

Gestion des fichiers en Python

Table des matières

I. Pourquoi lire et écrire dans des fichiers	3
A. Fichiers en informatique	3
B. Utilité de gérer les fichiers par programme	3
C. Différentes façons de s'adresser à un fichier par programme.....	4
II. Exercice : Quiz	4
III. Comment accéder aux fichiers depuis un programme ?	5
A. Chemins absolus	5
B. Chemins relatifs	5
C. Chemins UNC.....	6
D. Chemins URL	6
IV. Exercice : Quiz	7
V. Et Python dans tout cela ? Quelles instructions utilise-t-on pour gérer les fichiers ?	8
A. Ouverture des fichiers	8
B. Différentes méthodes d'ouverture et de traitement de fichiers.....	8
C. Opérations sur les fichiers	10
1. Lecture.....	10
2. Écriture	10
D. Codage des fichiers textes	11
VI. Exercice : Quiz	11
VII. Essentiel	12
VIII. Auto-évaluation	12
A. Exercice	12
B. Test.....	12
Solutions des exercices	14

I. Pourquoi lire et écrire dans des fichiers

Contexte

Quel est le but de lire et d'écrire dans des fichiers à partir d'un programme ?

Il est très utile de récupérer le résultat d'un programme dans des fichiers, par exemple pour remplir des tableaux pour stocker des résultats ou simplement éditer un texte ! De même, il est utile de lire le contenu de fichiers pour aller chercher des données nécessaires au programme depuis un tableur, depuis un texte, la lecture d'un mail par exemple, la recherche de mots dans un texte, etc.

D'une manière générale tout ce qui est numérique se stocke dans des fichiers et comme tout ce qui est numérique est traité par programmation. Ceci explique cela !

A. Fichiers en informatique

En informatique tout est fichier ! Fichier texte, fichier exécutable, fichier *backup*, etc. Chaque type de fichier possède son extension : .txt .doc .exe .bck .zip .png .mp3.

Définition

On peut définir deux grands types de fichiers : les fichiers dits textes et les fichiers dits binaires. Les fichiers textes sont ceux dont le contenu est lisible et compréhensible par l'humain, les fichiers binaires sont tous les autres, uniquement interprétés par les machines.

Complément

Ce lien vous expliquera en détail les extensions des fichiers.

Techno-science.net¹

B. Utilité de gérer les fichiers par programme

On se rend bien compte que la gestion des fichiers en informatique est le cœur de l'activité. Écrire un texte tel que celui-ci, sur un ordinateur nécessite un programme, le traitement de texte qui ouvre un fichier vide permet d'écrire dedans, puis de le sauvegarder à l'endroit souhaité avec la bonne extension : .doc, .PDF, etc. Lorsque l'on veut faire une modification dans un fichier texte existant, on clique dessus dans l'explorateur. On exécute de ce fait un programme qui va ouvrir ce fichier et l'afficher à l'écran et nous permettre de modifier son contenu. Une fois notre travail terminé nous le sauvegardons et refermons le fichier.

Remarque

Pour gérer notre fichier, nous avons provoqué une séquence d'ouverture, puis de lecture, puis d'écriture, et enfin de fermeture.

Exemple

Lorsque vous prenez une photo avec votre smartphone, la lumière captée est transformée en pixels et une suite de 1 et de 0, et qui doivent être stockés dans un fichier avec l'extension .jpg. Les mêmes séquences d'ouverture / écriture / fermeture / enregistrement dans le dossier adéquat que pour notre fichier texte sont déclenchées.

¹ <https://www.techno-science.net/definition/7677.html>

Exemple

Lorsque vous voulez écouter votre morceau de musique préféré depuis votre smartphone, vous commandez une séquence d'ouverture d'un fichier ayant l'extension .Wav, par exemple, puis une séquence de lecture du contenu sans autoriser la modification de celui-ci.

C. Différentes façons de s'adresser à un fichier par programme

Comme nous l'avons vu précédemment, il y a plusieurs façons de gérer un fichier, voici ici les principales façons de gérer un fichier :

- Lire sans risque de le modifier ;
- Ouvrir lire / modifier / remplacer le contenu puis sauvegarder et fermer ;
- Ouvrir lire / modifier / ajouter du contenu puis sauvegarder et fermer ;
- Créer / remplir / sauver ;
- Créer s'il n'existe pas ou ouvrir si existe ;
- Créer s'il n'existe pas ou ne rien faire si existe déjà.

Rappel

Techno-science.net¹

Exercice : Quiz

[solution n°1 p.15]

Question 1

On peut classer les fichiers en deux catégories :

- ☐ Les fichiers textes et les fichiers binaires
- ☐ Les fichiers qui peuvent être lus et les fichiers qui peuvent être écrits

Question 2

Il n'y a que les fichiers textes qui peuvent être modifiés (qui acceptent l'écriture).

- ☐ Vrai
- ☐ Faux

Question 3

L'extension d'un fichier détermine son utilité.

- ☐ Vrai
- ☐ Faux

Question 4

Pour lire un fichier il faut l'ouvrir avant.

- ☐ Vrai
- ☐ Faux

¹ <https://www.techno-science.net/definition/7677.html>

Question 5

Parmi les extensions suivantes, indiquez celles qui correspondent à des fichiers textes.

- ☐ .doc
- ☐ .png
- ☐ .xls
- ☐ .ppt

III. Comment accéder aux fichiers depuis un programme ?

A. Chemins absolus

Un chemin absolu est un chemin complet, c'est-à-dire qu'il contient toutes les informations nécessaires pour arriver à destination. En informatique ce chemin part de la racine du support par exemple :

`C:\MesDocuments\mesProgrammesPython\lecturefichier.py`

Ce chemin est unique, il part de « C : » (la racine du support) et arrive au fichier « *lecturefichier.py* » (la destination).

Un chemin absolu (ou encore complet) est toujours vrai et unique.

B. Chemins relatifs

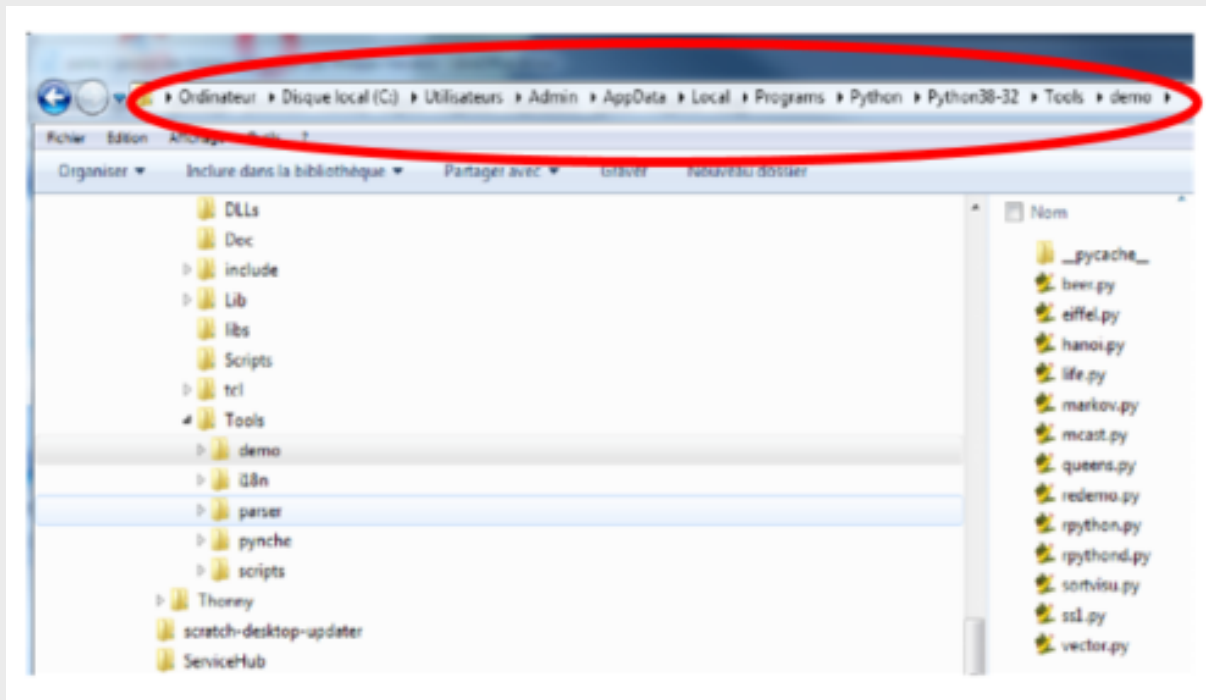
Un chemin relatif part de l'emplacement du programme appelant.

Par exemple, depuis le répertoire `demo` si on va chercher un script `.py` dans le répertoire `lib` son chemin relatif sera :

`..\lib` le répertoire `demo` étant dans le dossier `Tools` lui-même dans `Python38-32`. Le dossier `lib` étant comme `Tools` dans le répertoire `Python38-32`.

Dans l'image ci-dessous, le chemin absolu est entouré en rouge.

Exemple `c:\Utilisateurs\Admin\AppData\Local\Programs\Python\Python38-32\Tools\demo`



Conseil

Attention ! Ne pas mélanger les chemins relatifs et les chemins absolus dans la gestion des fichiers comme nous le verrons après. Par exemple, si vous avez un chemin absolu tel que : `c:\Dossier1\sousdossier2\sousdosier3\fichier.ext`. Si l'on se trouve dans sous-dossier 2 et que l'on travaille avec des chemins relatifs, le chemin retenu devient :

`..\..\Dossier1\sousdossier2\sousdosier3\fichier.ext`

C. Chemins UNC

Définition

Le terme **UNC** signifie **Universal Naming Convention** (convention d'appellation universelle). Une appellation universelle des chemins de fichiers sur un réseau d'ordinateurs.

Un chemin en UNC se présente sous la forme suivante :

`\nomOrdinateur\repertoire_partagé\fichierpartagé`

Remarque

Un chemin UNC ne peut pas remonter dans l'arborescence, ne peut pas contenir de lettre de lecteur (un seul lecteur par chemin).

D. Chemins URL

Définition

Le terme **URL** veut dire **Uniform Resource Locator** (Localisateur de Ressources Uniformes). Cette notation sert à localiser un fichier sur Internet. Le protocole utilisé est HTTP qui fera l'objet d'un cours spécifique.

Exemple

[https://fr.wikipedia.org/wiki/Python_\(langage\)](https://fr.wikipedia.org/wiki/Python_(langage))

En général les documents textes reconnaissent les liens UNC et URL en les notant en bleu.

Exercice : Quiz

[solution n°2 p.16]

Question 1

L'écriture suivante « *E:\python\demo\script1.py* » représente :

- ☐ Un chemin relatif
- ☐ Une URL
- ☐ Un chemin absolu

Question 2

Le chemin suivant « *.\C:* », est correct.

- ☐ Vrai
- ☐ Faux

Question 3

Le répertoire courant est « *D:\dossier1\sousdossier1* » et l'on va chercher le fichier en « *D:\dossier1\sousdossier2\fichieraouvrir.txt* », le chemin est :

- ☐ *..\.\soudossier2*
- ☐ *.\sousdossier2*
- ☐ *..\..\sousdossier2*
- ☐ *../sousdossier2*

Question 4

Comment appelle-t-on un chemin de cette forme « *\ordinateur1\folder\file.txt* » ?

- ☐ absolu
- ☐ URL
- ☐ UNC

Question 5

Le chemin « *c://web.com/site.html* » est correct.

- ☐ Vrai
- ☐ Faux

V. Et Python dans tout cela ? Quelles instructions utilise-t-on pour gérer les fichiers ?

A. Ouverture des fichiers

Pour travailler avec un fichier, la première chose à faire c'est de l'ouvrir. La fonction Python pour cela c'est « *open* », ensuite il faut savoir quel fichier ouvrir et enfin quelle action faire avec ce fichier.

Méthode

La syntaxe est de la forme :

```
resultat = open(filename, mode)
```

ou `with open(filename, mode) as resultat :`

Dans lesquelles :

`resultat` est la variable retour de la fonction `open`, le contenu du fichier y est stocké pour le traitement par python.

`filename` est le nom complet du fichier à ouvrir incluant son chemin (cf. sous partie « *Comment accéder aux fichiers depuis un programme ?* »).

`mode` détermine la méthode d'ouverture du fichier relative à l'action faite avec le fichier. (cf ; paragraphe « *Différentes méthodes d'ouverture et de traitement de fichiers* »).

La fonction `open` crée un lien entre le fichier enregistré sur un support et une variable python où est transférée une image du fichier pour le traiter.

L'avantage d'utiliser `with` est que à la fin de l'action sur le fichier le `close` n'est pas nécessaire.

Complément

Python¹

B. Différentes méthodes d'ouverture et de traitement de fichiers

L'attribut « *mode* » détermine l'usage de la fonction « *open* ». Les différentes utilisations sont décrites dans le tableau suivant :

Fondamental

Caractère utilisé pour l'attribut mode	Description de la fonctionnalité de gestion du fichier ouvert
'r'	Ouvre le fichier en lecture seule (par défaut). « <i>r</i> » comme « <i>read</i> ».
'w'	Ouvre le fichier en écriture, efface le contenu existant du fichier (avant de permettre d'écrire dedans). « <i>w</i> » comme « <i>write</i> ».
'x'	Crée un fichier neuf, si le nom de fichier existe déjà une erreur est générée !

¹ <https://docs.python.org/fr/3/library/functions.html?highlight=open#open>

Caractère utilisé pour l'attribut mode	Description de la fonctionnalité de gestion du fichier ouvert
'a'	Ouvre le fichier en écriture et permet d'ajouter du contenu à la fin du fichier s'il existe. « a » comme « <i>append</i> ».
'b'	Ouvre le fichier en code binaire. « b » comme « <i>binaire</i> ».
't'	Ouvre le fichier en texte (par défaut). « t » comme « <i>texte</i> ».
'+'	Ouvre le fichier en modification (lecture et écriture) une combinaison de « r », « w » et « a ». Permet de modifier un contenu déjà existant ou d'ajouter du contenu en fin de fichier.

Remarque

Il est possible de combiner plusieurs modes, par exemple pour stocker une photo retouchée :

```
new_picture = open(« photoRetouchée.jpg », « xb »)
with open(« photoRetouchée.jpg », « xb ») as new_picture :
```

« x » pour créer une nouvelle image.

« b » car une image est un fichier binaire.

Une fois la gestion du fichier terminée il faut le refermer en utilisant la méthode `close()` de l'objet `file` que nous avons créé :

```
newPicture.close()
```

Remarque

La fermeture libère de précieuses ressources système. Si vous avez oublié de fermer le fichier, Python fermera automatiquement le fichier à la fin du programme ou lorsque l'objet fichier ne sera plus référencé dans le programme. Toutefois, si votre fichier est volumineux et que vous lisez ou écrivez plusieurs fichiers dans le même programme Python, cela peut prendre beaucoup de ressources sur le système. Vous risquez de manquer de ressources et votre programme ne sera pas optimisé. Prenez de bon réflexes de développeur et fermez le fichier dès que vous avez terminé.

Conseil

Pour être sûr de la bonne exécution de la procédure il est possible d'écrire l'instruction de la manière suivante :

Méthode

```
1 if (new_picture = open( " photo_retouche.jpg ", " xb ")) is True :
2     #traitements sur le fichier
3 if new_picture.close() is True :
4     print(« Opération réussie »)
5     # instructions suivantes
```

Avec l'instruction `with`, l'écriture est plus compacte :

```
with open(« photo_retouche.jpg », « xb ») as new_picture :  
# Dans cette méthode après le traitement du fichier l'instruction close n'est pas nécessaire.
```

C. Opérations sur les fichiers

1. Lecture

Pour lire le contenu d'un fichier il faut l'ouvrir avec le mode `r` puis afficher le contenu avec l'instruction `read()` :

Méthode

```
fichier = open('C:\DONNEES\Phyl\Z.txt', 'r')  
print(fichier.read()) # affiche le contenu du fichier Z.txt  
Ceci est le contenu du fichier :  
Z.txt  
fichier.close()  
print(fichier) # affiche les références du fichier Z.txt  
<_io.TextIOWrapper name='C:\\DONNEES\\Phyl\\Z.txt' mode='r' encoding='cp1252'>
```

2. Écriture

Pour écrire dans un fichier, il faut l'ouvrir avec le mode `w`, `x`, `a` ou `+` puis écrire un contenu avec l'instruction `write()` :

Méthode

```
fichier = open('C:\DONNEES\Phyl\Z.txt', 'a')  
fichier.write('ajout de texte dans le fichier') # avec le mode a on ajoute le contenu entre  
guillemet à la fin du fichier Z.txt  
fichier.close() # on referme le fichier après modification  
fichier = open('C:\DONNEES\Phyl\Z.txt', 'r') # on ouvre le fichier pour la lecture  
print(fichier.read()) # on affiche le contenu du fichier modifié  
Ceci est le contenu du fichier :  
Z.txt  
ajout de texte dans le fichier  
fichier.close()
```

Remarque

Pour insérer des valeurs numériques dans un fichier texte par exemple des `int` ou des `float` il faut les convertir en chaîne de caractères avec la fonction `str()`. De même si on veut indexer les noms de fichiers avec la méthode « `w` » on peut concaténer la date dans le nom du fichier. De cette façon, en ayant au préalable importé la date et l'heure de l'instant dans la variable `dateheure` :

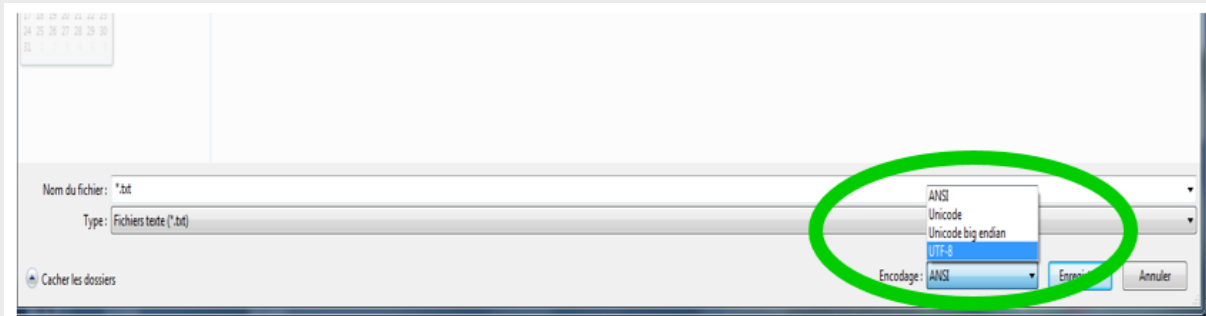
```
fichier = open('C:\DONNEES\Phyl\Z'+str(dateheure)+'.txt', 'w')
```

D. Codage des fichiers textes

Les fichiers textes ont plusieurs façons d'être codés. La plus courante est `utf-8` mais il existe plusieurs types d'encodage.

Exemple

Lorsque l'on veut enregistrer un fichier Bloc-notes, il nous est proposé plusieurs encodages.



```
f = open("etudiants.txt", "r", encoding = 'utf-8')
```

Nous avons vu dans le cours « *Écriture* » que l'instruction :

```
print(fichier) # affiche les références du fichier Z.txt
```

```
<_io.TextIOWrapper name='C:\\DONNEES\\Phyl\\Z.txt' mode='r' encoding='cp1252'>
```

Cela nous donne des informations sur l'encodage du fichier.

Si nous voulons définir un encodage spécifique lors de l'ouverture d'un fichier il est possible de faire :

Méthode

```
f = open("monFichiercode.txt", "w", encoding = 'utf-8')
```

Exercice : Quiz

[solution n°3 p.17]

Question 1

Pour ouvrir un fichier en lecture seule avec Python, on utilise :

- ☐ `openfile()`
- ☐ `readfile()`
- ☐ `writefile()`
- ☐ `open()`

Question 2

`FichierImg = open('image.jpg', 'r')` permet de :

- ☐ Ouvrir `image.jpg` pour retouche
- ☐ Ouvrir `image.jpg` et l'afficher
- ☐ Ouvrir `image.jpg` dans `fichierImg`

Question 3

Après avoir ouvert un fichier avec la méthode « `r` » dans la variable `aff_fichier`, l'instruction **`print(aff_fichier)`** affiche :

- ☐ Le contenu du fichier
- ☐ Les références du fichier
- ☐ Une erreur

Question 4

Après avoir utilisé l'instruction **`fichier.close()`**, on peut afficher le contenu de fichier.

- ☐ Vrai
- ☐ Faux

Question 5

L'instruction `open('fichier.txt', 'wab')` est correcte.

- ☐ Vrai
- ☐ Faux

VII. Essentiel

Il y a deux grandes familles de fichiers, les « *textes* », lisibles par l'humain, et les « *binaires* », compréhensibles par les machines. La gestion de fichiers consiste à les ouvrir pour les lire, écrire dedans à la création ou pour modification. Pour accéder à un fichier il faut connaître son chemin relatif depuis là où on l'appelle, ou absolu depuis la racine du support. En Python la fonction `open` permet d'ouvrir un fichier. La finalité est définie par la méthode, lecture, écriture ou les deux.

VIII. Auto-évaluation

A. Exercice

Dans un jeu vidéo, à la fin de chaque partie gagnée ou perdue, on demande au joueur de renseigner son nom, on lui affecte son score et le tout est stocké à la fin du fichier des scores.

Attention : le joueur ne peut pas modifier ce qui est déjà renseigné dans le fichier ! Il ne peut pas falsifier les scores ou les noms des joueurs précédents ! Ce fichier est affiché à chaque début de nouvelle partie pour que le joueur connaisse le score à battre !

Par contre, si le jeu a été réinitialisé, ce fichier est alors effacé avant de le renseigner à nouveau.

Le programme de gestion se trouve dans le répertoire « *process* », le fichier des scores sera stocké dans le répertoire « *Data* » qui se trouve dans le même répertoire que « *process* ».

Question 1

[solution n°4 p.18]

Quelles seront toutes les caractéristiques données par `print(score)` du fichier dans lequel seront stockés les scores, et pourquoi ?

Question 2

[solution n°5 p.18]

Comment faire pour que les joueurs ne puissent pas modifier le contenu du fichier ne les concernant pas quand ils écrivent leur propre score dans ce fichier ouvert ?

B. Test

Dans un jeu vidéo, à la fin de chaque partie gagnée ou perdue, on demande au joueur de renseigner son nom, on lui affecte son score et le tout est stocké à la fin du fichier des scores.

Attention : le joueur ne peut pas modifier ce qui est déjà renseigné dans le fichier ! Il ne peut pas falsifier les scores ou les noms des joueurs précédents ! Ce fichier est affiché à chaque début de nouvelle partie pour que le joueur connaisse le score à battre !

Par contre, si le jeu a été réinitialisé, ce fichier est alors effacé avant de le renseigner à nouveau.

Le programme de gestion se trouve dans le répertoire « *process* », le fichier des scores sera stocké dans le répertoire « *Data* » qui se trouve dans le même répertoire que « *process* ».

Exercice 1 : Quiz

[solution n°6 p.18]

Question 1

Quel est l'intrus ?

- ☐ Chemin absolu
- ☐ Chemin virtuel
- ☐ Chemin URL

Question 2

Un fichier avec l'extension .ZIP est un fichier :

- ☐ Binaire
- ☐ Texte

Question 3

`Open('..\image.png', 'r')` veut dire que :

- ☐ On ouvre le fichier image.png dans le répertoire parent
- ☐ On ouvre le fichier image.png dans le répertoire parent du parent
- ☐ On ouvre le premier fichier trouvé qui s'appelle image.png sur le support

Question 4

Cette instruction est correcte.

```
resultat=open("fichier.mp3", "r")
print(resultat.read())
resultat.close()
```

- ☐ Vrai
- ☐ Faux

Question 5

Après avoir ouvert un fichier avec la méthode « *w* » dans la variable `ModifFichier`, l'instruction **`print(modif_fichier)`** affiche :

- ☐ Le contenu du fichier
- ☐ Les références du fichier
- ☐ Une erreur

Question 6


L'écriture suivante « *E:html\python\demo\script1.py* » représente une URL.

- ☐ Vrai
- ☐ Faux

Solutions des exercices


Exercice p. 4 Solution n°1**Question 1**

On peut classer les fichiers en deux catégories :

- ☒ Les fichiers textes et les fichiers binaires
- ☐ Les fichiers qui peuvent être lus et les fichiers qui peuvent être écrits
-  Les fichiers textes sont ceux que l'humain peut lire. Les fichiers binaires sont ceux que les machines interprètent.


Question 2

Il n'y a que les fichiers textes qui peuvent être modifiés (qui acceptent l'écriture).

- ☐ Vrai
- ☒ Faux
-  Tous les fichiers sont modifiables par programmes.


Question 3

L'extension d'un fichier détermine son utilité.

- ☒ Vrai
- ☐ Faux
-  Chaque type de fichier a au moins une extension dédiée.

Question 4


Pour lire un fichier il faut l'ouvrir avant.

- ☒ Vrai
- ☐ Faux
-  Un fichier c'est comme un cahier pour lire ce qu'il y a dedans il faut l'ouvrir.

Question 5

Parmi les extensions suivantes, indiquez celles qui correspondent à des fichiers textes.

- ☒ .doc
- ☐ .png
- ☒ .xls
- ☒ .ppt

-  .doc est l'extension d'un fichier document texte, type word.
- .xls est l'extension d'un fichier tableur, type Excel.
- .ppt est l'extension d'un fichier de présentation, type PowerPoint.

Q .png est l'extension d'un fichier image.

Exercice p. 7 Solution n°2

Question 1

L'écriture suivante « *E:\python\demo\script1.py* » représente :

- ☐ Un chemin relatif
- ☐ Une URL
- ☒ Un chemin absolu
- Q Un chemin absolu démarre de la racine du lecteur, ici « E:\... ».

Question 2

Le chemin suivant « *.\C:* », est correct.

- ☒ Vrai
- ☐ Faux
- Q Le répertoire courant est dans la racine d'un autre disque et l'on vient dans la racine du disque « C:\. ».

Question 3

Le répertoire courant est « *D:\dossier1\sousdossier1* » et l'on va chercher le fichier en « *D:\dossier1\sousdossier2\fichierouvrir.txt* », le chemin est :

- ☐ *..\.\soudossier2*
- ☐ *.\sousdossier2*
- ☒ *..\..\sousdossier2*
- ☐ *../sousdossier2*
- Q On remonte d'un niveau dans l'arborescence depuis le répertoire courant et on redescend dans le sous-répertoire voulu.


Question 4

Comment appelle-t-on un chemin de cette forme « *\ordinateur1\folder\file.txt* » ?

- ☐ absolu
- ☐ URL
- ☒ UNC
- Q Ce genre de chemin est un peu comme un chemin absolu mais pour des ordinateurs en réseau (UNC veut dire Universal Naming Convention ou convention d'appellation universelle en français).

Question 5


Le chemin « *c://web.com/site.html* » est correct.

- ☐ Vrai
- ☒ Faux
-  Lorsque l'on utilise des « / » au lieu de « \ » c'est un chemin URL pour le web.

Exercice p. 11 Solution n°3


Question 1

Pour ouvrir un fichier en lecture seule avec Python, on utilise :

- ☐ `openfile()`
- ☐ `readfile()`
- ☐ `writefile()`
- ☒ `open()`
-  C'est bien la fonction `open()` qu'il faut utiliser.


Question 2

`FichierImg = open('image.jpg', 'r')` permet de :

- ☐ Ouvrir `image.jpg` pour retouche
- ☐ Ouvrir `image.jpg` et l'afficher
- ☒ Ouvrir `image.jpg` dans `fichierImg`
-  La fonction `open` permet d'ouvrir un fichier informatique et de charger son contenu dans une variable de travail ici `fichierImg`.


Question 3

Après avoir ouvert un fichier avec la méthode « `r` » dans la variable `aff_fichier`, l'instruction **`print(aff_fichier)`** affiche :

- ☐ Le contenu du fichier
- ☒ Les références du fichier
- ☐ Une erreur
-  L'instruction `print(affFichier)` affichera les références du fichier. Pour afficher son contenu il faut utiliser `print(affiFichier.read())`.

Question 4

Après avoir utilisé l'instruction **`fichier.close()`**, on peut afficher le contenu de fichier.


- ☐ Vrai
- ☒ Faux
-  Une fois le fichier fermé son contenu n'est plus accessible.

Question 5

L'instruction `open('fichier.txt', 'wab')` est correcte.

☐ Vrai

☒ Faux

 Cette instruction n'est pas correcte car les méthodes « *w* » et « *a* » ne sont pas compatibles. « *w* » efface le contenu du fichier ouvert pour pouvoir en mettre un autre, alors que « *a* » permet d'ajouter du contenu à la fin du fichier ouvert.

p. 12 Solution n°4

Le fichier prévu pour recevoir les enregistrements des noms des joueurs et de leur score respectif sera un fichier de type texte, car il doit pouvoir être lu par les joueurs avant le commencement de leur jeu et renseigné par eux à la fin de leur partie. Il sera encodé en UTF-8, la norme unicode universelle la plus courante. Le chemin relatif du fichier sera : « `..\Data\` ». Le résultat de l'instruction : `print(score)` devra donner les informations du genre :

```
<_io.TextIOWrapper name='..\Data\scores.txt' mode='r' encoding='UTF-8'>
```

En début de programme (appel avant chaque partie).

Et du genre :

```
<_io.TextIOWrapper name='..\Data\scores.txt' mode='a' encoding='UTF-8'>
```

En fin de partie.

p. 12 Solution n°5

L'astuce pour que les joueurs qui renseignent leur score dans le fichier ouvert en fin de partie est d'utiliser la méthode « *a* », pour qu'ils ne puissent pas modifier le reste du fichier et qu'ils n'aillent pas directement écrire dans ce fichier ! En effet, en stockant dans une variable chaîne de caractères leur nom, puis ensuite écrire cette chaîne dans le fichier lorsqu'ils valident. On évite toute modification intempestive du fichier.

Le code serait de la forme suivante :

```
nom_joueur = str(input('entrez votre nom:'))
score = open('..\Data\scores.txt', 'a')
fichier.write(nom_joueur)
fichier.close()
```

Exercice p. 13 Solution n°6


Question 1

Quel est l'intrus ?

☐ Chemin absolu

☒ Chemin virtuel

☐ Chemin URL


 Le type « chemin virtuel » n'existe pas. Il y a les chemins relatifs, chemins absolus , chemins URL, chemins UNC.

Question 2

Un fichier avec l'extension .ZIP est un fichier :

☒ Binaire

☐ Texte

 Un fichier .ZIP est un fichier compressé qui est compris par les machines mais pas lisible directement par l'humain. Il est donc binaire.


Question 3

`Open('..\image.png', 'r')` veut dire que :

☒ On ouvre le fichier image.png dans le répertoire parent

☐ On ouvre le fichier image.png dans le répertoire parent du parent

☐ On ouvre le premier fichier trouvé qui s'appelle image.png sur le support

 Ici, il s'agit d'un chemin relatif et « ..\ » veut dire que l'on va chercher dans le répertoire parent.

Question 4

Cette instruction est correcte.


```
resultat=open("fichier.mp3", "r")
```

```
print(resultat.read())
```

```
resultat.close()
```

☒ Vrai

☐ Faux

 Tout est correct : On ouvre un fichier binaire en lecture seule, puis on demande son affichage et enfin il est refermé.


Question 5

Après avoir ouvert un fichier avec la méthode « w » dans la variable `ModifFichier`, l'instruction **`print(modif_fichier)`** affiche :

☐ Le contenu du fichier

☒ Les références du fichier

☐ Une erreur

 Cette instruction permet d'ouvrir les différentes références du fichier. Les références donnent des informations d'identification du fichier.

Question 6

L'écriture suivante « `E:html\python\demo\script1.py` » représente une URL.

☐ Vrai

☒ Faux

Q L'URL ou Uniform Resource Locator est l'adresse d'un site web, elle permet à l'utilisateur d'y accéder via un navigateur web.