'étudiant :

- Quelle est la différence entre `IOError` et `FileNotFoundError` ?

- Est-ce que tu pourrais capturer une exception plus spécifique ?

Correction :

```python
def compter_mots(nom_fichier):
    try:
        with open(nom_fichier, 'r') as f:
            contenu = f.read()
        mots = contenu.split()
        return len(mots)
    except FileNotFoundError:
        print(f"Erreur : Le fichier {nom_fichier} n'existe pas.")
        return -1  # Retourne -1 en cas d'erreur
```

3. Problème 3 : Écrire dans un fichier

Code de l'étudiant avec erreurs :

```python
def ecrire_dans_fichier(nom_fichier, texte):
    try:
        f = open(nom_fichier, 'a')  # Ouverture en mode ajout
        f.write(texte)
    except IOError:
        print(f"Erreur : Impossible d'écrire dans le fichier {nom_fichier}.")
    # Oublie de fermer le fichier !
```

```

Erreur :

- Le fichier n'est pas fermé après l'écriture. De plus, l'utilisation de `open` sans `with` est risquée.

Questions pour l'étudiant :

- Comment pourrais-tu garantir que le fichier est toujours fermé, même en cas d'exception ?

- Pourquoi est-ce important de fermer un fichier après l'avoir utilisé ?

Correction :

```python
def ecrire_dans_fichier(nom_fichier, texte):
    try:
        with open(nom_fichier, 'a') as f:  # Utilisation de 'with' pour garantir la fermeture du fichier
            f.write(texte)
    except IOError:
        print(f"Erreur : Impossible d'écrire dans le fichier {nom_fichier}.")
```

4. Problème 4 : Fusionner deux fichiers

Code de l'étudiant avec erreurs :

```python
def fusionner_fichiers(nom_fichier1, nom_fichier2, nom_fichier_sortie):
    f1 = open(nom_fichier1, 'r')
    f2 = open(nom_fichier2, 'r')
    f_out = open(nom_fichier_sortie, 'w')
```
```

```
    f_out.write(f1.read())
    f_out.write(f2.read())
    # Oublie de fermer les fichiers ! Et pas de gestion d'exception
```

Erreur :

- Pas de gestion d'exception pour vérifier si les fichiers existent.
- Les fichiers ouverts ne sont pas fermés après leur utilisation.

Questions pour l'étudiant :

- Que se passe-t-il si l'un des fichiers n'existe pas ?
- Comment peux-tu améliorer cette fonction en ajoutant une gestion d'erreur et une fermeture des fichiers ?

Correction :

```python
def fusionner_fichiers(nom_fichier1, nom_fichier2, nom_fichier_sortie):
    try:
        with open(nom_fichier1, 'r') as f1, open(nom_fichier2, 'r') as f2, open(nom_fichier_sortie, 'w') as f_out:
            f_out.write(f1.read())
            f_out.write(f2.read())
    except FileNotFoundError:
        print("Erreur : L'un des fichiers n'existe pas.")
```

5. Problème 5 : Ajouter une gestion des erreurs

Questions pour l'étudiant :

- Pourquoi est-il important de gérer les erreurs dans un programme qui manipule des fichiers ?

- Quelles autres erreurs possibles devrais-tu gérer dans cet exercice ?

Réponse attendue de l'étudiant :

- Il est important de gérer les erreurs pour éviter que le programme plante si un fichier est manquant, que l'utilisateur n'a pas les permissions nécessaires pour accéder au fichier, ou si le fichier est corrompu. L'ajout de messages d'erreur clairs permet à l'utilisateur de comprendre ce qui ne va pas.