Un fichier stocke des informations sur un support physique (disque dur, clé USB, CD, DVD, carte mémoire SD...). Ouvrir un fichier consiste à le charger dans la mémoire vive (RAM) de l'ordinateur (c'est une mémoire volatile : elle s'efface quand on éteint l'ordinateur). Enregistrer un fichier consiste à l'écrire sur un support physique de stockage (l'information est alors conservée de manière permanente). Il existe deux types de fichiers :

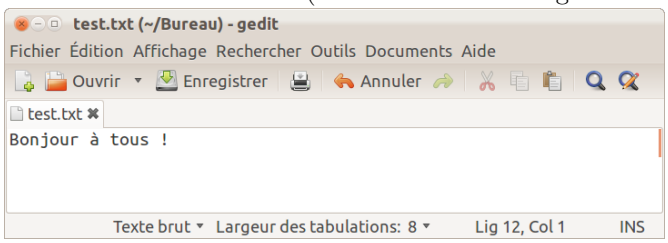
- Les **fichiers textes** : l'information est stockée sous forme de caractères lisibles par un éditeur de texte (principalement des lettres et des chiffres).
- Les **fichiers binaires** : l'information est stockée en binaire (une suite d'octets dont la valeur est comprise entre 0x00 et 0xFF).

### Ecriture dans un fichier

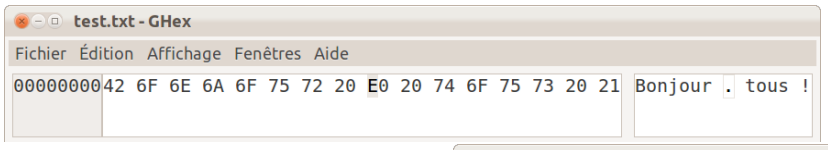
**Le mode write** L'écriture dans un fichier se fait avec la fonction `open()` en mode écriture :

```
# script lecture.py

NomFichier = 'test.txt'
# création et ouverture du fichier test.txt en mode write 'w' (écriture)
# si le fichier test.txt existe déjà, il est écrasé
Fichier = open(NomFichier,'w')      # instanciation de l'objet Fichier de la classe file

# écriture dans le fichier avec la méthode write()
Fichier.write('Bonjour à tous !')

# fermeture du fichier avec la méthode close()
Fichier.close()
```

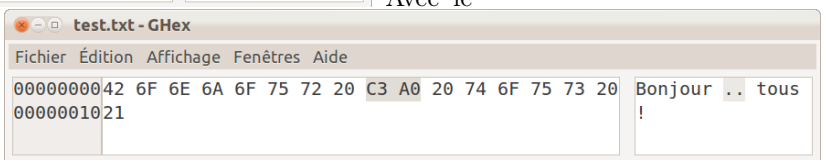
Le fichier `test.txt` est créé dans le répertoire courant (sous Windows c'est normalement le répertoire `C:/PythonXX`, mais on peut aussi choisir un emplacement quelconque en spécifiant le chemin absolu, par exemple `NomFichier = 'F:/Mon dossier/test.txt'`). Vous pouvez vérifier son contenu en l'ouvrant avec un éditeur de texte (ou même avec un logiciel de traitement de texte) :




**Remarques sur le codage des caractères** Pour Python, le codage par défaut des caractères est cp1252 (Windows-1252) sous Windows, et UTF-8 sous GNU/Linux. Avec le codage cp1252, le fichier fait 16 octets :




Avec le

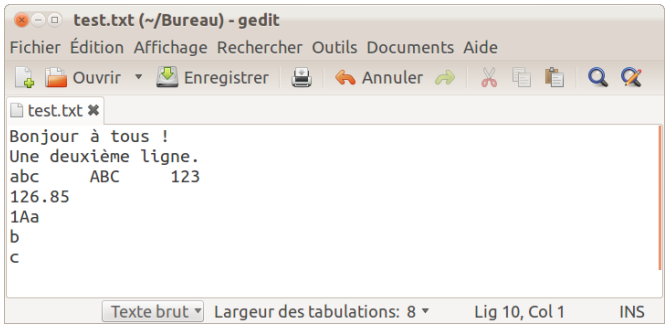



codage UTF-8, le fichier fait 17 octets !
La différence est due au codage du caractère accentué **à** : 1 octet (0xE0) pour le codage cp1252 et 2 octets (0xC3 0xA0) pour le codage UTF-8. Les autres caractères ont un codage commun sur un octet (codage ASCII).

**Le mode append** Pour écrire à la fin d'un fichier, on utilise la fonction `open()` en mode ajout. Repartons du fichier précédent :

```
# ouverture du fichier test.txt en mode append 'a' (ajout)
Fichier = open('test.txt','a')      # instanciation de l'objet Fichier
Fichier.write('\nUne deuxième ligne.\n')# '\n' saut de ligne
Fichier.write('abc\tABC\t123\n')    # '\t' tabulation
Fichier.write(str(126.85)+'\n')     # str() transforme un nombre en chaîne de caractères
Fichier.write('\x31\x41\x61\n')     # écriture de '1Aa' en code ASCII
Fichier.write(chr(0x62)+'\n')       # écriture de 'b' en code ASCII
Fichier.write(chr(99))          # écriture de 'c' en code ASCII
Fichier.close()
```

Ce qui donne :




### Lecture dans un fichier

**Lecture en mode texte** La lecture dans un fichier texte se fait avec la fonction `open()` en mode ... lecture :

```
# ouverture du fichier test.txt en mode read 'r' (lecture en mode texte)
Fichier = open('test.txt','r')      # instanciation de l'objet Fichier de la classe file
# lecture dans le fichier avec la méthode read()
```

```
chaine = Fichier.read()
# affichage du contenu du fichier
print('Contenu du fichier :\n' + chaine)
# fermeture du fichier avec la méthode close()
Fichier.close()
```

```
>>>
Contenu du fichier :
Bonjour à tous !
Une deuxième ligne.
abc ABC 123
126.85
1Aa
b
c
```

**Lecture en mode binaire** En mode texte, un fichier est lu caractère par caractère. En mode binaire, un fichier est lu octet par octet : c'est un peu différent ! Par exemple, si on veut la taille d'un fichier, il faut utiliser la lecture en mode binaire 'rb' (read binary) :

```
# ouverture du fichier test.txt en mode read binary 'rb' (lecture en mode binaire)
Fichier = open('test.txt','rb')
data = Fichier.read()
# affichage de la taille du fichier
print('Taille du fichier :',len(data),'octets')
# affichage du contenu (en hexadécimal)
import binascii
print(binascii.hexlify(data))
# fermeture du fichier avec la méthode close()
Fichier.close()
```

```
>>>
Taille du fichier : 69 octets
b'426f6e6a6f757220e020746f757320210d0a556e65206465757869e86d65206c69676e652e0d0a61626309414
```

La taille du fichier est 69 octets avec le codage cp1252 (sous Windows) et 65 octets avec le codage UTF-8 (sous Linux). Avec la lecture en mode texte 'r', on obtient 63 octets avec le codage cp1252 (sous Windows) et 65 octets avec le codage UTF-8 (sous Linux) ! Pourquoi ce décalage ? Car sous Windows, un saut de ligne est codé sur 2 octets (0x0D 0x0A) et le caractère spécial d'une chaîne `\n` sur un seul octet ! Sous Linux, c'est plus simple avec un saut de ligne codé sur un seul octet (0x0A). En définitive : 63 + 2 (2 caractères accentués **à** et **è**) = 65 63 + 6 (6 sauts de ligne) = 69

### Sérialisation des données avec le module Pickle

Le module `Pickle` est extrêmement pratique pour sauvegarder dans un fichier des structures de données comme les listes (type `list`) ou les dictionnaires (type `dict`).

**Pickling** Un exemple de sauvegarde d'un dictionnaire :

```
import pickle

# création d'un dictionnaire
departement = {36:'Indre',30:'Gard',75:'Paris'}

# enregistrement du dictionnaire dans un fichier
Fichier = open('data.txt','wb')
pickle.dump(departement,Fichier)    # sérialisation
Fichier.close()
```

**Unpickling** L'opération inverse est tout aussi simple :

```
import pickle

# récupération du dictionnaire
Fichier = open('data.txt','rb')
Dept = pickle.load(Fichier)    # désérialisation
Fichier.close()

print(Dept)
print(Dept[36])
```

```
>>>
{75: 'Paris', 36: 'Indre', 30: 'Gard'}
Indre
```

### Exercices

**Exercice 8.1 : fichiers binaires et fichiers textes** Les fichiers suivants sont-ils de type binaire ou de type texte ?

- Une photo au format JPEG
- Une image au format PNG
- Une image au format SVG
- Un fichier audio au format MP3
- Un document texte au format TXT
- Une page web au format HTML
- Un fichier exécutable (extension .exe)
- Un document au format PDF
- Un fichier compressé au format ZIP

- Un script en langage Python (extension .py)

**Exercice 8.2** 1) Créer un fichier texte contenant une suite aléatoire de chiffres. On utilisera la fonction `randint()` du module `random`.
Exemple
2) Avec un logiciel tel que **7-Zip**, compresser le fichier avec différents algorithmes (zip, gzip, bzip2).
Le taux de compression (en %) est donné par la formule : 100×(taille initiale - taille après compression)/taille initiale
Comment expliquer ce taux de compression modeste (au mieux 57 %)?

**Projet**

**Election des délégués de classe par un vote électronique** Ecrire un ensemble de scripts qui gère une élection par vote électronique.
On utilisera des fichiers pour stocker la liste électorale, les mots de passe, la liste des candidats et le résultat des votes.

Pour une sécurité optimale, le vote se fera sur un ordinateur seul (sans connexion réseau, sans connexion à Internet...).

Exemple d'interface :

```
>>>
Election des délégués de classe

Actuellement, 18 élèves ont voté (sur 30)

Identifiant ? martin
Mot de passe ?

L'authentification a réussi.

Electeur : Martin Rémi

Liste des candidats :

Durand Yohan    --> 0
Barot Pauline   --> 1
Dupont Simon    --> 2

Vote blanc      --> 3

Quel est votre choix ? 1
Confirmer votre choix : 1
```

```
A voté !
Merci et bonne journée.

>>>
Résultat de l'élection des délégués de classe

Nombre de votants : 30 / 30

Durand Yohan    -->  5 voix (17.9 %)
Barot Pauline   -->  16 voix (57.1 %)
Dupont Simon    -->  7 voix (25.0 %)

Vote blanc      -->  2 voix
```

**Webographie**

- Documentation sur la fonction open()
- Input and Output : Reading and Writing Files
- Documentation sur le module Pickle
- Le jeu de caractères ASCII
- Le jeu de caractères Unicode UTF-8
- Le jeu de caractères Windows-1252
- 7-Zip un logiciel libre de compression et d'archivage (pour Windows)