TD 1 : Gestion des fichiers en Python

Exercice 1 : QCM

1) Quelles instructions permettent d'ouvrir un fichier en lecture ?
- a) open("fichier.txt","w")
- b) open("fichier.txt","r")
- c) open("fichier.txt")
- d) open("fichier.txt","r+")

2) Quel mode permet d'écrire dans un fichier en écrasant son contenu ?
- a) r
- b) a
- c) w
- d) r+

3) Que fait la méthode tell() ?
- a) ferme le fichier
- b) écrit dans le fichier
- c) retourne la position du curseur dans le fichier
- d) lit une ligne

4) Que fait l'instruction suivante ? f.seek(0)
- a) ferme le fichier
- b) déplace le curseur au début du fichier
- c) supprime le fichier
- d) lit la première ligne

5) Pourquoi doit-on fermer un fichier à la fin du traitement ?
- a) pour supprimer le fichier
- b) pour libérer les ressources du système
- c) pour éviter les erreurs d'accès au fichier
- d) pour déplacer le curseur

6) Que fait read(5) ?
- a) lit 5 lignes
- b) lit 5 caractères
- c) lit tout le fichier
- d) lit la première ligne

7) Quelle est la principale différence entre un fichier texte et un fichier binaire ?
- a) Le fichier texte contient des nombres uniquement
- b) Le fichier binaire contient des caractères uniquement
- c) Le fichier texte contient des caractères lisibles, le fichier binaire contient des données non lisibles directement

8) Quel mode permet d'écrire dans un fichier binaire ?
- a) "w"
- b) "r"
- c) "wb"
- d) "rt"

9) Que retourne read() lorsqu'on lit un fichier binaire ?
- a) une chaîne de caractères
- b) une liste
- c) des bytes (suite d'octets)
- d) un entier

10) Qu'est-ce que la sérialisation ?
- a) Lire un fichier texte
- b) Transformer un objet Python en une forme stockable dans un fichier
- c) Supprimer un fichier
- d) Convertir un nombre en chaîne

11) Quel type d'objet peut être sérialisé avec pickle ?
- a) uniquement les chaînes de caractères
- b) uniquement les listes
- c) plusieurs types d'objets Python (listes, dictionnaires, etc.)
- d) uniquement les nombres

Manipulation des fichiers textes :

Exercice 2 : Exécution d'un script

On dispose d'un fichier texte nommé **noms.txt** (voir Figure 1).

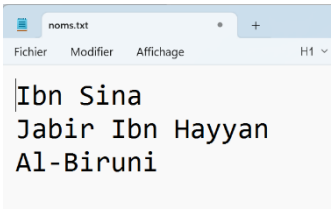

Figure 1 : fichier : noms.txt

Donner le résultat d'exécution du script suivant :

```
f = open("noms.txt", "r")
a = f.readline()
print("A =", a)
b = f.read(5)
print("B =", b)
f.seek(4)
c = f.read(4)
print("C =", c)
d = f.read()
print("D =", d)
f.close()
```

Exercice 3 : Informations sur un fichier

Écrire un script python qui :

1. Ouvre le fichier texte situé au chemin « C:/myfile.txt ».
2. Lit tout le contenu du fichier avec la méthode **read()** puis calcule et affiche :
   - le nombre de caractères
   - le nombre de mots

Exercice 4 : Lire les notes puis afficher et enregistrer les étudiants qui ont validé.

Soit un fichier « **c:/tp/notes.txt** » avec des lignes de la forme suivante :
KARIM 10 12
YASSINE 8 6
MERYEM 11 13
…
Écrire un script Python qui lit le fichier **c:/tp/notes.txt** ligne par ligne, puis, pour chaque ligne, utiliser la méthode **string.split()** afin d'extraire les différentes informations concernant chaque étudiant (nom, note 1, note 2). Le programme doit ensuite calculer la moyenne de chaque étudiant.

Les noms et les moyennes des étudiants ayant une moyenne supérieure ou égale à 10 doivent être affichés à l'écran et également enregistrés dans un nouveau fichier nommé « **c:/tp/resultats.txt** » avec la forme suivante :
KARIM a validé avec 11
MERYEM a validé avec 12
…

Manipulation des fichiers binaires :

Exercice 5 : sérialisation des objets

On souhaite gérer les notes des étudiants et enregistrer les données dans un fichier binaire « etudiants.dat » en utilisant le module pickle.

La liste des étudiants sera représentée sous la forme d'une liste de dictionnaires :

```
listeEtudiants  = [
        {"nom" : "Ali", "note" : 14} ,
        {"nom" : "Sara", "note" : 12} ,
        {"nom" : "Omar", "note" : 9  }
]
```

1. Ecrire une fonction « **lireEtudiant()** » qui demande à l'utilisateur de saisir le nom d'un étudiant et sa note, puis retourne l'étudiant sous forme d'un dictionnaire.
   Exemple : {"nom" : "Ali", "note" : 14}.
2. Ecrire une fonction « **sauvegarder(L)** » qui sérialise et écrit dans le fichier binaire « etudiants.dat » la liste L passée en paramètre.
3. Ecrire une fonction « **charger()** » qui charge et retourne une liste d'étudiants à partir du fichier « etudiants.dat ».
4. Ecrire un programme Python, qui :
   a. Crée une liste vide L.
   b. Demande à l'utilisateur le nombre d'étudiants « **n** » à saisir.
   c. A l'aide d'une boucle **for**, récupère les **n** étudiants et les stocke dans la liste L.
   d. Sauvegarde la liste **L** dans le fichier « etudiants.dat ».
   e. Charge la liste des étudiants dans une autre variable **S**.
   f. Afficher la liste chargée **S**.

Manipulation des fichiers et dossiers avec le module pathlib :

Exercice 6 : Module pathlib

Écrire un programme Python qui réalise les opérations suivantes :

1. Créer un dossier nommé **projet** dans le répertoire courant (s'il n'existe pas).
2. À l'intérieur de ce dossier, créer les fichiers suivants :
   - rapport.txt ; data.log
3. Écrire un contenu différent dans chacun des fichiers.
4. Afficher la liste de tous les fichiers présents dans le dossier.
5. Afficher uniquement les fichiers ayant l'extension .txt.
6. Renommer le fichier **rapport.txt** en **compte_rendu.txt**.
7. Supprimer le fichier **data.log.**
8. Supprimer tous les fichiers restants, puis supprimer le dossier **projet**.

Exercices d'entraînement (travail personnel)

Exercice 7 : Enregistrer la table de multiplication dans un fichier.

Écrire une fonction « **table(n, chemin)** » qui prend en paramètres :

- **n** : un entier dont on veut calculer la table de multiplication
- **chemin** : le chemin (nom ou chemin complet) du fichier texte à créer.

La fonction doit créer (ou écrase s'il existe déjà) le fichier indiqué par **chemin** puis écrit dans ce fichier la table de multiplication du nombre **n** de 1 à 10.

Par exemple pour l'appel de : **table(7, 'table7.txt')**, un fichier table7.txt doit être crée dans le dossier de travail courant et contenir :
7 x 1 = 7
7 x 2 = 14
…
7 x 10 = 70

Exercice 8 : Chiffrement de César.

Le chiffrement de César est une méthode simple de chiffrement qui consiste à décaler chaque lettre d'un message d'un certain nombre de positions dans l'alphabet.

Par exemple, avec un décalage de **3** :
A → D ; B → E ; C → F
Ainsi le mot :
BONJOUR → ERQMRXU

1. Écrire une fonction **chiffrer(message, k)** qui :
   - prend en paramètre une chaîne de caractères message
   - prend un entier k représentant le **décalage**
   - retourne le **message chiffré** avec le chiffrement de César.

   Les règles suivantes doivent être respectées :
   - seules les **lettres alphabétiques** sont décalées
   - les **espaces restent inchangés**
   - le chiffrement se fait **en majuscules**

   Exemple : chiffrer("BONJOUR",3) retourne ERQMRXU

2. On dispose d'un fichier texte **message.txt** contenant un message.

Écrire un **script Python** qui :

1. lit le contenu du fichier **message.txt**
2. chiffre le message avec le chiffrement de César
3. enregistre le résultat dans un nouveau fichier **message_chiffre.txt**

Exemple :
Si **message.txt** contient :
BONJOUR LE MONDE
avec un décalage **k = 3**, le fichier **message_chiffre.txt** doit contenir :
ERQMRXU OH PRQGH